



UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento di Ingegneria Gestionale, dell'Informazione e
della Produzione

Corso di Laurea in Ingegneria delle Tecnologie per la Salute
Classe n. L-9 – Classe delle Lauree in Ingegneria Industriale

Automatizzazione dello stress test di un tavolo per macchina raggi X

Candidato:
Giorgio Gandolfi
Matricola n. 1054545

Relatore:
Chiar.ma. Prof.ssa Elisa Riceputi

Anno Accademico
2020/2021

Indice

Abstract	7
1 Introduzione	9
2 Piattaforme hardware	11
2.1 Arduino Uno	11
2.1.1 Topologia e pinout della scheda	12
2.1.2 Connessione con la macchina a raggi X	14
2.1.3 Sensore di inclinazione a sfera	15
2.1.4 Problematiche del sensore di inclinazione a sfera	17
2.2 Arduino Nano 33 BLE Sense	17
2.2.1 Topologia e pinout della scheda	18
2.2.2 Accelerometro	20
2.2.3 Giroscopio	22
2.2.4 Vantaggi dell'utilizzo della board	23
3 Protocolli di comunicazione	25
3.1 Comunicazione seriale	25
3.1.1 Il protocollo UART	25
3.2 Bluetooth®	28
3.2.1 Bluetooth® Low Energy	29
3.2.2 Connessione point-to-point	30
3.2.3 Profili generici del protocollo	31
3.2.4 Advertising e Scan Response Data	32
3.2.5 Transazioni GATT	33

3.2.6	Servizi e caratteristiche	34
4	Software	37
4.1	Firmware Arduino Uno	37
4.1.1	Struttura del software	37
4.1.2	Convenzione della struttura dei comandi	38
4.1.3	Inizializzazione e setup della piattaforma	39
4.1.4	Funzione loop	39
4.1.5	Elaborazione dei comandi	40
4.1.6	Stop di emergenza del macchinario	40
4.2	Firmware Arduino Nano	41
4.2.1	Librerie utilizzate	41
4.2.2	Design dei servizi e delle caratteristiche	42
4.2.3	Dichiarazione dei servizi e delle caratteristiche	42
4.2.4	Inizializzazione e setup della piattaforma	43
4.2.5	Funzione loop	44
4.2.6	Lettura ed invio dati	45
4.3	Applicativo Qt	46
4.3.1	Interfaccia del programma	46
4.3.2	Struttura del programma	48
4.3.3	Macchina a stati finiti	49
4.3.4	Struttura del file contenente i comandi di movimento	51
4.3.5	Gestione della connessione seriale	52
4.3.6	Gestione della connessione Bluetooth®	53
4.3.7	Raccolta dati	54
4.3.8	Verifica del movimento del macchinario	58
4.3.9	Gestione del conteggio dei cicli effettuati	60
4.3.10	Stop di emergenza del macchinario	60
5	Conclusioni e sviluppi futuri	61
5.1	Conclusioni	61
5.2	Sviluppi futuri	62

Elenco delle figure

2.1	Arduino Uno	12
2.2	Arduino Uno: layout del top della piattaforma	13
2.3	Pinout della scheda	14
2.4	Arduino Uno con modulo relè	15
2.5	Sensore a sfera	15
2.6	Funzionamento del sensore di inclinazione a sfera	16
2.7	Arduino Nano 33 BLE Sense	17
2.8	Arduino Nano: layout del top della piattaforma	18
2.9	Pinout della scheda	20
2.10	Gli assi dell'accelerometro	21
2.11	Schematizzazione dell'accelerazione	21
2.12	Gli assi del giroscopio	22
2.13	Schematica per l'alimentazione della board	23
3.1	Schematizzazione della connessione UART	26
3.2	Pacchetto dati UART	27
3.3	Panoramica del Bluetooth® Classic	28
3.4	Schematizzazione di una rete piconet	29
3.5	Panoramica del Bluetooth®Low Energy	30
3.6	Schema della connessione point-to-point	31
3.7	Rappresentazione del processo di advertising	32
3.8	Rappresentazione di una transazione dati utilizzando il profilo GATT	33
3.9	Rappresentazione di servizi e caratteristiche	34

4.1	Pagina iniziale del programma	47
4.2	Pagina iniziale con barra laterale dei menù	47
4.3	Pagina di setup	48
4.4	Pagina iniziale con pulsante di stop	48
4.5	Grafo della macchina a stati finiti	50
4.6	Andamento nei tre assi dell'accelerazione di gravità lungo un ciclo	54
4.7	Progressione dell'angolo α nel tempo calcolata con l'accelerometro	55
4.8	Andamento nei tre assi della velocità angolare lungo un ciclo	56
4.9	Progressione dell'angolo α nel tempo calcolata con il giroscopio	57
4.10	Menù di scelta del sensore	58

Elenco delle tabelle

2.1	Componenti principali della scheda Arduino Uno	13
2.2	Componenti della scheda Arduino Nano 33 BLE Sense	19
3.1	Proprietà delle caratteristiche Bluetooth® Low Energy	35

Abstract

Il presente progetto di tesi tratta l’automazione del movimento del tavolo di una macchina per radiografie, in modo da testare i componenti del sistema che generano e trasmettono tale movimento, come il motore passo-passo e la catena, per un numero molto elevato di cicli.

Per raggiungere l’obiettivo prefissato, è stato necessario sviluppare un applicativo creato ad hoc. L’applicativo è stato sviluppato tramite l’utilizzo di Qt Creator, utilizzando C++ come linguaggio di programmazione.

Tramite comunicazione seriale, l’applicativo si connette ad una piattaforma Arduino Uno, la quale manda i comandi di movimento da far eseguire al motore stesso, sostituendo la cloche normalmente addetta al movimento del tavolo radiografico.

L’aggiunta di un sensore a pallina posizionato sul macchinario che comunica i dati direttamente alla piattaforma Arduino Uno, fornisce informazioni sull’avvenuta inclinazione o meno del tavolo stesso. Grazie all’utilizzo combinato dell’applicativo, della scheda Arduino Uno e del sensore a pallina, si è quindi in grado di far muovere il macchinario, di verificare l’avvenimento effettivo del movimento e di contare il numero di cicli effettuati.

Il progetto di tesi prosegue con la sostituzione del sensore a pallina tramite una piattaforma Arduino Nano 33 BLE Sense. Questa scheda monta diversi sensori integrati, tra cui accelerometro e giroscopio. Inoltre, monta un modulo Bluetooth® Low Energy che rende possibile la comunicazione wireless dei dati. Posizionando questo dispositivo sul tavolo radiogeno è dunque possibile quantificare l’effettiva inclinazione del tavolo stesso.

Capitolo 1

Introduzione

L'efficienza dell'utilizzo delle risorse è ricercata da tutte le aziende, in modo da massimizzare il contributo economico che queste apportano alle imprese.

Il lavoro svolto con questo progetto di tesi punta all'aumento dell'efficienza di parte del processo produttivo delle macchine a raggi X, attraverso l'automatizzazione dei test svolti sulle componenti che permettono al tavolo radiogeno di muoversi.

Grazie all'automatizzazione di questo processo, è possibile sottoporre la macchina e le sue componenti a prove di fatica, per un numero molto elevato di cicli, in maniera automatica, senza bisogno di personale che segua tutto il processo.

Ne consegue dunque un risparmio per l'azienda: in termini economici in quanto non è necessario avere dipendenti che si occupino principalmente di seguire i test che vengono eseguiti; in termini di tempo poiché è possibile eseguire questi test in maniera efficiente limitando al minimo le pause e le interruzioni.

Lo stato dell'arte consiste attualmente nell'impiego di personale incaricato di seguire gli stress test effettuati ai macchinari. Una volta ultimato il progetto, grazie all'aumento dell'efficienza degli stress test dovuto all'automatizzazione di questi, sarà possibile ridurre sensibilmente il tempo che il personale dedica seguendo l'andamento delle prove.

Nei successivi capitoli verranno presentate le piattaforme hardware Arduino Uno ed Arduino Nano 33 BLE Sense, utilizzate per il progetto. Si proseguirà poi con l'esposizione dei protocolli di comunicazione seriale e Bluetooth® Low Energy. Infine, verrà descritta la parte software scritta per entrambe le piattaforme Arduino e l'applicativo per computer sviluppato con Qt Creator.

Capitolo 2

Piattaforme hardware

In questo capitolo verranno presentate le due piattaforme programmabili Arduino Uno ed Arduino Nano 33 BLE Sense, utilizzate nel progetto. Verranno descritti il processo decisionale che ha portato alla scelta delle stesse, le loro principali caratteristiche e tutte le connessioni fisiche necessarie per il funzionamento dell'apparato.

2.1 Arduino Uno

Un microcontrollore con le caratteristiche soddisfacenti i requisiti del progetto deve essere in grado di comunicare attraverso una porta USB con l'applicativo per computer, avere la capacità di elaborare i messaggi che riceve e dei pin che funzionino sia da input che da output.

La scheda Arduino Uno corrisponde a tutti questi requisiti e, data la grande quantità di documentazione sul suo hardware e sulle sue librerie software, la scelta è ricaduta su di essa.



Figura 2.1: Arduino Uno [1]

La scheda, rappresentata nella figura 2.1, viene alimentata tramite un cavetto USB type B, il quale ha la doppia funzionalità di alimentatore e di consentire lo scambio di dati tramite comunicazione seriale con l'applicativo per computer.

Il microprocessore incorporato nella piattaforma permette l'elaborazione dei comandi inviati dall'applicativo e, una volta fatto questo, i 14 pin digitali presenti su di essa consentono di simulare la cloche addetta al movimento del tavolo radiogeno.

Tutte queste caratteristiche, unite alla semplicità di programmazione della board, rendono Arduino Uno perfetto per il progetto realizzato.

Di seguito andremo ad analizzare più nel dettaglio le caratteristiche della scheda, unite al sensore a sfera utilizzato per verificare il movimento del tavolo radiogeno.

2.1.1 Topologia e pinout della scheda

La scheda Arduino Uno include un microcontrollore AVR Atmega328, 6 pin analogici e 14 pin digitali programmabili come input oppure output.

In figura 2.2 si può osservare la disposizione dei vari componenti della scheda, elencati nella tabella 2.1.

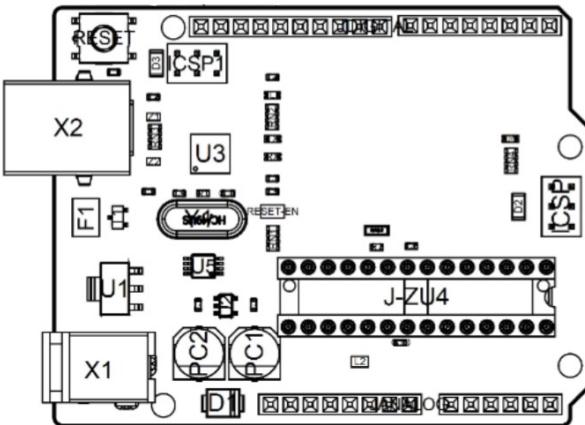


Figura 2.2: Arduino Uno: layout del top della piattaforma [2]

Rif.	Descrizione	Rif.	Descrizione
X1	Power jack 2.1x5.5mm	U1	SPX1117M3-L-5 Regulator
X2	USB B Connector	U3	ATMEGA16U2 Module
PC1	EEE-1EA470WP 25V SMD Capacitor	U5	LMV358LIST-A.9 IC
PC2	EEE-1EA470WP 25V SMD Capacitor	F1	Chip Capacitor, High Density
D1	CGRA4007-G Rectifier Diode	ICSP	Pin header connector
J-UZ4	ATMEGA328P Module	ICSP1	Pin header connector
Y1	ECS-160-20-4X-DU Oscillator		

Tabella 2.1: Componenti principali della scheda Arduino Uno

Il microcontrollore ATmega328 è il "cervello" della piattaforma: è qui che il firmware viene conservato ed eseguito, i dati vengono elaborati e da qui partono tutti i comandi alle altre componenti della scheda.

Altri componenti utilizzati nel progetto sono l'interfaccia USB type B, sia come fonte di alimentazione sia come via per la trasmissione dati dall'applicativo per computer al microcontrollore, e i 14 pin digitali.[2][3][4]

Di questi 14 pin digitali ne sono necessari tre per il funzionamento del progetto: il pin n° 3, 4 e 5. I primi due vengono impostati come uscite digitali per simulare la funzionalità

della cloche addetta al movimento del tavolo radiogeno, l'ultimo viene invece impostato come ingresso digitale per controllare il movimento del tavolo attraverso la lettura della corrente che passa all'interno di un sensore di tilt a sfera.

Nella figura 2.3 si possono vedere più nel dettaglio il diagramma di tutti i pin ed i circuiti integrati della scheda con le loro relative funzioni.

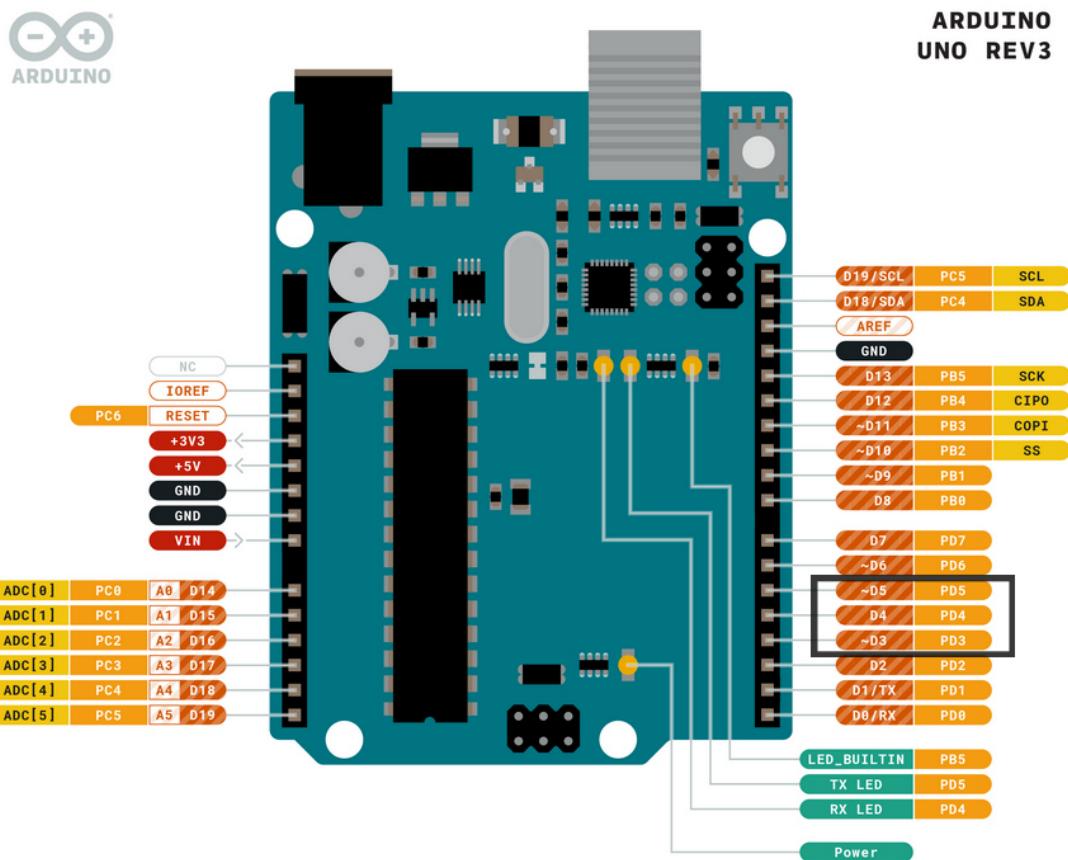


Figura 2.3: Pinout della scheda [5]

2.1.2 Connessione con la macchina a raggi X

Per sostituire la cloche addetta all'inclinazione del tavolo radiogeno, è necessario connettere Arduino Uno con la scheda madre della macchina, alla quale la cloche normalmente si collega.

Per fare questo si utilizzano due cavi elettrici, uno per pin, che vengono connessi tramite un adattatore allo slot dedicato alla cloche.

Grazie a questa connessione, quando i pin vengono settati allo stato HIGH, in essi passa

corrente elettrica di intensità pari a 20 mA e potenziale di 5 V. Il passaggio di questa corrente simula quello che normalmente avverrebbe con l'attivazione della cloche ed il macchinario inizia così il movimento.



Figura 2.4: Arduino Uno con modulo relè

In figura 2.4 è raffigurato Arduino Uno a cui è montato sopra un modulo relè, utilizzato per l'attivazione dei motori che consentono il movimento del tavolo radiogeno.

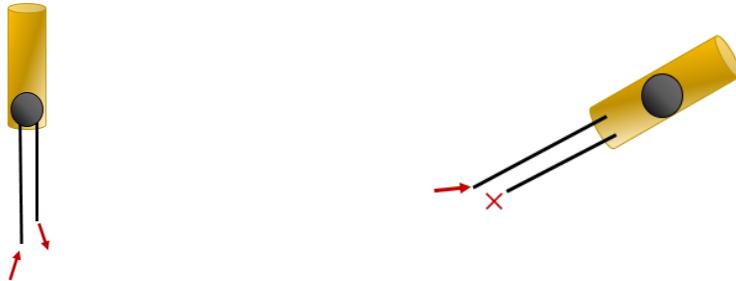
2.1.3 Sensore di inclinazione a sfera

Per verificare l'avvenimento effettivo dell'inclinazione del tavolo radiogeno è possibile utilizzare un sensore a sferetta. Esso viene fissato al di sopra del tavolo della macchina, parallelamente ad esso, come se fosse sdraiato su di esso, e connesso ad Arduino Uno.



Figura 2.5: Sensore a sfera [6]

Questo sensore è immaginabile come un tubo al cui interno si trova una pallina metallica libera di muoversi. Se il tubo si inclina in una direzione la pallina scorre all'interno di esso e chiude il circuito, come riportato nella figura 2.6. Inclinando il sensore nell'altra direzione invece il circuito viene aperto.



- (a) La corrente è in grado di scorrere quando il circuito è chiuso
(b) La corrente non è in grado di scorrere quando il circuito è aperto

Figura 2.6: Funzionamento del sensore di inclinazione a sfera [7]

Entrambi i fili metallici del sensore sono connessi con un cavo della lunghezza di qualche metro alla scheda Arduino Uno. In particolare, uno dei due è connesso con un pin a 5 V, mentre l'altro con il pin n° 5 impostato come ingresso digitale.

Durante un ciclo di movimento viene letto due volte il valore del pin n°5. Questa lettura avviene nella parte di ciclo in cui, se non sono avvenuti errori, il macchinario si trova all'inclinazione di +90° oppure di -90°. Essendo il sensore posizionato parallelamente al tavolo, in una delle due inclinazioni risulterà che il suo circuito è aperto mentre nell'altra questo sarà chiuso.

Se il circuito del sensore è chiuso, il pin si trova a 5 V e il microcontrollore legge il valore 1 digitale. Al contrario, se il circuito del sensore è aperto il pin si trova a 0 V essendo il circuito configurato con una resistenza pull-down. Viene dunque letto un valore pari allo 0 logico.

La lettura di questi valori consente di determinare l'avvenimento o meno del movimento del macchinario.

2.1.4 Problematiche del sensore di inclinazione a sfera

Il sensore di inclinazione a sfera è senz'altro il sistema che con il minor numero di risorse impiegate riesce a verificare l'avvenimento del movimento del macchinario. Questo sensore è però soggetto ad alcune problematiche, le quali rendono il suo utilizzo sub-ottimale rispetto ad altre soluzioni.

La problematica senza dubbio principale rilevata in seguito all'implementazione del sensore è l'attorcigliamento del cavo che lo connette con la scheda Arduino Uno. Questo cavo, lungo svariati metri, dopo un numero elevato di cicli inizia ad attorcigliarsi su se stesso. A lungo andare questo comporta l'usura e la cessazione di funzionamento del sensore.

Problematica secondaria del sensore è la sua non-sensibilità ai livelli di inclinazione: non si possono fare misure sull'angolo di inclinazione e può capitare che questo a volte non segni correttamente la chiusura del circuito se non si trova perfettamente perpendicolare al terreno.

Entrambe le problematiche hanno indirizzato il progetto verso la sostituzione del sensore con la scheda Arduino Nano 33 BLE Sense.

2.2 Arduino Nano 33 BLE Sense

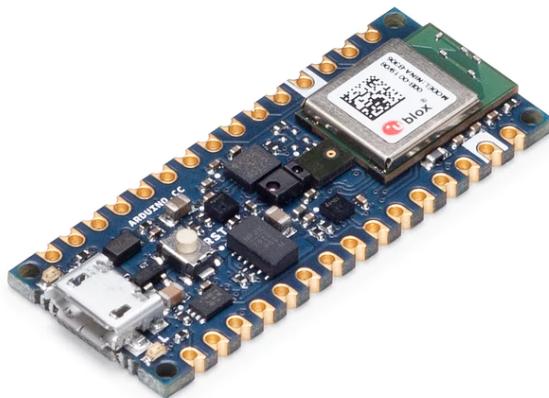


Figura 2.7: Arduino Nano 33 BLE Sense [8]

La scheda Arduino Nano 33 BLE Sense è stata scelta per sostituire il sensore di inclinazione a pallina. La piattaforma è infatti dotata di un'unità di misura inerziale contenente accelerometro e giroscopio, utilizzabili per la verifica del movimento del macchinario. Essa è inoltre dotata di capacità di comunicazione tramite Bluethooth® Low Energy ed è alimentabile tramite una batteria. Questa caratteristiche la rendono perfetta per la sostituzione del sensore di inclinazione a sfera.

Attraverso l’impiego di questa piattaforma si può infatti verificare il movimento del macchinario in modo wireless, senza bisogno di fili che possono risultare intralcianti per il macchinario. Attraverso la collocazione della board sul tavolo del macchinario, la verifica del movimento viene resa *wearable*.

2.2.1 Topologia e pinout della scheda

I componenti della board si collocano su di essa secondo la disposizione raffigurata nella figura 2.8.

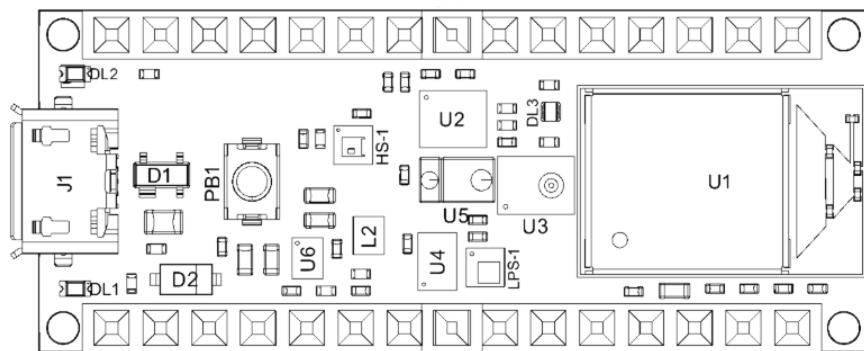


Figura 2.8: Arduino Nano: layout del top della piattaforma [9]

La tabella 2.2 mostra la disposizione e la descrizione delle varie componenti della piattaforma.

Rif	Descrizione	Rif	Descrizione
U1	NINA-B306 Module Bluetooth® Low Energy 5.0 Module	U6	MP2322GQH Step Down Converter
U2	LSM9DS1TR Sensor IMU	PB1	IT-1185AP1C-160G-GTR Push button
U3	MP34DT06JTR Mems Microphone	HS-1	HTS221 Humidity Sensor
U4	ATECC608A Crypto chip	DL1	Led L
U5	APDS-9660 Ambient Module	DL2	Led Power

Tabella 2.2: Componenti della scheda Arduino Nano 33 BLE Sense

Il microcontrollore presente sulla scheda è dotato di un modulo che consente la comunicazione tramite Bluetooth® Low Energy.

La board è provvista di una IMU (Inertial Measurement Unit) LSM9DS1 a 9 assi che può essere utilizzata per misurare l'accelerazione, la velocità angolare e i campi magnetici, ciascuno lungo i tre assi X, Y, Z.

Nel progetto viene valutato l'utilizzo di accelerometro e giroscopio per determinare il movimento del tavolo radiogeno.



ARDUINO
NANO 33 BLE SENSE

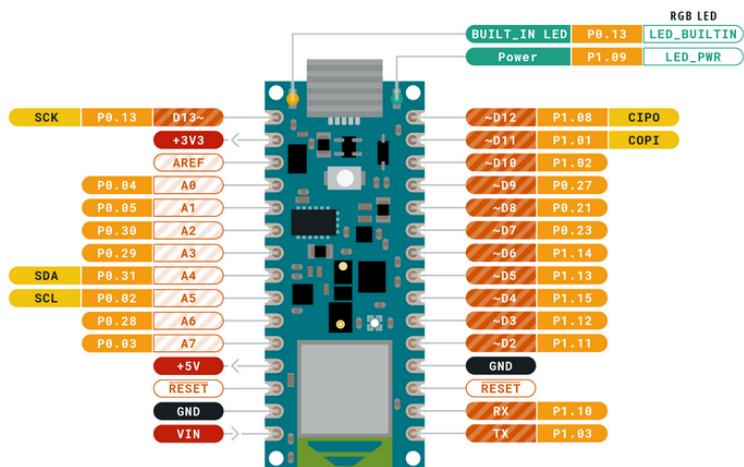


Figura 2.9: Pinout della scheda [10]

La porta micro-USB viene utilizzata per l'upload del firmware, ma potrebbe essere usata anche per l'alimentazione della scheda. In alternativa, per l'alimentazione, può essere utilizzato anche il pin denominato V_{IN} .[9][11]

2.2.2 Accelerometro

L'accelerometro fa parte del modulo IMU presente sulla scheda. L'accelerometro è un dispositivo in grado di misurare le accelerazioni lineari proprie, ossia la forza di gravità e le accelerazioni esterne. Esso è basato su un sistema massa-molla-smorzatore, la misura dell'accelerazione viene calcolata in base alla variazione capacitiva di condensatori al suo interno.

È possibile definire un fondo scala diverso a seconda dell'applicazione. I possibili valori sono i seguenti: $\pm 2/\pm 4/\pm 8/\pm 16$ g, dove con "g", si intende l'accelerazione di gravità, cioè 9.81 m/s^2 . Se si è interessati a dinamiche lente, è possibile quindi impostare il fondo scala più basso, che permetterà di avere una precisione maggiore.[12]

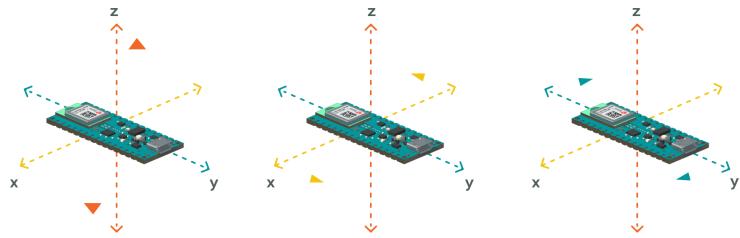


Figura 2.10: Gli assi dell'accelerometro [13]

Attraverso la lettura dei valori di accelerazione lungo gli assi X e Y è possibile risalire all'inclinazione a cui si trova il tavolo radiogeno.

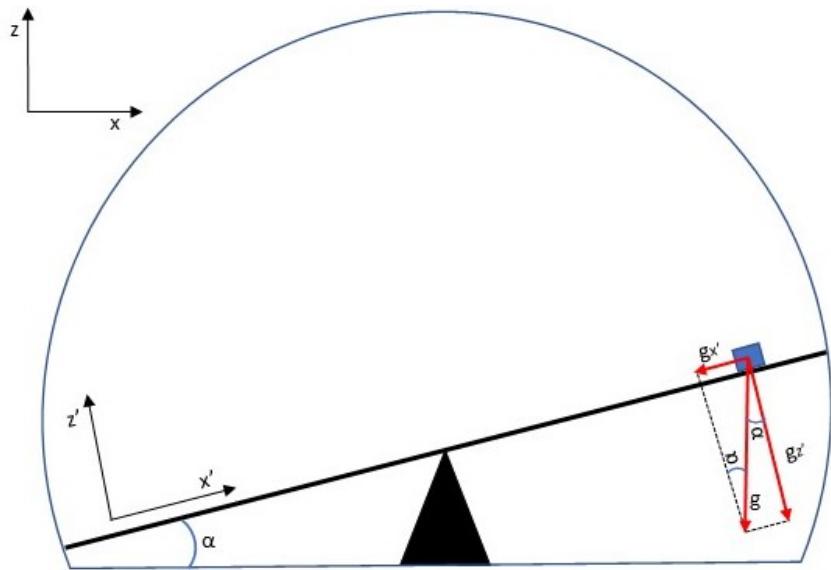


Figura 2.11: Schematizzazione dell'accelerazione

Dopo una iniziale accelerazione, il tavolo radiogeno si muove con velocità angolare costante. Le accelerazioni a cui il sensore è sottoposto durante il movimento sono due: l'accelerazione di gravità e l'accelerazione centripeta.

L'accelerazione centripeta, vista la bassa velocità angolare del movimento, ha valore basso rispetto a quella di gravità. È inoltre possibile sfruttare la sua dipendenza dal raggio di rotazione ($a_c = \omega^2 r$) per minimizzarla fino al renderla totalmente trascurabile. Questo

viene effettuato ponendo la scheda Arduino Nano 33 BLE Sense vicino al centro di rotazione del macchinario.

In questa configurazione è possibile affermare che tutta l'accelerazione vista dal sensore è dovuta all'accelerazione gravitazionale, la quale si distribuisce in maniera diversa nelle direzioni X' e Z' a seconda dell'inclinazione del tavolo, secondo lo schema riportato in figura 2.11.

Utilizzando le leggi trigonometriche, si può risalire all'angolo α di inclinazione del tavolo:

$$g_{x'} = g \sin \alpha$$

$$g_{z'} = g \cos \alpha$$

$$\tan \alpha = \frac{g_{x'}}{g_{z'}}$$

$$\alpha = \arctan \frac{g_{x'}}{g_{z'}}$$

2.2.3 Giroscopio

Come l'accelerometro, anche il giroscopio fa parte del modulo IMU presente sulla scheda. Il giroscopio è un dispositivo in grado di misurare la velocità angolare. Esso si basa sul movimento meccanico di una struttura MEMS (Micro Electro Mechanical System) dovuto alla forza di Coriolis.

Anche nel caso del giroscopio è possibile definire un fondo scala diverso a seconda dell'applicazione. I possibili valori son i seguenti: $\pm 245/\pm 500/\pm 2000$ dps (degrees per second).[12]

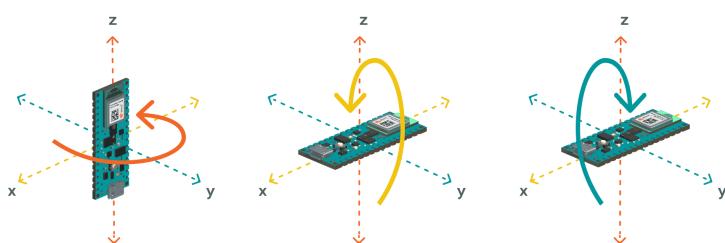


Figura 2.12: Gli assi del giroscopio [14]

È possibile risalire all'inclinazione del tavolo radiogeno integrando nel tempo il valore della velocità angolare letta dal giroscopio:

$$\alpha = \int_0^T \omega dt$$

2.2.4 Vantaggi dell'utilizzo della board

Come già accennato, è possibile alimentare la scheda tramite una batteria con voltaggio compreso fra i 4.5 V ed i 21 V sul pin V_{IN} evidenziato in figura 2.13.

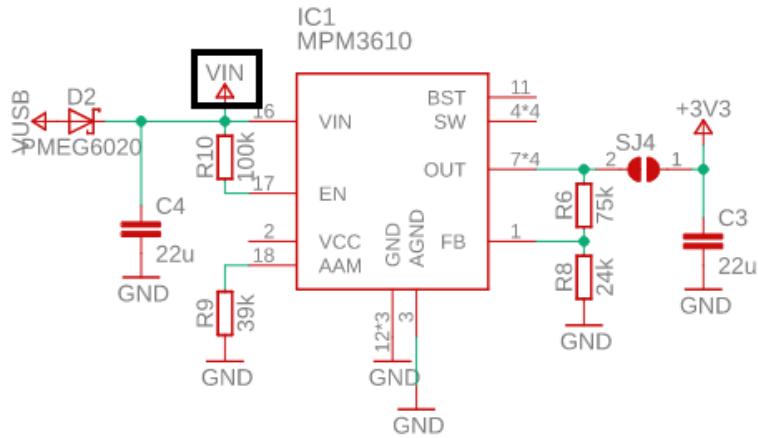


Figura 2.13: Schematica per l'alimentazione della board [15]

Posizionando la scheda sul tavolo radiogeno e collegandola all'applicativo, questa è in grado di fornire i dati di accelerazione e velocità angolare tramite connessione Bluetooth® Low Energy, permettendo quindi all'utente di controllare la posizione del tavolo stesso. Il sensore di controllo del movimento, con questo sistema, diventa quindi wireless. Questo comporta il beneficio di non dover avere una connessione fisica per la verifica del movimento del macchinario, superando così gli inconvenienti causati dall'usura del cavo che si manifestano dopo circa un migliaio di cicli, ben al di sotto dell'obiettivo di cicli desiderati.

Capitolo 3

Protocolli di comunicazione

In questo capitolo verranno presentati i due protocolli di comunicazione utilizzati nel progetto. Questi due protocolli sono la comunicazione seriale e la trasmissione di dati via Bluetooth® Low Energy. Entrambi vengono utilizzati per la trasmissione di dati fra le tre piattaforme utilizzate: l'applicativo per computer, Arduino Uno e Arduino Nano.

3.1 Comunicazione seriale

La trasmissione seriale è protocollo di comunicazione tra dispositivi digitali. La comunicazione avviene attraverso il trasferimento di bit lungo un canale di comunicazione. I bit sono trasferiti l'uno di seguito all'altro e raggiungono il ricevente in maniera sequenziale, nello stesso ordine di trasmissione.

La modalità di trasmissione seriale è molto diffusa in quanto, rispetto alla modalità di trasmissione parallela, richiede un minor numero di connessioni fisiche per la comunicazione (e comporta quindi un risparmio in termine di costi). Oltre a ciò, questa modalità è più tollerante rispetto alle interferenze e agli errori di trasmissione.[16]

3.1.1 Il protocollo UART

Il protocollo di comunicazione seriale utilizzato nel progetto è il protocollo UART. Esso viene utilizzato per consentire la trasmissione di dati dall'applicativo per computer alla

piattaforma Arduino Uno, tramite un cavetto USB type B.

UART è acronimo di Universal Asynchronous Receiver/Transmitter e definisce un protocollo per lo scambio di dati seriali tra due dispositivi. Il protocollo UART utilizza solamente due linee per connettere il trasmettitore ed il ricevitore. Una linea (TX) viene utilizzata per trasmettere i dati e una linea (RX) per riceverli. Entrambe le estremità hanno anche una connessione a massa.

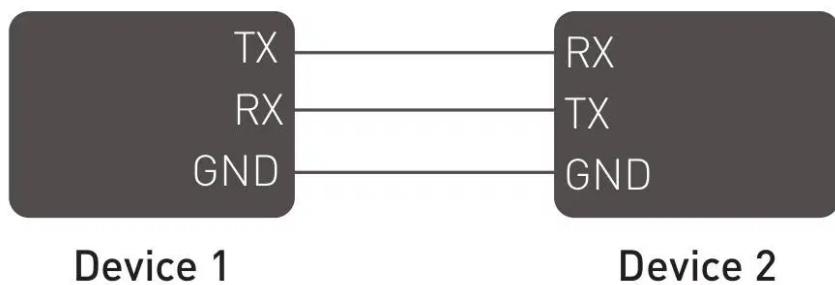


Figura 3.1: Schematizzazione della connessione UART [17]

La comunicazione UART può essere simplex (i dati sono inviati in una sola direzione), half-duplex (ogni lato parla ma solo uno alla volta) o full-duplex (entrambi i lati possono trasmettere contemporaneamente).

Questo protocollo è un protocollo detto asincrono, in quanto la comunicazione fra due dispositivi non dipende da nessun segnale clock. Siccome la velocità di comunicazione non è definita da nessun clock, il dispositivo che manda i dati non può essere certo che il ricevitore li abbia ottenuti tutti in modo corretto. Per ovviare a questo problema, entrambi i dispositivi devono trasmettere alla stessa velocità prestabilita (detta anche baud rate). I baud rate più comunemente utilizzati sono 4800, 9600, 19200.

I dispositivi che comunicano utilizzando il protocollo UART mandano dati in pacchetti di dimensioni fisse. Questi contengono inoltre delle informazioni aggiuntive riguardo l'inizio del messaggio, la fine del messaggio e la conferma di corretta ricezione.

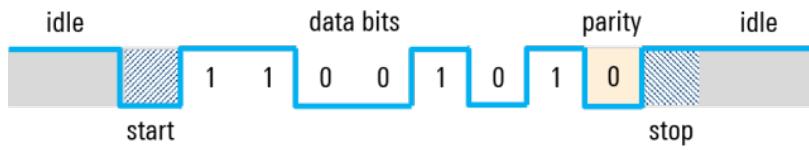


Figura 3.2: Pacchetto dati UART [18]

Per iniziare una trasmissione dati, il trasmettitore deve segnalare l'intenzione di invio di questi. Per fare questo viene utilizzato il bit di inizio. Questo bit è la transizione dallo stato high di inattività a quello low ed è subito seguito dai bit rappresentanti il messaggio che si vuole mandare.

Una volta terminata la trasmissione dei dati, viene utilizzato il bit di stop per far capire al ricevente che questa è conclusa. Il bit di stop è il ritorno allo stato high di inattività o la permanenza allo stato alto per un ulteriore tempo di bit.

Un pacchetto dati UART può anche contenere (opzionalmente) un bit detto bit di parità, utilizzato per rilevare eventuali errori avvenuti durante la trasmissione. Il bit in questione viene trasmesso alla fine dei bit dati e prima del bit di stop. Il valore di questo bit dipende dalla tipologia di parità che si è deciso di utilizzare: nel caso venga usata la parità pari, il bit viene impostato in modo che il numero totale di bit con valore 1 sia pari; nel caso di parità dispari invece viene impostato in modo che il numero di bit con valore 1 sia dispari. L'utilizzo di questi bit rende la trasmissione tramite protocollo UART più lenta rispetto ad una forma di comunicazione sincrona, questo è dovuto al fatto che solamente una parte dei bit inviati sono i dati che si vogliono mandare, il resto serve per la gestione della trasmissione.[19][20]

L'implementazione del protocollo UART sulla piattaforma Arduino consente all'utente di non doversi occupare della gestione della comunicazione a livello di bit. Esistono infatti librerie apposite di più alto livello che consentono la gestione di questa in maniera

automatica.

3.2 Bluetooth®

Il Bluetooth®, detto anche Bluetooth® Classic, è uno standard di trasmissione dati per reti wireless a basso raggio, ideato per permettere la comunicazione tra device come computer, cellulari ed altri dispositivi elettronici, senza aver bisogno dell'utilizzo di cavi.

Il protocollo di trasmissione prevede una tipologia di sincronizzazione sincrona ed una struttura master/slave.

Lo scambio di dati avviene utilizzando 79 canali, in una frequenza compresa fra i 2.402 GHz ed i 2.480 GHz, equispaziate l'una dall'altra.[21] [22]

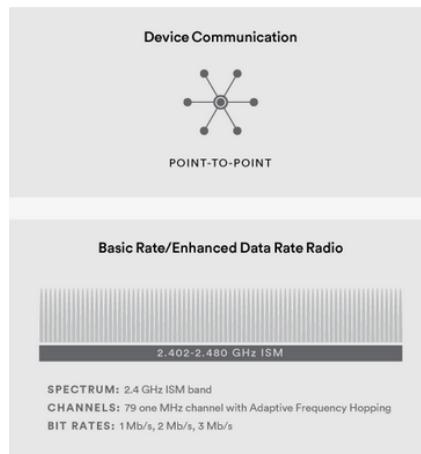


Figura 3.3: Panoramica del Bluetooth® Classic [23]

Attraverso questo standard di comunicazione, è possibile connettere fino ad un massimo di 8 devices in una rete creata ad hoc, chiamata *piconet*.[24]

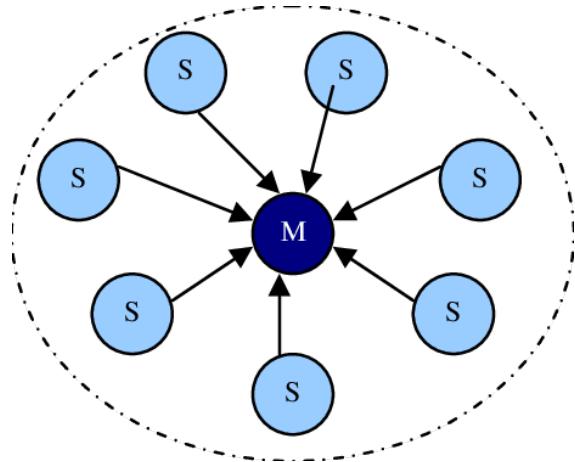


Figura 3.4: Schematizzazione di una rete piconet [25]

In questa rete soltanto uno di questi device ha la funzione di master, mentre gli altri hanno la funzione di slave. I ruoli possono però essere scambiati durante la connessione. I device così interconnessi sono sincronizzati con la *hopping sequence* e il clock del master all'interno della piconet.

3.2.1 Bluetooth® Low Energy

Il protocollo Bluetooth® Low Energy, noto anche con l'acronimo BLE, è lo standard di comunicazione utilizzato nel progetto. Esso è uno sviluppo dello standard Bluetooth® Classic descritto precedentemente.

Il BLE è stato ideato per risolvere il più grande problema del suo predecessore: l'elevato consumo di energia durante l'utilizzo.

Per ridurre al minimo il dispendio energetico, il dispositivo BLE viene tenuto in modalità riposo per la maggior parte del tempo. Quando un evento avviene, il device viene risvegliato e avviene la breve trasmissione di dati.

Il picco massimo di consumo di corrente è di 15 mA e il consumo medio di corrente invece si aggira attorno ad 1 μ A. Il consumo di potenza viene così ridotto a circa un decimo di quello del Bluetooth® Classic.

La trasmissione di dati avviene sempre ad una frequenza compresa tra i 2.402 GHz ed i 2.480 GHz ma i canali in cui questi vengono trasmessi sono adesso 40, non più 79.

Il raggio fino al quale in linea teorica si può avere una trasmissione dati affidabile è di

30 metri, ma un più tipico utilizzo si limita ad un raggio attorno ai 5 metri, in modo da limitare i possibili problemi dovuti alla perdita di segnale ed il consumo di batteria, il quale aumenta all'aumentare del raggio di trasmissione.

Il Bluetooth® Low Energy supporta ulteriori topologie di comunicazione: è possibile infatti avere, oltre alla classica trasmissione da punto a punto, anche la trasmissione *broadcast*, utile nel caso in cui si vogliano mandare dati a più di un dispositivo contemporaneamente e la trasmissione chiamata *mesh*, utilizzata per la creazione di un network di device di larga scala. [26][27][28]

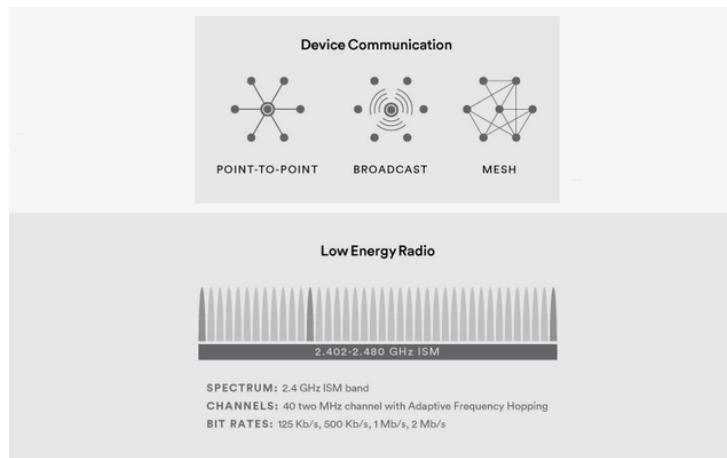


Figura 3.5: Panoramica del Bluetooth® Low Energy [29]

La topologia di trasmissione scelta, in quanto adatta ai requisiti del progetto, è quella "standard", ossia la trasmissione da punto a punto.

3.2.2 Connessione point-to-point

Questa tipologia di connessione, detta connessione punto a punto o anche *connected topology*, prevede due ruoli separati:

- Central (master): Scansiona le frequenze prestabilite e quando appropriato stabilisce la connessione. Una volta che questa è stabilita, controlla la temporizzazione e lo scambio periodico di dati.

- Peripheral (slave): Un device che si pubblica ed è in grado di accettare le connessioni. Quando la connessione è attiva, la periferica segue la temporizzazione del central e scambia dati con esso.

Per iniziare una connessione, il device centrale rileva dei *connectable advertising packets* inviati dalla periferica e manda ad essa una richiesta di stabilire la connessione esclusiva fra i due dispositivi.[26]

Una volta che la connessione è stabilita, i due dispositivi iniziano a scambiarsi dati in entrambe le direzioni.

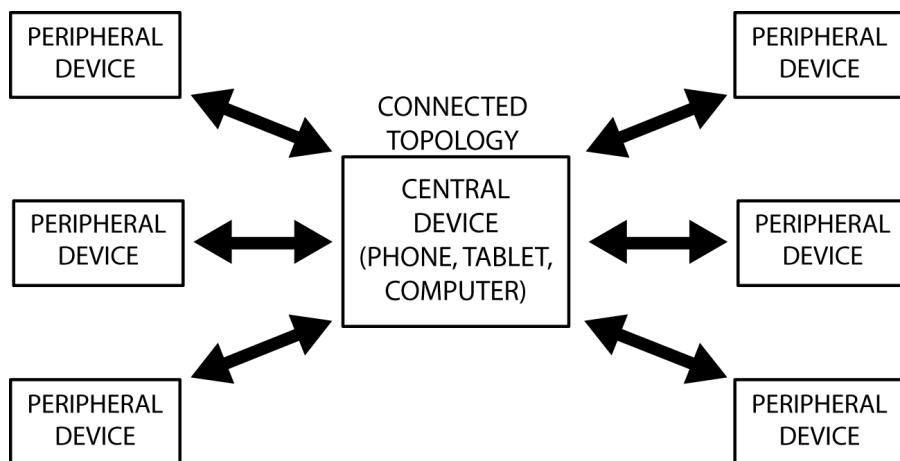


Figura 3.6: Schema della connessione point-to-point [30]

3.2.3 Profili generici del protocollo

I profili generici sono definiti all'interno delle specifiche della tecnologia Bluetooth® Low Energy. Essi sono fondamentali per garantire l'interoperabilità tra due o più dispositivi BLE.

Questi due profili sono il Generic Access profile (GAP) ed il Generic Attribute Profile (GATT).

- Generic Access Profile (GAP): specifica le metodologie di utilizzo a basso livello dei protocolli radio per riuscire a definire i ruoli, le procedure e le modalità che permettono ai vari dispositivi di trasmettere dati, trovare servizi, gestire le connessioni. Questo profilo è obbligatorio per tutti i dispositivi BLE.

- Generic Attribute Profile (GATT): si occupa dello scambio di dati, definisce un modello base di scambio dati e le procedure che permettono ai dispositivi di rilevare, leggere, scrivere e inviare dati da un dispositivo all’altro.

A differenza del profilo GAP, il profilo GATT non si occupa delle interazioni a basso livello tra i dispositivi ma solamente delle procedure di scambio dei dati e del loro format.[26][27]

3.2.4 Advertising e Scan Response Data

Esistono due modi per pubblicizzare con il profilo GAP: il carico utile "Advertising Data" ed il carico utile "Scan Response".

Entrambi i carichi utili sono identici e possono contenere fino a 31 bytes di dati, ma solamente il payload "advertising data" è obbligatorio, essendo l'unico payload dei due che viene costantemente trasmesso all'esterno dal device, in modo da pubblicizzare la propria esistenza ai device centrali in raggio utile.

Lo "scan response" payload è un'opzione secondaria che il device centrale può richiedere e permette ai disegnatori dei dispositivi di inviare più informazioni nell'advertising payload, come per esempio il nome del device in questione.[27]

Il processo di pubblicizzazione, ossia il processo di advertising funziona nel seguente modo:

- una periferica imposta un intervallo di pubblicizzazione e, ogni volta che questo intervallo temporale passa, ritrasmette il pacchetto "advertising data";
- se un device in ascolto è interessato ad ottenere anche lo "scan response" payload allora può richiederlo e il device periferica risponderà con i dati addizionali.

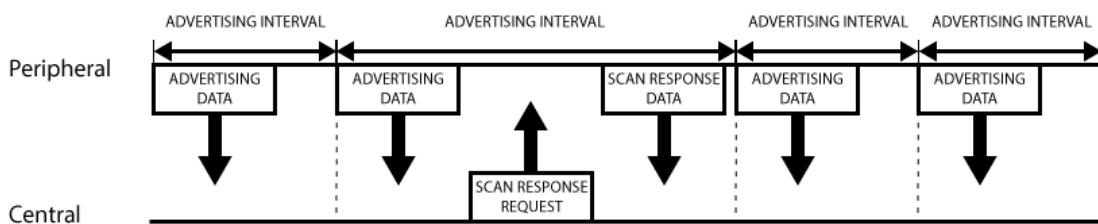


Figura 3.7: Rappresentazione del processo di advertising [31]

3.2.5 Transazioni GATT

Il profilo GATT, come già spiegato in precedenza, si occupa di definire come due dispositivi Bluetooth® Low Energy possono scambiarsi dati.

Questo profilo si basa concetti di services (servizi) e di characteristics (caratteristiche) per lo scambio di dati. Entrambi verranno spiegati più in dettagli nella successiva sezione.

Utilizzando un protocollo dati generico, chiamato Attribute Protocol (ATT), i servizi e le caratteristiche di un device BLE vengono conservati in una lookup table, ossia una struttura dati, usando un identificativo a 16 bit per ciascuno di questi.

Il GATT entra in gioco quando si è già formata una connessione fra due dispositivi, ossia quando si è già passati dal processo di pubblicizzazione regolato dal profilo GAP.

Le connessioni create sono esclusive, una periferica BLE può essere connessa solo ed esclusivamente ad una periferica centrale. Quando la connessione avviene, dunque, la periferica slave smette di pubblicizzarsi all'esterno finché questa connessione cessa di esistere.

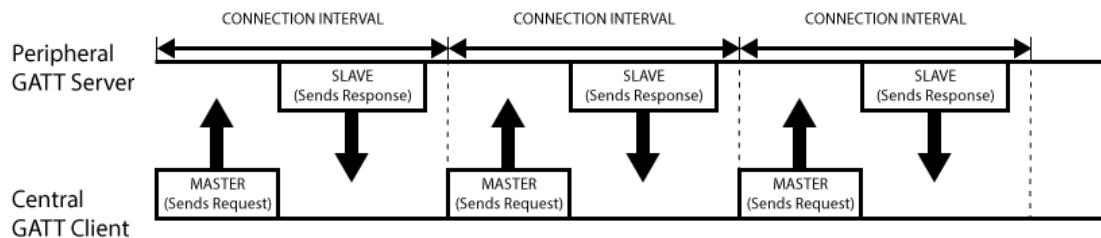


Figura 3.8: Rappresentazione di una transazione dati utilizzando il profilo GATT [32]

Il profilo GATT utilizza una relazione server/client.

La periferica slave è nota con il nome di "GATT server" e contiene la lookup table nominata in precedenza, assieme alla definizione dei suoi servizi e delle sue caratteristiche. Il dispositivo centrale invece viene chiamato "GATT client" ed è in grado di mandare richieste al GATT server.

Tutte le transazioni di dati sono iniziate dal device centrale, ossia dal GATT client, che riceve le risposte dalla periferica server.[27][33]

3.2.6 Servizi e caratteristiche

Le transazioni di dati del profilo GATT si basano sull'utilizzo di servizi e caratteristiche nominati in precedenza. Questi, sono degli oggetti di alto livello innestati l'uno nell'altro.

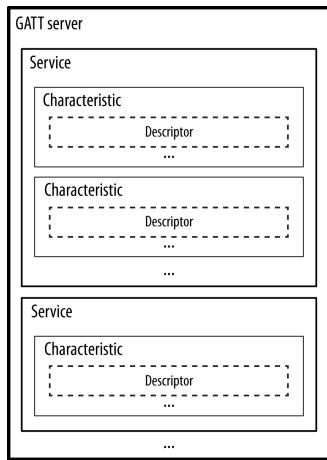


Figura 3.9: Rappresentazione di servizi e caratteristiche [34]

Ciascuna periferica BLE, ossia ciascun GATT server, può avere uno o più servizi. Ognuno di questi servizi può avere zero o più caratteristiche e, a loro volta, ciascuna caratteristica può avere zero o più descrittori.

Questa gerarchia è da rispettare in ogni dispositivo che deve essere compatibile con il profilo GATT, come i dispositivi BLE.

Profili

Un profilo è una lista predefinita di servizi, creata dal Bluetooth® Special Interest Group [35] (abbreviato SIG, è un network di organizzazioni che si occupano del mantenimento e dello sviluppo della tecnologia Bluetooth)®, oppure dal produttore del dispositivo BLE.[27]

Servizi

I servizi (services) sono delle unità logiche che contengono specifici dati chiamati caratteristiche (characteristics). Un servizio, come già detto può avere una o più caratteristiche e ciascuna di queste si distingue delle altre utilizzando un ID numerico unico chiamato

UUID. Questo UUID può essere di 16 bit se adottato ufficialmente dai servizi BLE, oppure di 128 bit nel caso di servizi personalizzati.[27]

Caratteristiche

Alla base del funzionamento del profilo GATT ci sono le characteristics. Queste forniscano un punto di accesso dati per le transazioni che si vogliono effettuare. Le caratteristiche sono come dei container per i dati utente.

Come per i servizi, anche le caratteristiche sono identificate attraverso un UUID da 16 oppure 128 bit, a seconda del fatto che queste siano definite dal Bluetooth® SIG oppure meno.

Molto importanti sono le proprietà delle caratteristiche, ossia l'insieme delle operazioni delle operazioni e delle procedure che possono essere usate con la caratteristica stessa.

Le tre più importanti proprietà delle caratteristiche sono riportate nella tabella 3.1.

Proprietà	Descrizione
Read	Se attivata, consente al client di leggere il valore della caratteristica
Write	Se attivata, consente al client di scrivere il valore della caratteristica
Notify	Se attivata, consente al server di notificare il client del cambiamento di valore della caratteristica

Tabella 3.1: Proprietà delle caratteristiche Bluetooth® Low Energy

Il client legge le proprietà della caratteristica per determinare quali operazioni è in grado di eseguire su di essa.[27][33]

Capitolo 4

Software

In questo capitolo verrà descritta il software sviluppato. Di questa fanno parte i firmware per le board Arduino Uno e Arduino Nano 33 BLE Sense e l'applicativo per computer creato utilizzando l'IDE Qt Creator.

Tutti e tre sono scritti utilizzando come linguaggio di programmazione C++.

Tutto il codice sviluppato, le cui parti più interessanti saranno presentate di seguito, è accessibile al link <https://github.com/giogan98/daffodil>.

4.1 Firmware Arduino Uno

Il firmware per Arduino Uno è stato creato e caricato sulla piattaforma utilizzando l'IDE (Integrated Development Environment) ufficiale fornito dalla compagnia che produce le board, ossia l'Arduino IDE.

Questo IDE permette di scrivere il codice e di caricarlo facilmente all'interno della board attraverso la comunicazione seriale.

4.1.1 Struttura del software

Come ogni programma scritto per una piattaforma Arduino, è necessario che siano presenti le funzioni `setup()` e `loop()`.

La funzione `setup()` viene eseguita subito dopo l'accensione oppure il reset della

scheda. È utilizzata per inizializzare le variabili, impostare le modalità (input/output) dei pin della scheda ed eseguire tutte le operazioni preliminari del caso. Questa funzione viene eseguita solamente una volta.[36]

Subito dopo aver eseguito questa funzione, la piattaforma inizia ad eseguire la funzione `loop()`. Come il nome suggerisce, questa funzione viene eseguita in continuazione, in un loop infinito.[37]

4.1.2 Convenzione della struttura dei comandi

Per poter trasmettere i comandi dall'applicativo per computer ad Arduino Uno è stato necessario stabilire una convenzione per la scrittura di questi comandi.

Il messaggio inviato deve per forza essere racchiuso fra parentesi quadre.

Subito dopo la prima parentesi quadra deve esserci uno dei due seguenti caratteri:

- il carattere ! utilizzato per impostare lo stato;
- il carattere ? utilizzato per domandare lo stato.

Per accedere ai pin digitali della scheda bisogna inoltre utilizzare l'acronimo D0 seguito dal numero del pin stesso.

Nel caso in cui il comando sia un comando di set dello stato, questo deve essere specificato nel seguente modo:

- si mette un punto subito dopo il numero del pin scelto;
- per settare lo stato HIGH del pin si scrive il numero 1;
- per settare lo stato LOW del pin si scrive il numero 0.

Un esempio di comando utilizzato è il seguente:

[!D03 . 1]

[!D04 . 0]

Il primo comando setta il pin digitale numero 3 allo stato HIGH, il secondo invece imposta il pin digitale numero 4 allo stato LOW.

4.1.3 Inizializzazione e setup della piattaforma

La funzione `setup()` eseguita all'avvio della board è la seguente:

```
void setup(void)
{
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, INPUT);
    Serial.begin(9600);
}
```

Grazie a questa funzione si impostano come output i pin 3 e 4 della board in modo da poterli comandare e poter settare in futuro il loro stato in LOW oppure HIGH. Il pin numero 5 viene invece settato come input. Questi tre pin serviranno per simulare la cloche del tavolo radiogeno: con il pin numero 3 si simula una direzione di movimento e con il pin 4 la direzione opposta di movimento. Il pin 5 invece serve per verificare il passaggio o meno di corrente all'interno del sensore a pallina.

Per far sì che la board riceva i dati inviati dall'applicativo per computer è inoltre necessario inizializzare la connessione seriale impostando i bit per secondo (ossia il baud rate) della trasmissione. In questo caso il baud rate impostato è di 9600 bit per secondo. [38]

4.1.4 Funzione loop

La funzione `loop()` è la seguente:

```
void loop(void)
{
    crunchSerial();
    checkActivity();
    delay(50);
}
```

La prima funzione chiamata al suo interno, la funzione `crunchSerial()` si occupa di leggere i comandi in ingresso alla board, trasmessi dall'applicativo, di elaborarli e di svolgere le istruzioni contenute in essi.

La seconda funzione invece è un meccanismo di sicurezza e verrà spiegata nelle successive sezioni.

4.1.5 Elaborazione dei comandi

L'elaborazione dei comandi ricevuti avviene ad opera della funzione `crunchSerial()`. Essa si appoggia su altre funzioni per l'elaborazione dei comandi.

La funzione `extractMsg()` ha il compito di leggere la porta seriale e di trovare eventuali nuovi messaggi in ingresso.

Successivamente, dopo un controllo sulla lunghezza di questo messaggio, viene chiamata la funzione `getValidMessage()`, la quale ha il compito di controllare la presenza delle parentesi quadre di inizio e fine messaggio e rimuoverle.

Una volta fatto questo, vengono trovati il carattere di comando, la periferica ed il pin:

- la funzione `getCmd()` ha il compito di trovare il carattere di comando del messaggio, che esso sia un ! oppure un ?;
- la funzione `getPeriph()` serve per trovare la periferica, ovvero i pin dichiarati come uscita digitale. In questo caso il messaggio dopo il carattere di comando contiene la sigla DO, acronimo di Digital Output;
- viene identificato il pin nominato nel messaggio di comando, attraverso la funzione `getSubDevice()`.

Una volta fatto tutto ciò, vengono chiamate le funzioni `processSetDigitalPin()` e `processReqDigitalPin()`, in base al tipo di carattere di comando presente nel messaggio.

In queste due funzioni verrà settato oppure richiesto lo stato (HIGH/LOW) del pin.

4.1.6 Stop di emergenza del macchinario

La funzione `checkActivity()` è stata pensata per interrompere il movimento del tavolo radiogeno nel caso in cui l'applicativo per computer si blocchi ma la board Arduino Uno rimanga alimentata.

Per fare questo, si utilizza un contatore `iCounter` che viene incrementato ogni volta che la board esegue la funzione `loop()`.

```

void checkActivity(void)
{
    if (iCounter > 400)
    {
        for (int ii = 1; ii <= 13; ii++)
        {
            digitalWrite(ii,LOW);
        }
    }
    iCounter++;
}

```

Se la board non riceve quindi nessun messaggio per 400 cicli della funzione loop (ossia per circa 20 secondi), tutti i pin venogono settati allo stato LOW, simulando così una cloche ferma.

Se invece viene letto un messaggio prima del raggiungimento dei 400 cilci, il contatore viene resettato.

4.2 Firmware Arduino Nano

Il programma utilizzato per scrivere il firmware per questa piattaforma è lo stesso utilizzato anche per la board Arduino Nano, ossia l'Arduino IDE.

4.2.1 Librerie utilizzate

Per il corretto funzionamento del software è necessario importare due librerie:

```

#include <ArduinoBLE.h>
#include <Arduino_LSM9DS1.h>

```

La prima libreria consente di utilizzare le funzionalità Bluetooth® Low Energy di cui la piattaforma è capace. In questo modo è possibile trasformare la board in una periferica BLE, un GATT server, con lo scopo di fornire i dati di giroscopio ed accelerometro al GATT central che sarà costituito dall'applicativo per computer.[39]

La seconda libreria consente invece di accedere ai dati di giroscopio ed accelerometro sfruttando il modulo LSM9DS1 di cui la board è dotata.[40]

4.2.2 Design dei servizi e delle caratteristiche

Per utilizzare il dispositivo come un GATT server ed inviare dati tramite BLE è necessario definire i servizi e le caratteristiche Bluetooth® Low Energy che saranno utilizzate.

Il design scelto è quello di utilizzare un singolo servizio, chiamato `dataService`.

Questo servizio contiene sei diverse caratteristiche:

- tre caratteristiche per i dati dell'accelerometro;
- tre caratteristiche per i dati del giroscopio.

In questo modo è possibile avere una caratteristica per asse per sensore utilizzato.

Questo permette di inviare (e leggere dal GATT central) i valori singoli di accelerometro e giroscopio divisi per asse X, Y e Z.

La tipologia scelta per le caratteristiche è quella "float" in quanto i valori letti dai sensori sono numeri float.

4.2.3 Dichiarazione dei servizi e delle caratteristiche

Per poter dichiarare il servizio `dataService` e le caratteristiche è necessario definire un UUID per ognuno di questi. Non essendo servizi caratteristiche definite dal Bluetooth® SIG, sono necessari degli UUID da 128 bit.

Gli UUID vengono definiti utilizzando la direttiva `define` in C++[41]. Il codice qui riportato si riferisce ad uno solo degli assi, la procedura è simile anche per i rimanenti due assi.

```
#define BLE_UUID_DATA_SERVICE
↪ "5aaeb650-c2cb-44d1-b4ab-7144e08aed2e"

#define BLE_UUID_ACCELEROMETER_CHARACTERISTIC_X
↪ "f4055745-6f5a-4e2b-8433-2704337cc3b5"

#define BLE_UUID_GYROSCOPE_CHARACTERISTIC_X
↪ "9936153d-65bc-4479-b079-aa25569f9ab1"
```

Una volta fatto ciò, è possibile dichiarare singolarmente il servizio `dataService` e ognuna delle sei caratteristiche, una per sensore per asse.

```

BLEService dataService(BLE_UUID_DATA_SERVICE);

BLEFloatCharacteristic accelerometerCharacteristi-
→ cX(BLE_UUID_ACCELEROMETER_CHARACTERISTIC_X, BLERead |
→ BLENotify);

BLEFloatCharacteristic
→ gyroscopeCharacteristicX(BLE_UUID_GYROSCOPE_CHARACTERISTIC_X,
→ BLERead | BLENotify);

```

Nella dichiarazione delle caratteristiche è importante l'attivazione delle proprietà BLERead e BLENotify. Come già spiegato nel capitolo riguardante i protocolli di comunicazione, queste permettono al central di leggere il valore della caratteristica e al server di notificare il client del cambiamento di valore della caratteristica rispettivamente.

4.2.4 Inizializzazione e setup della piattaforma

La funzione **setup()** eseguita all'avvio della board è la seguente:

```

void setup(void)
{
    initializeBLE();
    initializeIMU();
    setupBLE();
}

```

L'inizializzazione del modulo IMU è necessaria per ottenere i dati letti da accelerometro e giroscopio. Quella del modulo BLE invece è necessaria per poter utilizzare il Bluetooth® Low Energy.

Molto importante è la funzione **setupBLE()**:

```

void setupBLE(void)
{
    BLE.setLocalName("SenseBLE");

    BLE.setAdvertisedService(dataService);

    dataService.addCharacteristic(accelerometerCharacteristicX);
}

```

```

    dataService.addCharacteristic(accelerometerCharacteristicY);
    dataService.addCharacteristic(accelerometerCharacteristicZ);

    dataService.addCharacteristic(gyroscopeCharacteristicX);
    dataService.addCharacteristic(gyroscopeCharacteristicY);
    dataService.addCharacteristic(gyroscopeCharacteristicZ);

    BLE.addService(dataService);

    BLE.advertise();
}

```

Grazie a questa funzione, è possibile impostare il nome con il quale il dispositivo andrà a pubblicizzarsi all'esterno. In questo caso il nome scelto è "SenseBLE".

Viene definito come servizio da pubblicizzare il `dataService` ed ad esso vengono aggiunte le sei caratteristiche precedentemente descritte.

Infine, viene iniziata la pubblicizzazione del dispositivo. Questo sarà ora rilevabile da altri dispositivi esterni.

4.2.5 Funzione loop

Dopo aver eseguito la funzione `setup()`, la board passa all'esecuzione della funzione `loop()`.

```

void loop(void)
{
    BLEDevice central = BLE.central();

    if (central)
    {
        while (central.connected())
        {
            sendAccelerometerValues();
            sendGyroscopeValues();
            delay(500);
        }
    }
}

```

```
    }  
}
```

In questa funzione viene verificata la connessione ad un device centrale, ossia all'applicativo che deve connettersi alla piattaforma BLE per ricevere dati di giroscopio ed accelerometro.

Se la connessione ad un dispositivo centrale risulta essere attiva, vengono eseguite le funzioni `sendAccelerometerValues()` e `sendGyroscopeValues()`. Viene poi effettuato un delay di 500ms, in quanto il ciclo completo ha un tempo di esecuzione compreso fra i 25 secondi ed i 30 secondi ed aumentare ulteriormente la frequenza di misura dei dati non risulterebbe in ulteriori benefici.

4.2.6 Lettura ed invio dati

La lettura dei dati letti da accelerometro e giroscopio ed il loro invio avviene all'interno delle funzioni `sendAccelerometerValues()` e `sendGyroscopeValues()`. Queste due sono pressoché identiche: l'unica differenza fra le due è che la prima si occupa di interpellare l'accelerometro mentre la seconda il giroscopio.

```
void sendAccelerometerValues(void)  
{  
    float x, y, z;  
  
    if (IMU.accelerationAvailable())  
    {  
        IMU.readAcceleration(x, y, z);  
  
        accelerometerCharacteristicX.writeValue(x);  
        accelerometerCharacteristicY.writeValue(y);  
        accelerometerCharacteristicZ.writeValue(z);  
    }  
}  
  
void sendGyroscopeValues(void)  
{  
    float x, y, z;
```

```

if (IMU.gyroscopeAvailable())
{
    IMU.readGyroscope(x, y, z);

    gyroscopeCharacteristicX.writeValue(x);
    gyroscopeCharacteristicY.writeValue(y);
    gyroscopeCharacteristicZ.writeValue(z);
}
}

```

L'invio dei dati viene fatto attraverso la funzione `writeValue()`. Il valore float viene così scritto come valore della caratteristica e, attraverso la proprietà `BLENotify` il client viene notificato del cambiamento di questo valore.

4.3 Applicativo Qt

Di seguito verrà spiegato il funzionamento del programma sviluppato con QT creator. Il programma è una interfaccia grafica che consente la gestione delle connessioni ad Arduino Uno e Arduino Nano, oltre a consentire l'invio e la ricezione di dati da questi. Verranno descritte le principali funzionalità del programma, la sua struttura e le librerie utilizzate.

4.3.1 Interfaccia del programma

L'interfaccia del programma è pensata in modo che in essa siano subito visibili gli elementi essenziali: l'opzione di avviare oppure interrompere il movimento del macchinario ed il numero di cicli effettuati.



Figura 4.1: Pagina iniziale del programma

Cliccando sul bottone in alto a sinistra si accede alla barra laterale dei menù:

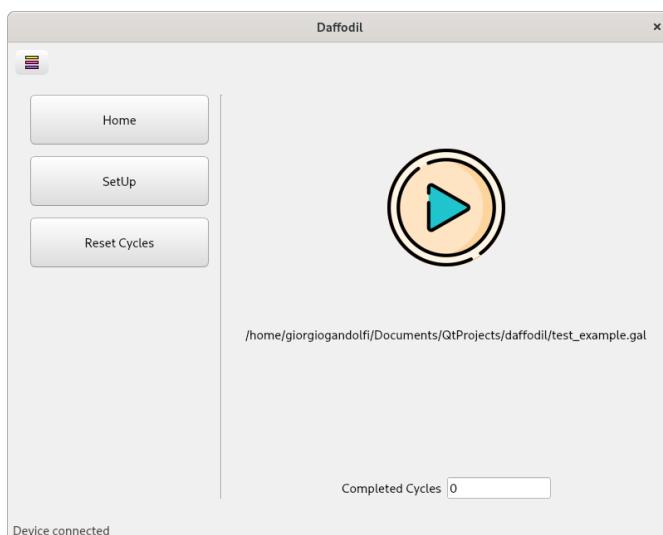


Figura 4.2: Pagina iniziale con barra laterale dei menù

Grazie a questa barra laterale è possibile accedere alla pagina di setup oppure resettare il numero di cicli effettuati.

Nella pagina di setup vengono gestite tutte le impostazioni riguardanti la connessione seriale, l'utilizzo di sensori per la verifica del movimento del macchinario, la scelta del file contenenti i comandi di movimento da fare eseguire alla macchina e, se viene selezionato come sensore Arduino Nano 33 BLE Sense, è possibile gestire anche la connessione BLE a questo.

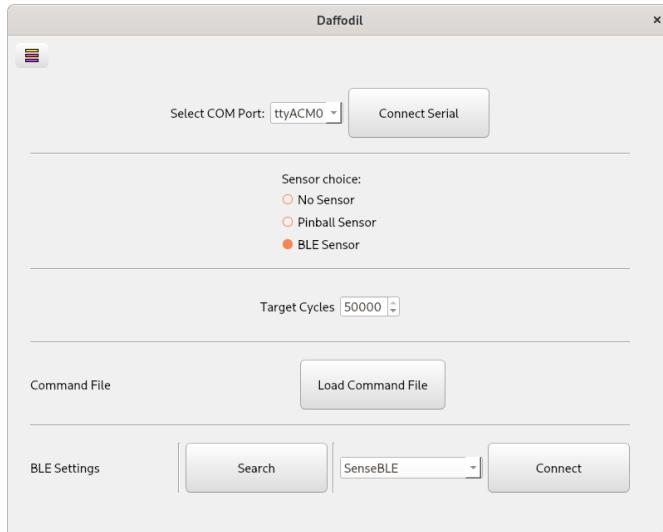


Figura 4.3: Pagina di setup

Una volta configurato il programma, si può iniziare il movimento del macchinario premendo il tasto "Play". Per fermarne l'esecuzione è sufficiente premere il tasto stop.

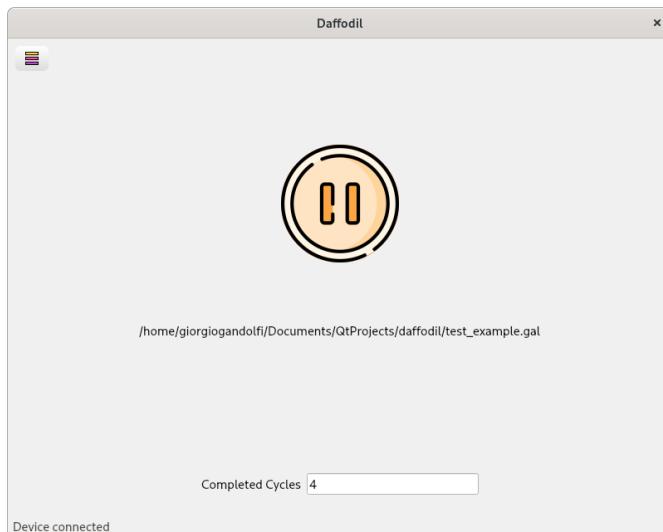


Figura 4.4: Pagina iniziale con pulsante di stop

4.3.2 Struttura del programma

Il programma è suddiviso in diverse classi, ciascuna delle quali avente un determinato scopo ed argomento.

Per la gestione a basso livello degli elementi del programma abbiamo le seguenti classi:

- **autologger**: si occupa del log automatico di alcune operazioni svolte dal programma, salvando i dati necessari in un file di testo;
- **ISettings**: si occupa del salvataggio delle impostazioni selezionate dall’utente, come ad esempio il baud rate della connessione seriale e la porta COM selezionata;
- **commandfileprocessor**: elabora il file contenenti i comandi da inviare ad Arduino Uno per poter far muovere il tavolo radiogeno;
- **serialcontroller**: gestisce la connessione seriale con Arduino Uno;
- **blecontroller**: gestisce la connessione Bluetooth® con Arduino Nano 33 BLE Sense.

Il livello più alto del programma invece è gestito da due ulteriori classi:

- **ISupervisor**: coordina i meccanismi di lettura e scrittura dati con Arduino Uno, la verifica del movimento del tavolo radiogeno;
- **mainwindow**: crea l’interfaccia del programma che l’utente vede e con la quale interagisce.

4.3.3 Macchina a stati finiti

La classe che si occupa di gestire la trasmissione e la ricezione dei dati delle piattaforme Arduino Uno ed Arduino Nano, chiamata **ISupervisor**, si basa su un automa a stati finiti.

Una macchina a stati finiti, detta anche automa a stati finiti è un sistema che possiede le seguenti caratteristiche:

- dinamicità, ossia la caratteristica di evolvere nel tempo passando da uno stato all’altro;
- discretezza, ossia che le variabili in ingresso e gli stati del sistema possono essere espressi tramite dei valori discreti;
- l’insieme dei possibili input, output e degli stati ha un numero finito di elementi.

Per indicare il funzionamento degli automi si usano degli schemi chiamati grafi. Il grafo della macchina a stati impiegata nel programma è rappresentato nella figura 4.5.

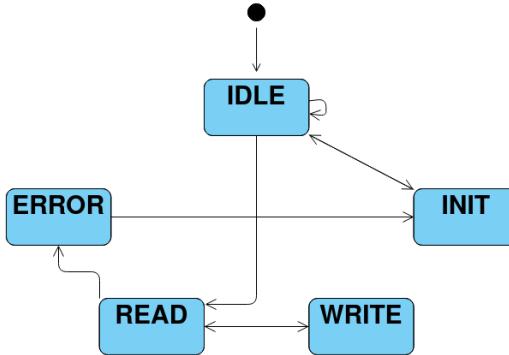


Figura 4.5: Grafo della macchina a stati finiti

I grafi sono costituiti da nodi, rappresentanti i vari stati possibili, collegati da archi orientati, i quali indicano il passaggio da uno stato all'altro. Il passaggio da uno stato all'altro avviene al verificarsi di determinate condizioni.[42][43][44][45]

La struttura di macchina a stati finiti è stata implementata utilizzando una struttura del tipo **switch-case**, lavorando su una variabile che rappresenta lo stato della macchina.

Gli stati dell'automa implementato sono cinque: *idle*, *init*, *read*, *write* ed *error*.

Il passaggio da uno stato all'altro avviene attraverso apposite funzioni. L'utente ha la possibilità di far eseguire il passaggio dallo stato di *idle* a quello di *read* premendo il pulsante "Play". Premendo invece il pulsante "Stop", lo stato passa a quello di *idle*.

Stato **idle**

Lo stato di *idle*, ovvero lo stato di inattività, è lo stato iniziale in cui il programma si trova. Non viene svolta alcuna operazione in questo stato, in esso si attende che l'utente decida di svolgere qualche azione.

Stato init

Lo stato di *init* è lo stato di inizializzazione. In esso vengono caricati il numero di cicli precedentemente effettuati e viene aperta la porta seriale, necessaria per comunicare con Arduino Uno.

Una volta fatto ciò, si ritorna allo stato *idle*, in attesa di altre azioni utente.

Stato read

Nello stato di *read* viene controllato lo stato della connessione seriale. Se non vengono rilevati problemi con questa, si imposta come stato successivo quello di *write*, altrimenti si passa allo stato *error*.

Stato write

All'interno dello stato di *write*, vengono eseguite le operazioni per la gestione di un nuovo ciclo, se ancora non si è raggiunto il numero di cicli target.

Se così non è, si eseguono le operazioni di scrittura sulla porta seriale, in modo da inviare alla piattaforma Arduino Uno i comandi da eseguire.

Successivamente, una volta fatto tutto questo, si imposta il successivo stato in quello di *read*.

Stato error

Se il programma entra nello stato di *error*, vengono settati tutti i pin di Arduino Uno allo stato LOW, in modo che il tavolo radiogeno smetta di muoversi. Viene poi chiusa la connessione seriale.

4.3.4 Struttura del file contenente i comandi di movimento

I comandi di movimento da fare eseguire al tavolo radiogeno sono conservati all'interno di un file. L'utente è in grado di selezionare questo file attraverso un pulsante nella schermata di setup del programma.

Il file in questione è un documento di testo contenente diverse righe di comandi da eseguire. Ciascuna di queste righe contiene due elementi: il primo elemento è il tempo che deve

passare (espresso in millisecondi) per l'esecuzione del comando; il secondo elemento è il comando stesso. I due campi sono separati dal carattere ";".

```
1000 ; [!D03.1]
1000 ; [!D04.1]
1000 ; [?D05]
1000 ; [!D03.0]
1000 ; [!D04.0]
1000 ; [?D05]
```

Questo è un esempio dei comandi contenuti all'interno di un file. Ciascun comando viene inviato tramite comunicazione seriale ad Arduino Uno dopo un tempo di 1000 ms. Non appena il microcontrollore riceve il comando, esso viene convalidato ed eseguito. I primi due comandi impostano i pin 3 e 4 allo stato HIGH rispettivamente. Il terzo comando invece domanda lo stato del pin n° 5. I successivi due comandi impostano allo stato LOW i pin 3 e 4. L'ultimo comando è uguale al terzo.

4.3.5 Gestione della connessione seriale

La classe `serialcontroller` gestisce la connessione seriale con la piattaforma Arduino Uno. Essa si appoggia alla libreria `QSerialPort`[46], grazie alla quale è possibile aprire una connessione seriale con capacità di lettura e di scrittura.

La funzione `openSerialPort()` è utilizzata per iniziare una nuova connessione seriale: questa carica una serie di dati scelti in precedenza dall'utente (fra i quali il nome della porta seriale ed il baud rate), li imposta come parametri della trasmissione seriale e inizia così una connessione con il dispositivo scelto.

Per la lettura dei dati viene utilizzata la funzione `read()`, mentre per la scrittura viene utilizzata la funzione `write()`. La funzione `write()` esegue un check se la struttura dei dati che si stanno per inviare rispetta quella scelta per convenzione e, se questo controllo viene passato, i dati vengono mandati al dispositivo connesso.

All'interno di questa classe è anche contenuta la funzione `emergencyShutdown()`, la quale se chiamata imposta allo stato LOW tutti i pin della scheda Arduino Uno, per evitare che il tavolo si muova.

4.3.6 Gestione della connessione Bluetooth®

La gestione della connessione Bluetooth® Low Energy dell'applicativo ad Arduino Nano 33 BLE Sense è gestita dalla classe `blecontroller`, la quale sfrutta il Qt Bluetooth Low Energy API [47].

Questa classe si occupa sia di stabilire una connessione con Arduino Nano 33 BLE Sense, sia della lettura dei dati dei sensori da esso inviati.

Connessione ad Arduino Nano 33 BLE Sense

Il programma consente di effettuare una scansione di tutti i dispositivi BLE nelle vicinanze utilizzando la funzione `startDeviceScan()`, in modo da poter decidere a quale dispositivo connettersi.

L'utente ha la capacità di selezionare il dispositivo con cui effettuare la connessione ed il programma si connette ad esso grazie alla funzione `connectToSelectedDevice()`.

Il programma aspetta poi che vengano rilevati i servizi resi disponibili dal dispositivo a cui si è connesso. Questi vengono filtrati per UUID, in ricerca del servizio `dataService`. Se questo viene trovato, si cercano a loro volta le caratteristiche (sempre filtrate per UUID) di accelerometro e giroscopio.

Una volta trovate, il programma inizia a ricevere i dati letti dai due sensori.

Lettura dei dati

La lettura dei dati avviene quando questi vengono resi disponibili dalla scheda Arduino Nano 33 BLE Sense. Questo meccanismo si basa sulla proprietà *Notify* del Bluetooth® Low Energy, della quale le caratteristiche di accelerometro e giroscopio sono dotate.

Quando i dati sono disponibili, il programma viene notificato della loro disponibilità e questi vengono letti sfruttando la funzione `updateSensorsData()`.

4.3.7 Raccolta dati

Per decidere il metodo da utilizzare per la verifica del movimento utilizzando Arduino Nano 33 BLE Sense, dotato di accelerometro e giroscopio, sono prima stati raccolti i dati dei sensori durante un ciclo di movimento. I dati raccolti sono stati raffigurati utilizzando Python e la libreria `matplotlib` e sono di seguito riportati.

Raccolta dati di accelerometro

La distribuzione dell'accelerazione di gravità g durante il movimento del macchinario da una posizione iniziale a -90° ad una finale di $+90^\circ$ è riportata nella figura 4.6.

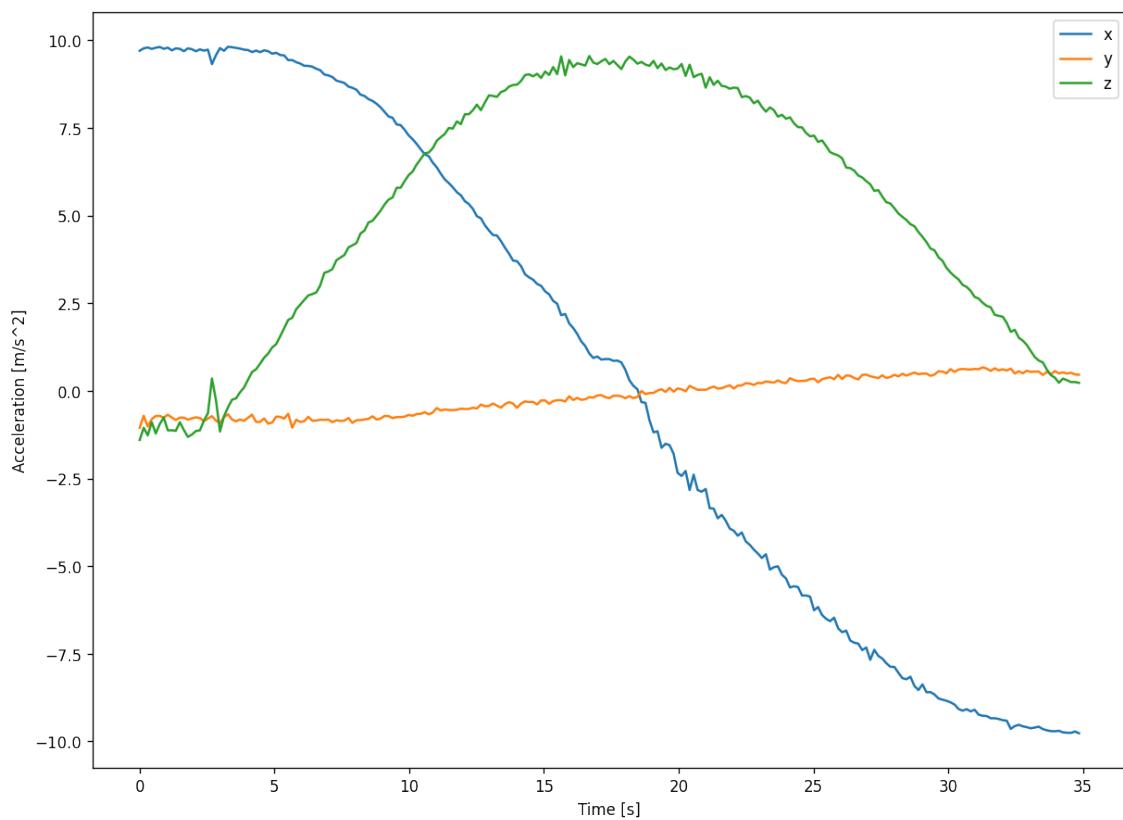


Figura 4.6: Andamento nei tre assi dell'accelerazione di gravità lungo un ciclo

Analizzando il grafico si possono effettuare alcune osservazioni:

1. Nella posizione iniziale di -90° l'accelerazione si distribuisce tutta lungo l'asse X, è nulla sull'asse Z.

2. La precedente osservazione vale anche nel punto finale, quando il macchinario si trova a $+90^\circ$.
3. Quando il macchinario si trova invece parallelo al suolo, l'accelerazione si distribuisce solamente sull'asse Z, è nulla negli altri due.
4. Il valore di accelerazione lungo l'asse Y è sempre nullo.

Tutte queste osservazioni sono coerenti con quello che ci aspettavamo a livello teorico. Si può quindi utilizzare la distribuzione dell'accelerazione di gravità lungo gli assi X e Z per calcolare l'angolo α di inclinazione del macchinario, effettuando l'arcotangente del rapporto fra il valore lungo la Z e quello lungo la X. I risultati ottenuti sono rappresentati nella figura 4.7.

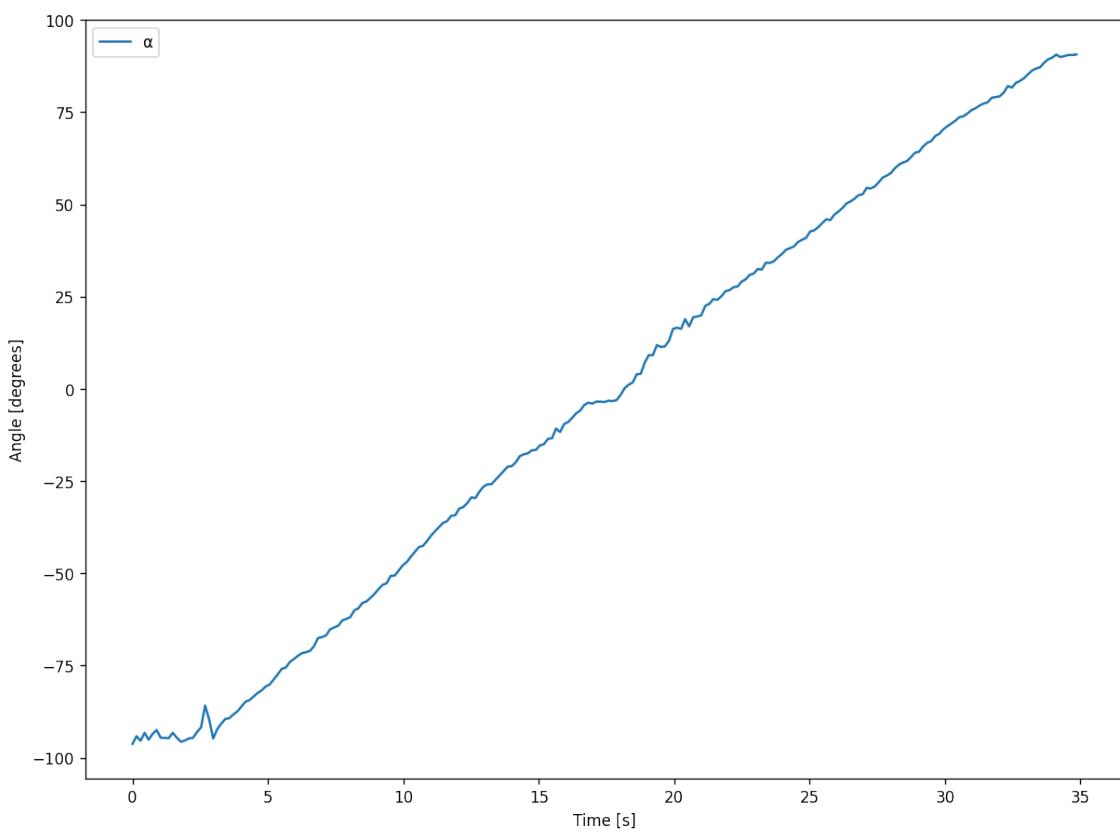


Figura 4.7: Progressione dell'angolo α nel tempo calcolata con l'accelerometro

Confrontando i risultati ottenuti utilizzando l'accelerometro con i dati reali di inclinazione, otteniamo nello scenario peggiore un errore di circa 5° , corrispondente ad un

errore di meno del 6%. Possiamo dunque ritenere che l'utilizzo dell'accelerometro porta a ottimi risultati nella stima dell'angolazione del tavolo radiogeno.

Raccolta dati di giroscopio

I dati raccolti dal giroscopio durante l'esecuzione di un ciclo di movimento sono riportati nella figura 4.8.

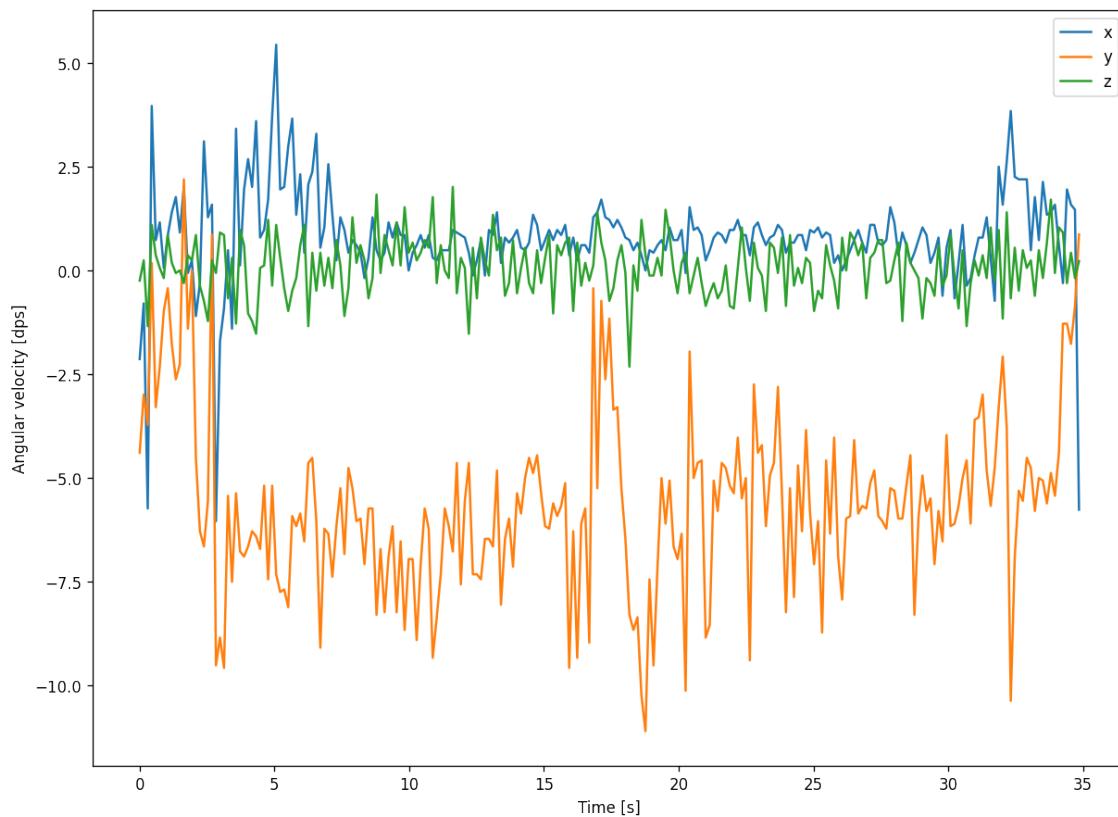


Figura 4.8: Andamento nei tre assi della velocità angolare lungo un ciclo

In questo caso il grafico delle velocità angolari è di meno immediata comprensione rispetto a quello ricavato precedentemente con l'accelerometro. Qui tutti e tre gli assi sembrano avere velocità angolare non nulla. L'asse di nostro interesse per ricavare l'angolo di inclinazione del tavolo radiogeno è però solamente l'asse Y, i valori non nulli sugli altri due assi sono dovuti alle vibrazioni del macchinario e all'offset dello stesso giroscopio. Per risalire all'inclinazione del macchinario è necessario integrare nel tempo il valore della velocità angolare.

L'integrazione rende la determinazione dell'angolo attraverso l'utilizzo del giroscopio più

complessa rispetto al metodo che utilizza l'accelerometro. I risultati di questa integrazione sono rappresentati nella figura 4.9.

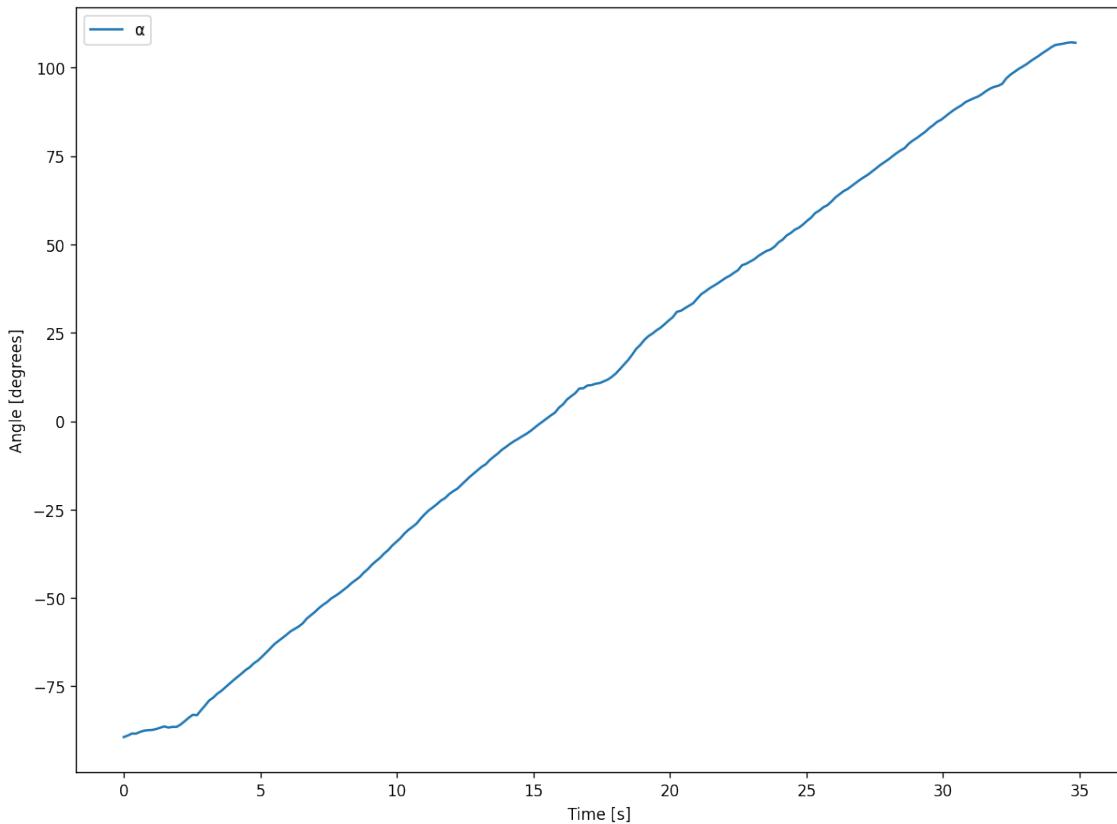


Figura 4.9: Progressione dell'angolo α nel tempo calcolata con il giroscopio

Si può notare come l'utilizzo del giroscopio per determinare il grado di inclinazione non sia troppo accurato, il valore finale trovato è di $+106^\circ$, rispetto ai $+90^\circ$ ai quali il macchinario si trova in realtà. Questo è dovuto al fatto che il giroscopio risente molto di tutte le vibrazioni del macchinario e integrando gli errori si propagano. Le misure vengono quindi sfalsate sempre più rispetto al valore reale.

Integrando tutte le imperfezioni del segnale, si ottiene alla fine un errore quasi del 18%, troppo elevato per essere ritenuto accettabile.

Tutto questo ha portato alla scelta di determinare l'angolo di inclinazione del macchinario utilizzando solamente l'accelerometro.

4.3.8 Verifica del movimento del macchinario

La verifica dell'avvenimento del macchinario si rende necessaria per avere un completo processo di automazione. Senza di essa non sarebbe possibile lasciare il macchinario senza supervisione da parte di qualcuno. Senza questa verifica il macchinario potrebbe, per svariati motivi fra i quali la rottura di una o più componenti dello stesso, smettere di muoversi e di conseguenza il conteggio dei cicli effettuati risulterebbe falso.

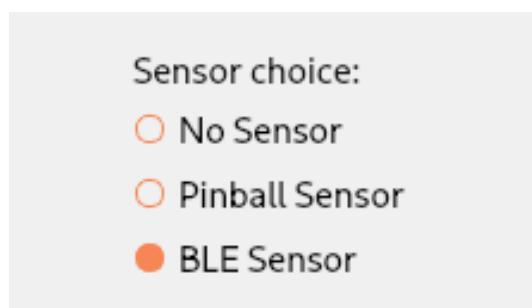


Figura 4.10: Menù di scelta del sensore

Il controllo viene effettuato in base alla tipologia di sensore selezionata dall'utente nel menù di setup, come mostrato nella figura 4.10. Di seguito verranno spiegate le varie opzioni selezionabili.

Opzione senza nessun controllo

Questa opzione non include un controllo sull'avvenimento del movimento del tavolo radiogeno. È un'opzione utile solamente nel caso in cui si vogliano eseguire test veloci senza bisogno di installazione di sensore a pallina oppure della piattaforma Arduino Nano 33 BLE Sense. Questi test necessitano di supervisione.

Sensore di inclinazione a sfera

Selezionando questa opzione, all'interno di un ciclo di movimento viene letto due volte il valore del pin digitale programmato come ingresso. Il valore del pin viene letto quando il macchinario si trova all'inclinazione di -90° e di +90°. Viene fatto un controllo sui valori letti e convalidato o meno il movimento del macchinario.

Vista la scarsa precisione che questo sensore offre, è stata implementata anche una soglia di tolleranza nella validazione del movimento del tavolo radiogeno. Può capitare infatti che in un ciclo il valore del sensore venga letto in maniera errata nonostante il corretto avvenimento del movimento del tavolo. Per ovviare a questa problematica, si è scelto di impostare come numero di errori di lettura consecutivi un massimo di 15 cicli.

Se questo valore viene superato, il programma si arresta, smette di mandare comandi da eseguire ad Arduino Uno e viene interrotto anche il conteggio dei cicli.

Arduino Nano 33 BLE Sense

L'utente può selezionare come alternativa di sensore da utilizzare per la verifica del movimento del tavolo radiogeno anche la scheda Arduino Nano 33 BLE Sense. La verifica del movimento in questo caso viene effettuata utilizzando i valori di accelerazione lungo l'asse X e l'asse Z, come riportato in precedenza nel capitolo riguardante l'hardware utilizzato. Durante il ciclo di movimento, la piattaforma rende disponibili nuovi valori di accelerazione lungo i due assi. Questi valori vengono letti dall'applicativo e da essi si risale al valore di inclinazione del macchinario, utilizzando la funzione `accelerationToDegrees()`.

```
void BLEcontroller::accelerationToDegrees(void)
{
    double dDegrees =
        → qRadiansToDegrees(qAtan(dAccelerometerX/dAccelerometerZ));
    vdDegrees.append(dDegrees);
}
```

Questa funzione risale al livello di inclinazione effettuando l'arcotangente del rapporto fra il valore di accelerazione misurato lungo l'asse X e quello misurato lungo l'asse Z. Il valore viene poi convertito da radienti a gradi.

I valori di inclinazione vengono salvati all'interno di un vettore, in modo che a fine del ciclo si possa fare un controllo su di essi. Per capire se il movimento del macchinario è avvenuto in modo corretto o meno. Questo controllo viene effettuato ad opera della funzione `checkBleSensor()`.

In questa funzione, vengono trovati il valore minimo e massimo di inclinazione raggiunti durante un ciclo e questi vengono confrontati con dei valori soglia. Un ciclo completo

comporta che il macchinario vada da un'inclinazione minima di -90° ad una massima di +90°, il controllo viene effettuato tenendo in considerazione eventuali imprecisioni del sensore, accontentandosi che il macchinario superi il range dei ±85°.

Anche in questo caso viene utilizzata una soglia di tolleranza per la validazione del movimento del macchinario: qui sono però consentiti un massimo di tre errori consecutivi nel determinare il movimento, data l'alta sensibilità del sensore.

4.3.9 Gestione del conteggio dei cicli effettuati

Il conteggio dei cicli effettuati avviene all'interno della classe supervisore, ossia la classe `ISupervisor`. All'interno di questa classe è presente la funzione `handleNewCycle()`, eseguita all'interno dello stato *write*.

La funzione in questione si occupa della conta dei cicli effettuati, confrontando il numero di comandi inviati con il numero di comandi presenti nel file di comando selezionato dall'utente. Ciascun ciclo, per essere contato, deve risultare valido e quindi passare il controllo di movimento del sensore (se questo è selezionato).

4.3.10 Stop di emergenza del macchinario

All'interno del software sviluppato sono presenti più controlli di sicurezza, i quali hanno lo scopo di evitare che il macchinario resti in movimento nel caso in cui la connessione tra l'applicativo ed Arduino Uno abbia problemi.

Uno di questi controlli è la funzione `emergencyShutdown()`, già citata in precedenza, la quale viene eseguita nel caso in cui il programma entri nello stato di *error*.

Un meccanismo di controllo ulteriore viene messo in gioco grazie alla combinazione del firmware sviluppato per Arduino e della funzione `dummyWrite()`. La funzione in questione ha lo scopo di inviare ad Arduino Uno il comando [?D013] ogni 400 ms, un comando che ha il solo scopo di domandare lo stato del pin n° 13, non utilizzato per la movimentazione del tavolo radiogeno. Inviando questo comando alla piattaforma Arduino Uno, questo viene processato e viene resettato il contatore della funzione `checkActivity()` presente nel firmware della scheda, che altrimenti avrebbe fatto scattare l'abbassamento di tutti i pin della stessa.

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Conclusioni

Questo progetto di tesi riporta una soluzione al problema dell’automazione del movimento di un tavolo radiogeno. E’ stato sviluppato un sistema in grado di sottoporre il motore passo-passo, la catena e gli altri componenti addetti al movimento a stress ciclici di fatica, senza avere la necessità di supervisionare tutto il processo.

L’obiettivo è stato raggiunto in due step: il primo step, più grezzo ma già sufficiente per la completa automazione, prevede l’utilizzo di un sensore di inclinazione a sferetta per valutare il movimento del tavolo; il secondo step invece utilizza Arduino Nano 33 BLE Sense ed il suo accelerometro per convalidare il movimento.

Si è riusciti, mediante l’impiego dell’hardware descritto e del software sviluppato, ad automatizzare tutto il processo di testing, minimizzando il tempo che è necessario dedicare al controllo del macchinario, avendo così un risparmio di risorse.

Attraverso l’impiego di un accelerometro montato su una piattaforma wireless si è reso il sistema *wearable*, in modo da minimizzare l’effetto di usura che i cicli hanno sull’apparato di automazione. Un beneficio secondario di ciò è il basso intralcio che il sistema comporta, lasciando ampio spazio di movimento al personale attorno al macchinario.

5.2 Sviluppi futuri

I risultati di questo progetto di tesi aprono la strada a sviluppi futuri.

Uno dei più interessanti possibili sviluppi è l'utilizzo della *PWM* (Pulse Width Modulation, cioè modulazione di larghezza d'impulso) dei pin di Arduino Uno utilizzati come uscita digitale per la sostituzione delle cloche di movimento del tavolo radiogeno. Utilizzando la *PWM*, si renderebbe possibile il controllo della velocità angolare con cui il tavolo radiogeno si muove. L'implementazione aggiuntiva di tale tecnica, unita alla già presente capacità del sistema di risalire all'angolazione del macchinario, permetterebbe di modulare la velocità a piacere, in base all'inclinazione del macchinario. Questo consentirebbe di limitare lo stress al quale i componenti vanno in contatto quando viene raggiunta la massima inclinazione, riducendo gradualmente la velocità angolare del macchinario e preservando così la vita utile di tali componenti.

Bibliografia e sitografia

- [1] URL: https://cdn.shopify.com/s/files/1/0438/4735/2471/products/A000066_01.iso_883x663.jpg?v=1629815860.
- [2] *Arduino Uno datasheet*. URL: <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>.
- [3] *Arduino - ArduinoBoardUno*. URL: <https://www.arduino.cc/en/pmwiki.php?n>Main/arduinoBoardUno>.
- [4] *What is Arduino UNO? A Getting Started Guide*. URL: <https://www.rs-online.com/designspark/what-is-arduino-uno-a-getting-started-guide>.
- [5] URL: https://content.arduino.cc/assets/Pinout-UN0rev3_latest.png.
- [6] URL: https://www.techmaker.it/243-large_default/sensore-di-inclinazione-tilt.jpg.
- [7] URL: <https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2016/10/Tilt-sensor-How-it-works-1.png?w=663&quality=100&strip=all&ssl=1>.
- [8] URL: https://cdn.shopify.com/s/files/1/0438/4735/2471/products/ABX00031_01.iso_640x480.jpg?v=1626445218.
- [9] *Arduino Nano 33 BLE Sense datasheet*. URL: <https://docs.arduino.cc/resources/datasheets/ABX00031-datasheet.pdf>.
- [10] URL: https://content.arduino.cc/assets/Pinout-NANOsense_latest.png.

- [11] *Arduino Nano 33 BLE Sense*. URL: <http://store.arduino.cc/products/arduino-nano-33-ble-sense>.
- [12] *iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer*. URL: https://content.arduino.cc/assets/Nano_BLE_Sense_lsm9ds1.pdf.
- [13] URL: https://docs.arduino.cc/tutorials/nano-33-ble-sense imu_accelerometer.
- [14] URL: https://docs.arduino.cc/tutorials/nano-33-ble-sense imu_accelerometer.
- [15] URL: https://content.arduino.cc/assets/NANO33BLE_V2.0_sch.pdf.
- [16] *Trasmissione seriale*. URL: https://it.wikipedia.org/w/index.php?title=Trasmissione_seriale&oldid=116460277.
- [17] URL: <https://i0.wp.com/www.moreware.org/wp/wp-content/uploads/2020/07/proto1.jpg?w=939&ssl=1>.
- [18] URL: https://cdn.rohde-schwarz.com/pws/solution/research___education_1/educational_resources/_oscilloscope_and_probe_fundamentals/05_Understanding-UART_02_w640_hX.png.
- [19] *Comprensione dell'UART*. URL: https://www.rohde-schwarz.com/it/prodotti/misura-e-collaudo/oscilloscopi/educational-content/comprensione-uart_254524.html.
- [20] *Arduino Serial Tutorial - Arduino Communication Protocols*. URL: <https://www.deviceplus.com/arduino/arduino-communication-protocols-tutorial/>.
- [21] *Bluetooth Technology Overview | Bluetooth® Technology Website*. URL: <https://www.bluetooth.com/learn-about-bluetooth/tech-overview/>.
- [22] *Overview of Bluetooth Technology*. URL: <https://inpressco.com/wp-content/uploads/2015/05/Paper191588-1592.pdf>.

- [23] URL: https://www.bluetooth.com/wp-content/uploads/2021/01/Bluetooth_Technology_Overview_Graphic.png?time=1645218476.
- [24] *Piconet*. URL: <https://en.wikipedia.org/w/index.php?title=Piconet&oldid=1023418518>.
- [25] URL: <https://www.computersec.it/2020/04/21/la-tecnologia-bluetooth/a-typical-bluetooth-piconet/>.
- [26] *Getting Started with Bluetooth Low Energy*. URL: <https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch01.html>.
- [27] *Introduction to Bluetooth Low Energy*. URL: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction>.
- [28] *Bluetooth low energy (BLE) fundamentals*. URL: <https://www.embedded.com/bluetooth-low-energy-ble-fundamentals/>.
- [29] URL: https://www.bluetooth.com/wp-content/uploads/2021/01/Bluetooth_Technology_Overview_Graphic.png?time=1645218476.
- [30] URL: https://www.oreilly.com/library/view/getting-started-with/9781491900550/assets/gsbl_0104.png.
- [31] URL: <https://learn.adafruit.com/assets/13827>.
- [32] URL: <https://learn.adafruit.com/assets/13827>.
- [33] *GATT (Services and Characteristics) - Getting Started with Bluetooth Low Energy*. URL: <https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html>.
- [34] URL: https://www.oreilly.com/library/view/getting-started-with/9781491900550/assets/gsbl_0401.png.
- [35] *Assigned Numbers*. URL: <https://www.bluetooth.com/specifications/assigned-numbers/>.

- [36] *setup()*. URL: <https://www.arduino.cc/reference/it/language/structure/sketch/setup/>.
- [37] *loop()*. URL: <https://www.arduino.cc/reference/it/language/structure/sketch/loop/>.
- [38] *Serial.begin()*. URL: <https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/>.
- [39] *ArduinoBLE*. URL: <https://www.arduino.cc/reference/en/libraries/arduinoble/>.
- [40] *Arduino_LSM9DS1 - Arduino Reference*. URL: https://www.arduino.cc/reference/en/libraries/arduino_lsm9ds1/.
- [41] *#define direttiva (C/C++)*. URL: <https://docs.microsoft.com/it-it/cpp/preprocessor/hash-define-directive-c-cpp>.
- [42] *Automa a stati finiti*. In: URL: https://it.wikipedia.org/w/index.php?title=Automa_a_stati_finiti&oldid=126021222.
- [43] *Automi a stati finiti*. URL: https://amplio.belluzzifioravanti.it/pluginfile.php/85545/mod_resource/content/1/dispensautomi.pdf.
- [44] *automi a stati finiti*. URL: <http://www.edutecnica.it/sistemi/automi/automi.htm>.
- [45] *Programmazione a stati finiti*. URL: <http://stor.altervista.org/prog-stati-finiti/prog-stati-finiti.php>.
- [46] *QSerialPort Class*. URL: <https://doc.qt.io/qt-6/qserialport.html>.
- [47] *Bluetooth LE Overview*. URL: <https://doc.qt.io/qt-5/qtbluetooth-le-overview.html>.