Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in

Ingegneria dell'Informazione

Progettazione e realizzazione di un sistema LiDAR per rilevamento e ricostruzione ambientale tridimensionale

Relatore:

Prof. Matteo Meneghini

Laureando:

Marco Panizzo

Matricola N. 1162517

Anno accademico 2021/2022

Indice

Capitolo 1

Introduzione

Il rapido sviluppo tecnologico in atto sta continuamente portando a riconsiderare quali siano effettivamente i limiti raggiungibili dalla tecnologia e quali siano i molteplici impieghi che ne conseguono. Gli strumenti sviluppati negli ultimi anni consentono alle macchine di avvicinarsi sempre più alle capacità umane.
La potenza computazionale ad oggi raggiunta ci consente di elaborare notevoli quantità di dati e informazioni, che possono essere sfruttate nella realizzazione di sistemi complessi capaci di eseguire procedure e azioni molto dettagliate ed elaborate.
Nei campi della robotica, dell'automazione e delle automotive, la specializzazione delle macchine usate ha portato all'implementazione di sensori specifici che consentono loro di percepire il mondo circostante, rendendole in grado di interagire con lo stesso.
Le tecnologie più utilizzate nella percezione spaziale sono: sensori Lidar, sensori a ultrasuoni o particolari videocamere per l'impiego di algoritmi di Computer Vision. Questi riescono a ricavare informazioni reali provenienti dallo spazio circostante e a gestirle per compiere molteplici operazioni, inerenti alle specifiche della macchina.
L'utilizzo dei sensori sopra citati può essere ritrovato in diversi sistemi per l'interazione, manipolazione, riconoscimento e ricollocazione di oggetti oppure impiegati per la guida autonoma di vetture e robot allo scopo di virtualizzare l'ambiente nel quale si trovano.
Considerando l'importanza e il valore aggiunto che questi strumenti possono apportare agli apparati tecnologici, in questa tesi viene presentata la progettazione e la realizzazione di un sistema di acquisizione tridimensionale basato su sensore Lidar. Il progetto ha lo scopo di testare e dimostrare le capacità di questa strumentazione anche in dispositivi relativamente semplici.

Capitolo 2

Presentazione

Il sistema realizzato funge da scanner ambientale, esso è in grado di eseguire una scansione dell'ambiente nel quale è situato e successivamente realizzarne una rappresentazione virtuale.
Il suo funzionamento è basato su un sensore con tecnologia Lidar (Light Detection and Ranging), il quale viene impiegato per acquisire informazioni sulla distanza che intercorre tra il sensore e una determinata superficie posta frontalmente a esso.
Per mezzo di due motori, il dispositivo viene fatto ruotare attorno all'asse orizzontale e verticale, consentendo al Lidar di analizzare tutti i punti circostanti. Ad acquisizione terminata, separatamente, il programma grafico elabora i dati importati al computer e li proietta in uno spazio tridimensionale costruendone una rappresentazione.
Il modello virtuale rappresentato è costituito da un insieme di punti (point cloud) che derivano dalle singole letture eseguite dal Lidar nelle diverse posizioni. Ne segue che il modello realizzato viene ricostruito unicamente dalle informazioni di distanza e di posizione del sensore.
La ricostruzione rappresenta con buona approssimazione la struttura dell'ambiente scansionato riportandone anche le dimensioni effettive.
Precisamente, la ricostruzione è paragonabile a una maschera delle superfici che il sensore è stato in grado di raggiungere. Essendo statico e non mobile, l'apparato riesce a scansionare i punti superficiali rivolti verso il sensore mentre nelle zone d'ombra, ovvero dove il sensore non è riuscito a raggiungere la superficie, non vi sono informazioni. Il modello virtuale, quindi, è comprensibile visivamente posizionandosi (virtualmente) nella stessa posizione del sensore. In altre parole, ponendosi nell'esatto punto dal quale sono state eseguite tutte le scansioni si è in grado di percepire in maniera corretta la ricostruzione eseguita.
Come già detto il sistema realizzato è modesto, questo è dovuto soprattutto all'utilizzo di componentistiche non all'avanguardia. In particolare, il sensore Lidar usato è il TFmini-S della Benewake, questo è costituito da un unico emettitore e un unico ricevitore ed è

in grado di compiere rilevazioni fino a una distanza massima di 12m ad una frequenza variabile tra 10 e 1000Hz. Il segnale luminoso utilizzato dal sensore rientra nello spettro dell'infrarosso (infrarosso vicino) e in particolare ha una lunghezza d'onda di 850nm.
In base alle sue caratteristiche, che verranno approfondite più dettagliatamente in seguito, le misure del sensore sono affette da errori causati da origini ambientali. [1][2]

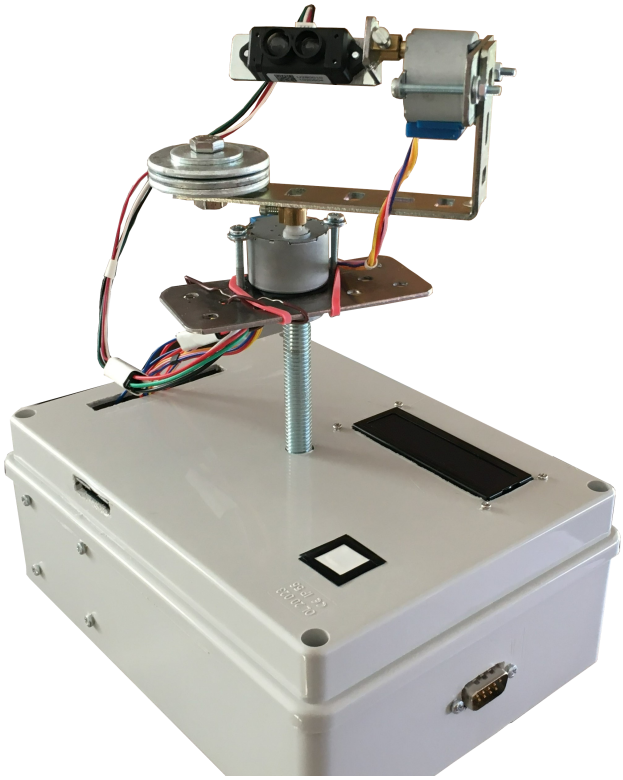


Figura 2.1: Dettaglio dispositivo

Figura 2.2: Dispositivo completo

Capitolo 3

Hardware

In questo capitolo vengono illustrati tutti i componenti che sono serviti alla realizzazione del progetto, inoltre per i principali verrà fornita anche una spiegazione dettagliata. Successivamente verrà esposta l'effettiva configurazione globale del sistema e come i moduli sono stati configurati e collegati al microcontrollore. In questa sezione verranno discusse anche le scelte fatte sulla base delle caratteristiche dei componenti. Nel presente capitolo saranno citati i protocolli di comunicazione coi quali i moduli e il controllore comunicano, questi verranno descritti in maniera più dettagliata nel successivo capitolo.

3.1 Componentistica

Gli elementi costitutivi del progetto sono:

- Scheda Arduino Uno Rev3
- Sensore Lidar TFmini-s
- Due motori stepper 28BYJ-48
- Due moduli driver ULN2003
- Modulo adattatore Micro SD
- Convertitore logico bidirezionale
- Tre moduli di alimentazione per breadboard
- Tre connettori DC
- Modulo joystick
- Scheda Micro SD
- Modulo relè (coppia)
- Display LCD 16x2 I2C
- Breadboard
- Alimentatore computer
- Connettori DB9 RS-232
- Treppiede
- Contenitore
- Supporto metallico

3.1.1 Arduino Uno Rev3

Arduino è il microcontrollore utilizzato per la gestione e il funzionamento di tutte le periferiche hardware impiegate nel progetto. Il microcontrollore in questione può essere programmato facilmente per mezzo del IDE Arduino con un linguaggio di programmazione derivante dal C++. Questa scheda può essere impiegata per progetti che non necessitano di particolari esigenze e prestazioni. Le principali caratteristiche della scheda sono:

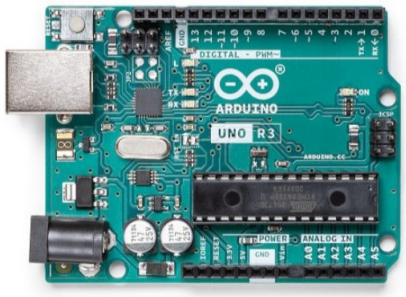


Figura 3.1: Arduino [3]

- Processore ATMega328P
- Alimentazione USB 5V
- Alimentazione presa DC 9-12V
- Frequenza di Clock 16MHz
- Memoria flash 32KB
- Memoria SRAM 2KB
- Memoria EEPROM 1KB
- Pin digitali 14 (6 PWM)
- Pin analogici 6
- Pin 5V, 3.3V e Gnd
- Protocollo USART
- Protocollo SPI
- Protocollo I2C

[3]

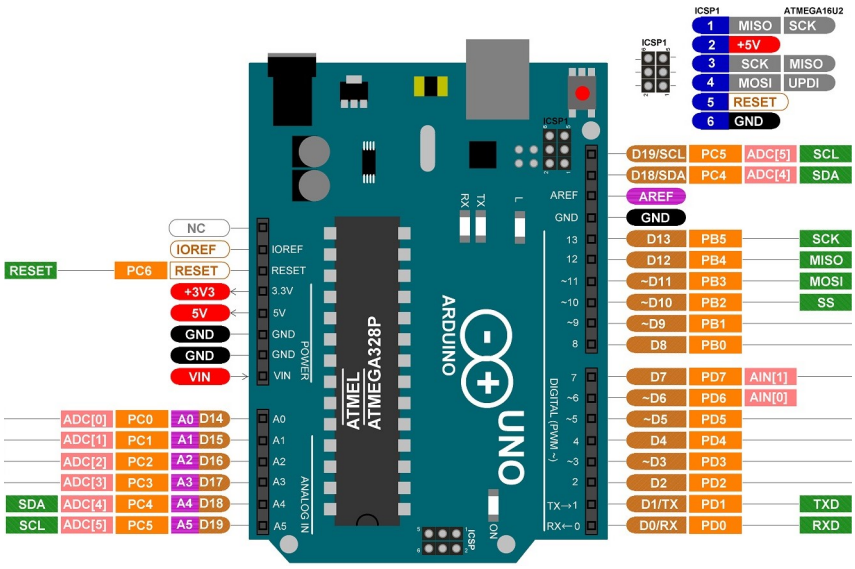


Figura 3.2: Arduino Pinout [4]

3.1.2 TFmini-s

Il TFmini-s è un sensore Lidar progettato dalla Benewake, la sua funzione è quella di misurare la distanza che intercorre tra lo stesso e oggetti e/o superfici che sono poste frontalmente a esso.
Il seguente sensore basa il suo funzionamento sul principio del tempo di volo (TOF) di un'onda elettromagnetica. Nello specifico l'onda modulata emessa dal trasmettitore ha una lunghezza d'onda di 850nm, essa non è visibile a occhio nudo trovandosi nello spettro dell'infrarosso. Le caratteristiche fondamentali del TFmini-s sono:

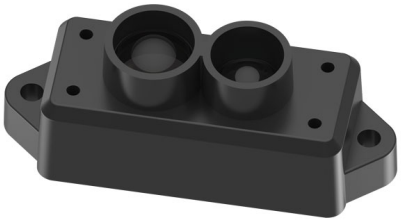


Figura 3.3: TFmini-s [1]

- Frequenza di campionamento variabile 1-1000Hz
- Range di funzionamento* 0.1-12m
- Precisione* ± 6cm se rilevazione tra 0.1-6m oltre ± 1%
- Campo visivo (FOV) 2°
- Risoluzione misura 1cm
- Alimentazione 5V
- Livello logico di comunicazione LVTTL (3.3V)
- Temperatura di funzionamento 0-60°C
- Protocollo UART (Universal Asynchronous Receiver/Transmitter)
- Protocollo I2C (Inter Integrated Circuit)

Come sopra descritto la frequenza di funzionamento è variabile ma deve essere conforme al tipo 1000/n, dove n è un numero intero, e deve essere compresa in un range [1-1000Hz]; di default il valore è impostato a 100Hz.
I parametri del range di funzionamento e della precisione sono affetti da disturbi e possono presentare ulteriori errori e restrizioni in base all'ambiente. I valori sopra indicati fanno riferimento ad ambienti chiusi con superfici riflettenti al 90%, qualora vi siano superfici scure oltre i 7m, superfici poco riflettenti, specchi (superfici altamente riflettenti) o superfici trasparenti (vetri e finestre) i dati ottenuti posso presentare errori non trascurabili o essere completamente inaffidabili. Anche le rilevazioni effettuate al di sotto dei 10cm non sono attendibili. [1][2]

Il tempo di volo viene ottenuto misurando lo sfasamento dell'onda in ricezione rispetto a quella in emissione e successivamente viene calcolata la distanza tra il sensore e il punto dove avviene la riflessione dell'onda. Il seguente schema illustra come viene calcolata la distanza dell'oggetto. Considerando: c velocità della luce, f frequenza dell'onda e $\Delta\varphi$ sfasamento dell'onda ne risulta:

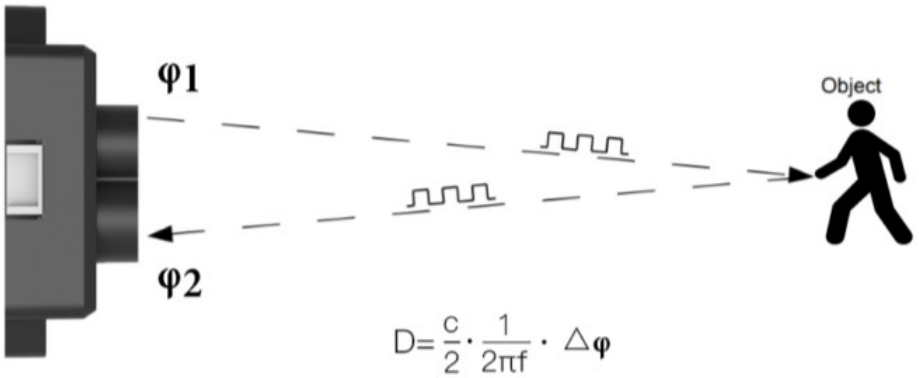


Figura 3.4: Rilevamento distanza [1]

Un particolare accorgimento va posto sul campo visivo del sensore che essendo di 2° può riscontrare dei problemi quando vi sono superfici a distanze diverse ma contigue nella vista prospettica. In questo caso il dato restituito sarà un valore compreso tra le due distanze come illustrato nella seguente figura:

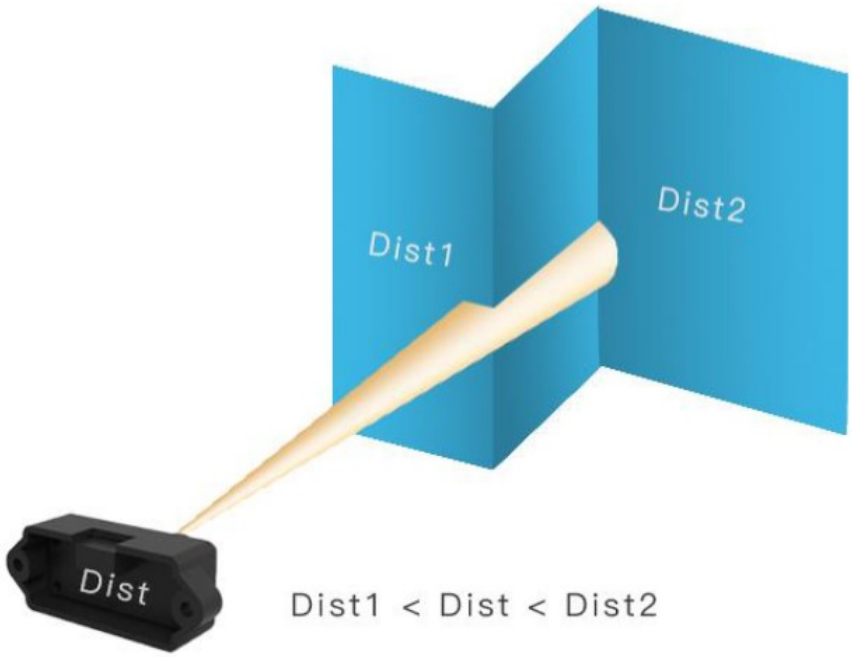


Figura 3.5: Identificazione distanza [1]

3.1.3 Stepper Motor 28BYJ-48

I motori passo passo 28BYJ-48 sono dei motori in grado di effettuare spostamenti molto precisi grazie al controllo di quattro bobine che ne pilotano il movimento. Le quattro fasi possono far muovere il rotore interno in quattro posizioni, se controllato tramite full-stepping, oppure otto posizioni, se controllato in half-stepping. Queste tipologie di controllo si riferiscono a come vengono comandate le bobine: nel primo caso (full-stepping) le bobine vengono eccitate a coppie e quindi le possibili coppie sono quattro, nel secondo caso (half-stepping) le bobine vengono eccitate sia singolarmente che a coppie, costituendo così otto possibili posizioni. Grazie a un riduttore di velocità i passi totali esterni del perno sono 2048 nel caso del full-stepping e 4096 nel caso di half-stepping. Le specifiche dei motori usati sono:

Figura 3.6: Stepper [5]

- Alimentazione 5V
- Numero di fasi 4
- Rapporto di variazione della velocità 1/64
- Passo (angolo) 5.625°/64
- Coppia 32 mNm

La configurazione degli avvolgimenti interni al motore è unipolare, ovvero tale da avere una tensione di 5V comune sulle bobine e tramite il controllo del Gnd sull'altra estremità della bobina, questa viene attivata o meno. I collegamenti appena descritti sono riportati nella seguente figura. [5]

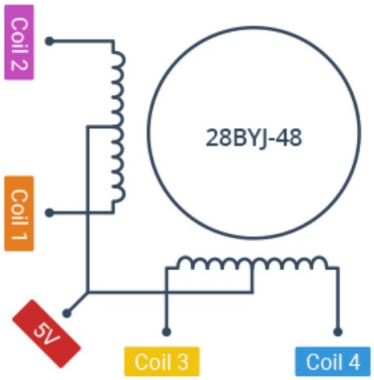


Figura 3.7: Collegamenti bobine [5]

3.1.4 Driver ULN2003

I motori stepper solitamente vengono collegati a un driver che a sua volta è collegato al controllore.
I driver hanno lo scopo di convertire i segnali logici provenienti dal controllore in segnali per l'attivazione e la disattivazione delle bobine presenti nel motore. Questo driver in particolare consente di pilotare i motori unipolari a quattro fasi come lo stepper 28BYJ-48. Tutta la potenza necessaria al motore viene fornita direttamente dal driver tramite apposito pin di alimentazione posto sul modulo, il quale può essere alimentato sia a 5V che a 12V.
I collegamenti in ingresso al modulo sono sei, quattro per identificare le bobine connessi al controllore e due dedicati all'alimentazione che solitamente è esterna.

Figura 3.8: ULN2003 [6]

I collegamenti in uscita verso il motore invece sono cinque, un Vcc comune oltre ai quattro collegamenti alle bobine. [6]

3.1.5 Modulo Micro SD

Il modulo adattatore Micro SD serve a collegare una scheda Micro SD al microcontrollore.
Essendo la scheda Arduino in uso sprovvista di una memoria capiente e non volatile, risulta necessario l'impiego di una memoria esterna per l'archiviazione e la lettura dei dati.
Il protocollo di comunicazione utilizzato per il passaggio dei dati è il SPI (Serial Peripheral Interface). Questa tipologia di comunicazione richiede l'utilizzo di quattro pin per i dati: MISO, MOSI, SCK, CS.
Per poter funzionare la memoria deve essere formattata in Fat32 o Fat16 e deve avere una dimensione inferiore a 32GB.
Il modulo richiede un'alimentazione a 5V. [7]

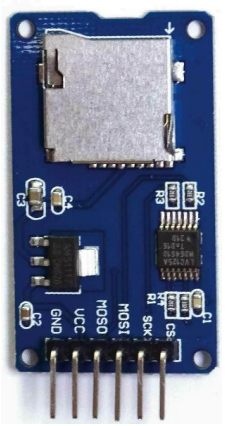


Figura 3.9: Modulo Micro SD [7]

3.1.6 Convertitore logico

Il convertitore logico bidirezionale consente di convertire i segnali logici da 3.3V a 5V e viceversa.
In questo caso anziché andare a definire tramite software una tensione di riferimento di 3.3V per la comunicazione proveniente dal sensore Lidar diretta al microcontrollore, si è preferito utilizzare il seguente modulo così da semplificare le operazioni di rielaborazione dei dati.
Questo modulo quindi non risulta essere strettamente necessario nella realizzazione del sistema progettato.

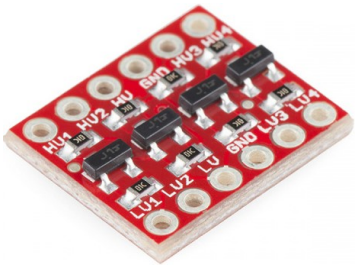


Figura 3.10: Convertitore logico [8]

Per poter funzionare il convertitore ha bisogno delle tensioni di riferimento da convertire (da 3.3V a 5V) che andranno collegate rispettivamente ai pin LV (Low Voltage) e HV (High Voltage) oltre al Gnd.
La scheda è divisa in quattro canali, ognuno dei quali può eseguire una conversione logica indipendente dalle altre. [8]

3.1.7 Display LCD 16x2

Il display impiegato è un display a cristalli liquidi, monocolore, retroilluminato, il quale presenta un'area di scrittura di due righe da 16 caratteri ciascuna.
Al display è associato un ulteriore modulo che consente di utilizzare il protocollo di comunicazione I2C così da impiegare esclusivamente due pin per la trasmissione dati invece dei sei richiesti nella configurazione di base. Questo modulo è posto sul retro ed è saldato allo stesso display, inoltre è presente anche un potenziometro per la regolazione della retroilluminazione del pannello.
Entrambi i moduli (Display e modulo di comunicazione) condividono l'alimentazione per mezzo di un unico collegamento con tensione a 5V. [9]

Figura 3.11: Display LCD 16x2 [9]

3.1.8 Scheda relè

La seguente scheda presenta due relè accoppiati, ognuno dei quali è pilotato indipendentemente dai rispettivi pin d'ingresso.
Anche le morsettiere sono separate e per ciascun relè sono presenti i contattati normalmente aperto, normalmente chiuso e la linea principale che si collega a uno dei due.
In termini di potenza ciascun relè può supportare un carico di 50V AC e 5A o 30V DC e 5A.
La scheda permette di alimentare separatamente le bobine dei relè necessarie per la commutazione del contatto comune, così facendo al controllore non viene richiesta l'erogazione della potenza necessaria, ma solo il controllo dello stato del pin per la commutazione. [10]

Figura 3.12: Relè [10]

3.1.9 Joystick

Il modulo in questione è joystick di tipo analogico e funziona per mezzo di due potenziometri da 10KΩ, uno per l'asse x e uno per l'asse y.
In base a come la leva è posizionata i due potenziometri avranno una determinata resistenza, diversa per ogni possibile posizione.
Alimentando il modulo e misurando i valori di tensione provenienti dai pin collegati ai rispettivi cursori dei potenziometri, si può determinare con buona precisione la posizione della leva.
Oltre a fornire i valori dei potenziometri vi è un ulteriore pulsante meccanico attivabile premendo il joystick lungo l'asse verticale (z).

Figura 3.13: Joystick [11]

Questa tipologia di cursori è soggetta a usura derivante dalla continua frizione tra cursore e resistenza, maggiore è lo stato di usura più i valori restituiti saranno errati, conseguentemente anche la posizione rilevata della leva sarà scorretta. [11]

3.1.10 Alimentatore PS-5251-08

L'alimentatore utilizzato per fornire l'energia necessaria a tutto il progetto è un alimentatore proveniente da un case di un computer desktop.
In particolare il modello in questione è il PS-5251-08 della Liteon.
Caratteristiche dell'alimentatore:

Figura 3.14: Alimentatore PS-5251-08 [12]

- Input 200-240V/ 4A
- Output 250W Max
- Configurazione singolo output:
  - +5V 25A max
  - +3.3V 18A max
  - +12V 14A max
  - -12V 0.8A max
- Configurazione combinata output:
  - +5V e +3.3V 165W max
  - +12V e +5V 218W max

3.1.11 Scheda alimentazione breadboard

Questa scheda consente di portare l'alimentazione necessaria alla breadbord, ai moduli o ad Arduino stesso, mediante appositi pin e porta USB. Caratteristiche di funzionamento:

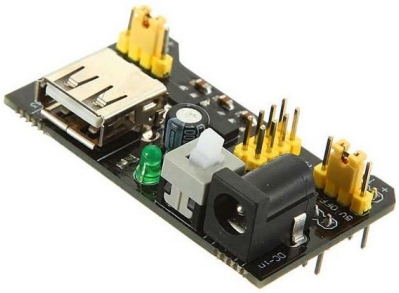


Figura 3.15: Scheda per alimentazione [13]

- Input 6.5-12V
- Output 3.3-5V
- Corrente di uscita massima: 700 mA
- USB utilizzabile sia come input alla scheda di alimentazione che come output, con tensione 5V in entrambe le configurazioni

[13]

3.2 Configurazione del sistema

In questa sezione vengono illustrate la rete di collegamenti e le decisioni prese nella realizzazione del progetto riguardanti la componentistica e infine viene fornito lo schema del circuito.
Tenendo in considerazione tutte le caratteristiche dei vari moduli sopra elencati, si è cercato di bilanciare tutte le prestazioni in maniera tale da consentire il corretto funzionamento dei moduli stessi e rendere quanto più prestante l'intero progetto.
A partire dal microcontrollore si è verificata la capacità dello stesso di poter gestire tutti i moduli necessari e quindi se i pin a disposizione fossero sufficienti. Per far fronte al considerevole numero di collegamenti richiesti si è optato per delle configurazioni particolari come l'utilizzo del display LCD con modulo I2C, così da poter utilizzare esclusivamente due pin per la comunicazione dei dati. Il display è stato collegato ai pin: SDA e SCL di Arduino.
Altra scelta fondamentale è stata la configurazione in multiplexing per il controllo dei due motori stepper agendo sull'alimentazione dei driver (ai quali ognuno dei motori è collegato) mediante il controllo di un relè. Il relè porta la potenza necessaria sulla linea comune mentre i driver sono collegati ai contatti normalmente aperto e normalmente chiuso; in base al motore che si vuole controllare, si attiva o disattiva la bobina del relè, il quale alimenterà il driver che a sua volta fornirà l'alimentazione al motore.

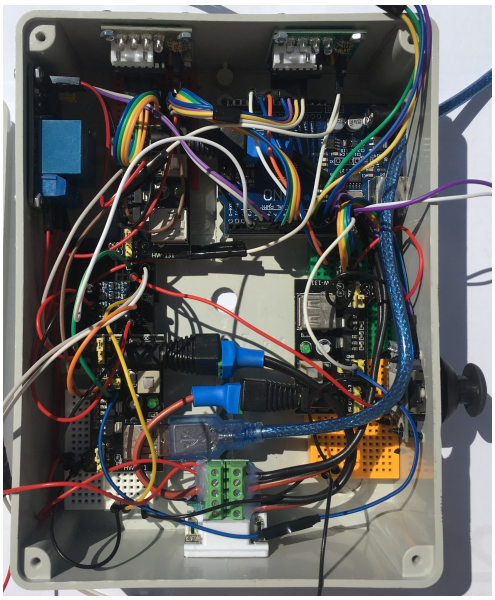


Figura 3.16: Circuito reale

Questa configurazione è possibile, in questa circostanza, perché i due motori non sono mai attivi contemporaneamente e l'alternanza tra i due avviene con un periodo sufficientemente lungo da non gravare particolarmente sulla componente meccanica dei relè.
Per il controllo dei motori i pin collegati ai driver sono: D6, D7, D8, D9.
Per quanto riguarda il pin per il controllo del relè per la selezione dei motori, si è utilizzato il pin D5.
Il sensore Lidar utilizza l'interfaccia UART con due cavi per la comunicazione dei dati: uno per la trasmissione e uno per la ricezione. Dovendo esclusivamente ricevere dati dal sensore, si è usato unicamente il collegamento per la trasmissione dal Lidar e la ricezione ad Arduino.
Il sensore è soggetto a surriscaldamento e nonostante il manuale riporti un corretto funzio-

namento fino a 60°C, sperimentalmente superati i 55°C i dati cominciano a essere distorti. Impostando l'acquisizione dati del sensore a una frequenza di 100Hz rispetto ai 1000Hz massimi raggiungibili, il sensore si surriscalda ma si assesta intorno a 46/47°C. Non essendo prevista la sospensione del sensore mediante comunicazione con lo stesso, si è utilizzato il secondo relè per il collegamento dell'alimentazione allo stesso, da disattivare qualora il sensore si surriscaldi troppo.
Il segnale dati trasmesso dal sensore Lidar in direzione di Arduino passa prima per il convertitore logico digitale che amplifica i segnali da 3.3V a 5V e arriva al pin di ricezione di Arduino D3.
Il pin usato invece per il controllo del secondo relè è il pin D4.
Il modulo Micro SD, utilizzando il protocollo di comunicazione SPI, necessita di quattro linee dati identificate come MISO, MOSI, SCK, CS, i rispettivi pin utilizzati sono: D12, D11, D13, D10.
Il joystick richiede tre pin, due per le letture dei valori degli assi x e y e uno per identificare lo stato del pulsante. Per leggere i valori delle resistenze variabili, servono due pin analogici in configurazione input, A0 per l'asse x e A1 per l'asse y, il pulsante invece è associato al pin digitale D16 che corrisponde al pin analogico A2, ma usato in configurazione input digitale.
Oltre ai collegamenti col microcontrollore ci sono le linee per l'alimentazione provenienti dell'alimentatore PS-5251-08 dirette verso: il display, la linea comune del relè per i motori e le schede di alimentazione per breadboard che a loro volta instradano tutte le linee verso i vari moduli e Arduino. L'alimentatore PS-5251-08 è stato ricablato per consentire di raggiungere le schede con tre linee da 12V e raggiungere il modulo LCD e i motori con due linee da 5V.
Per consentire il collegamento e la rimozione dell'alimentatore si sono utilizzati due connettori DB9 RS-232, uno (femmina) come terminale di uscita del PS-5251-08 e uno (maschio) come ingresso del sistema.
La scelta di utilizzare questo connettore risiede sulla possibilità di mantenere separate le linee di alimentazione sfruttando la numerosità dei contatti.
L'uso dei moduli di alimentazione al posto di una linea diretta, consente la protezione sia dei moduli a valle del collegamento sia dell'alimentatore stesso a monte, infatti le schede presentano una protezione da cortocircuito e una protezione da sovratensione. Allo stesso modo il collegamento alla porta USB di tipo B di Arduino consente una protezione aggiuntiva del microcontrollore essendo tale porta protetta da sovratensione.
La seguente figura illustra tutto lo schema del progetto. Per precisione i collegamenti (gialli) a 12V provenienti dal connettore DB9 in direzione delle schede di alimentazione, si collegano mediante il DC Jack delle schede e non tramite i due contatti illustrati.

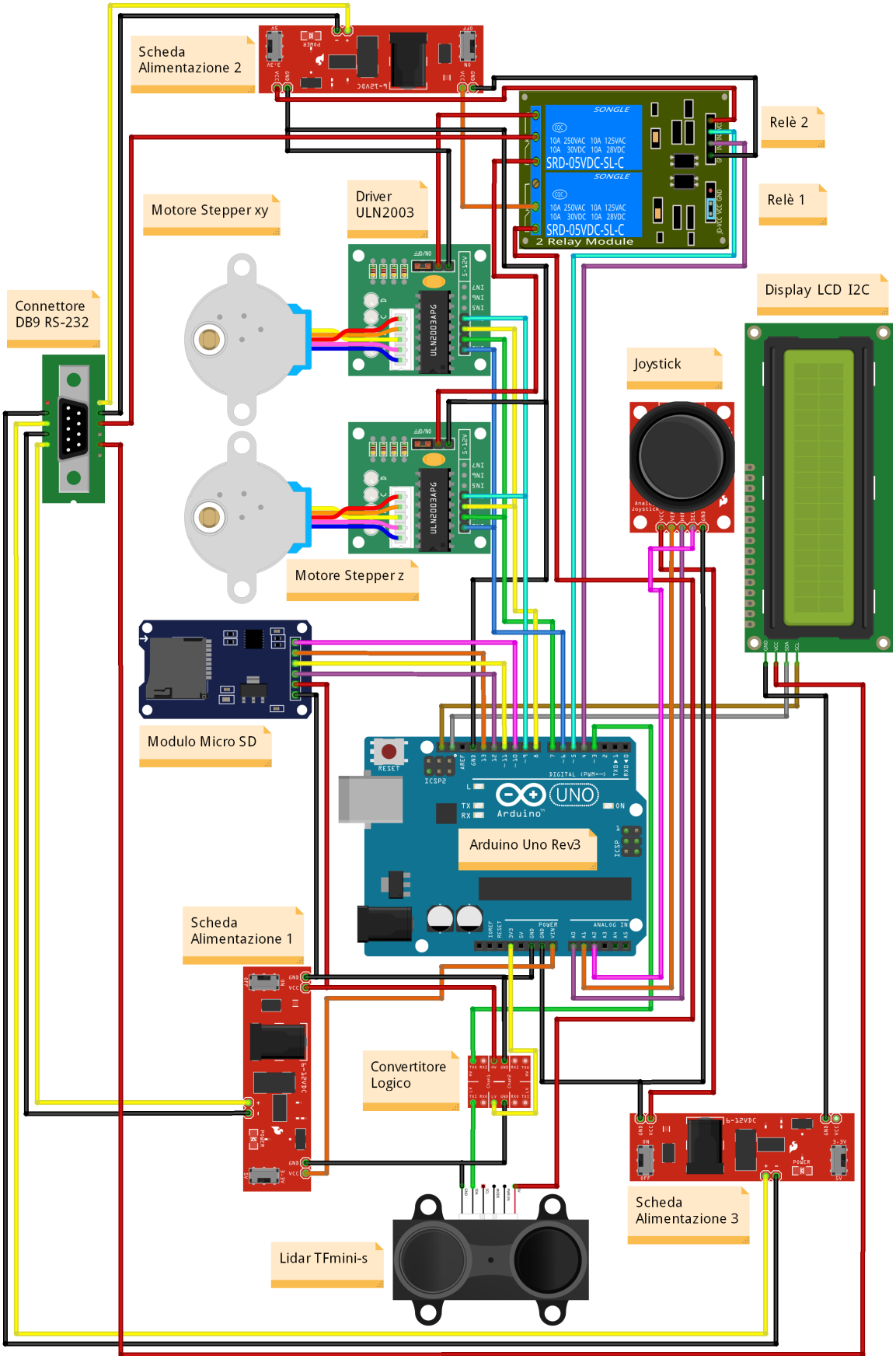


Figura 3.17: Schema circuitale

Capitolo 4

Software

In questo capitolo vengono illustrati: gli ambienti di sviluppo software utilizzati, i relativi programmi e una breve descrizione dei protocolli di comunicazione sopra citati.
La spiegazione dei programmi descriverà anche la sequenza di azioni che il sistema compie, spiegando così il funzionamento dello stesso.
Alla fine delle sezioni vengono riportati due schemi a blocchi che mostrano i passi principali dei due codici.
I software utilizzati per la stesura dei programmi sono:

- **Arduino IDE (Integrated Development Environment)**: è un ambiente di sviluppo open-source creato per rendere più intuitiva la scrittura e facilitare il caricamento del codice nelle schede che lo dovranno eseguire.
  Il linguaggio di programmazione è derivante e compatibile (utilizzando classi apposite) con il C/C++.
  I microcontrollori supportati sono diversi ma l'ambiente di sviluppo è ottimizzato per le schede Arduino che sfruttano processori Atmel con architettura AVR; [14] [15]

- **Matlab**: è un ambiente di sviluppo basato sull'ottimizzazione di calcoli in formato matriciale; può essere utilizzato per analizzare dati, sviluppare algoritmi, elaborare segnali e in molti altri ambiti grazie alla capacità di elaborare grandi quantità di dati in modo efficiente.
  L'ambiente risulta molto flessibile e ricco di funzionalità, implementabili anche grazie alla presenza di un toolbox che concentra e fornisce particolari strumenti in base all'area d'interesse. [16]

4.1 Programma Arduino

Il codice sviluppato per il microcontrollore Arduino rappresenta la serie di comandi necessari a gestire tutto il sistema di acquisizione dati.
Nella scrittura sono state utili le librerie messe a disposizione dall'IDE stesso, consentendo l'utilizzo di funzioni e di strutture preconfigurate per l'utilizzo specifico di alcuni dei moduli. Le librerie utilizzate sono:

- math.h
- SoftwareSerial.h
- LiquidCrystal_I2C.h
- Stepper.h
- SD.h
- SPI.h
- Wire.h

La libreria math.h consente di aggiungere funzioni matematiche aggiuntive come quelle trigonometriche.
La libreria LiquidCrystal_I2C consente l'utilizzo delle funzioni dedicate al relativo modulo semplificando così la procedura per la trasmissione delle stringhe sul display per la comunicazione esterna.
SD.h permette la corretta configurazione del modulo adattatore Micro SD e fornisce operazioni per scrittura, lettura e creazione di file all'interno della memoria.
Per il controllo dei motori stepper, invece, si è usata la relativa libreria: Stepper.h, che inizialmente richiede l'inserimento dei dati del motore; essa presenta delle funzioni che gestiscono in autonomia la sequenza di bobine da attivare in base al valore dei passi da eseguire.
A differenza delle librerie sopra illustrate, le rimanenti si occupano della configurazione dei protocolli di comunicazione che sono stati anticipati nel capitolo precedente e che verranno illustrati di seguito. Per la corretta gestione delle interfacce di comunicazione quindi sono state usate librerie come la SoftwareSerial.h che serve a configurare e utilizzare linee alternative di comunicazione seriale, differenti da quella preconfigurata sulla scheda Arduino. La comunicazione seriale di default si utilizza per mezzo dei pin D0 e D1 rispettivamente utilizzati come RX0 e TX0 necessari però alla comunicazione con il computer mediante collegamento USB.
L'implementazione di questa libreria ha consentito di configurare il pin D3 della scheda Arduino come porta RX del controllore per ricevere i dati provenienti dal sensore Lidar che comunica con interfaccia UART.
Wire.h consente la configurazione del protocollo I2C per la comunicazione con il display LCD I2C.
La libreria SPI.h gestisce invece i collegamenti e il trasferimento di dati con il modulo Micro SD.

4.1.1 Protocolli di comunicazione

I2C

Questo protocollo, denominato Inter Integrated Circuit e abbreviato in I2C, I^2C o IID, utilizza due linee bidirezionali: la SCL e la SDA.
La SCL è dedicata al segnale di clock essendo la connessione tra il controllore (master) e il modulo (slave) di tipo sincrono.
La linea SDA invece è dedicata alla trasmissione dei dati.
Oltre a queste linee vi è anche un collegamento per la Vcc e uno per la Gnd.
Utilizzando un unico collegamento per i dati ed essendo la comunicazione bidirezionale, la connessione è di tipo Half-duplex. [17]

SPI

Il Serial Peripheral Interface è una tipologia di interfaccia di tipo seriale che sfrutta per la comunicazione quattro linee: MISO, MOSI, SCK, CS.
Il MISO (Master Input Slave Output) è il collegamento che consente la trasmissione dei dati in output dal modulo (slave) e in ricezione al controllore (master).
Invece il collegamento MOSI (Master Output Slave Input) trasmette dal controllore (master) verso i moduli (slave).
La linea SCK (Serial Clock) trasmette la frequenza di clock in emissione dal controllore e il CS (Chip Select) si occupa della selezione e attivazione della comunicazione tra un determinato modulo e il controllore. Quest'ultimo consente di utilizzare più moduli mediante medesima connessione, selezionando la periferica di interesse tramite apposito pin, instaurando una trasmissione dati con la stessa.
Avendo a disposizione due linee dedicate a trasmissione e ricezione, e potendole sfruttare contemporaneamente, la connessione è di tipo Full-duplex. [17]

UART

Universal Asynchronous Receiver Transmitter, come esplicato dal nome, questa tipologia di comunicazione utilizza segnali asincroni ed è in grado di trasmettere e ricevere allo stesso tempo dati tra il controllore e le periferiche mediante l'utilizzo di due linee dati separate. In questo caso le due linee prendono il nome di RX (ricezione) e TX (trasmissione). Il tipo di collegamento è sempre Full-duplex. [17]

4.1.2 Codice e logica

Il codice può essere suddiviso inizialmente in due macro gruppi: uno relativo alla configurazione di tutti i parametri e uno relativo alla pura scansione. Nella parte di configurazione vanno inserite manualmente:

- Altezza dal suolo del sensore
- Livello di dettaglio della scansione

L'inserimento dei dati e tutto il controllo viene effettuato direttamente sul dispositivo mediante interfacciamento con il joystick per la selezione dei dati e il display LCD per la visualizzazione dei parametri e degli stati.
La risoluzione della scansione e del modello ricostruito dipende da come viene impostato il parametro di definizione iniziale, questo va a definire il numero di passi da eseguire in successione tra una scansione e un'altra, questo può essere scelto sulla base di potenze di due (2^k con $0 \leq k \leq 10$) quindi 1, 2, 4, 8, 16, 32, etc. . Questo parametro sarà usato sia per i passi dei paralleli sia per quelli dei meridiani.
L'altezza del sensore da terra è necessaria alla ricostruzione della posizione del sensore ovvero dell'angolo di inclinazione dello stesso in riferimento all'asse verticale.
Una volta impostata l'altezza del sensore, viene richiesto un posizionamento dello stesso che consenta di individuare la distanza di un punto sul terreno riproducendo così l'ipotenusa di un triangolo.
La ricostruzione dell'angolo ω di inclinazione del sensore deriva dalla formula:

$$\omega = arcos(H/dist)$$

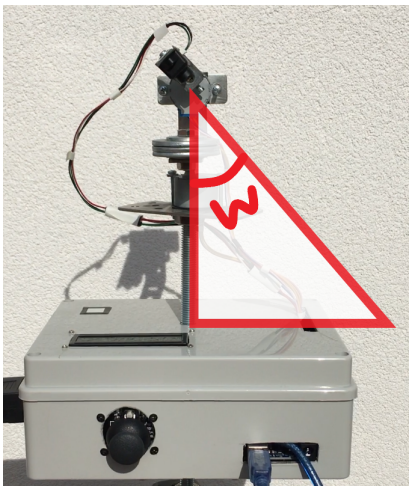


Figura 4.1: Angolo ω

dove H identifica l'altezza dal suolo e $dist$ la distanza rilevata dal sensore e quindi anche l'ipotenusa del triangolo che si crea col suolo.
Ottenuto l'angolo, si calcolano i passi necessari a portare il Lidar in posizione verticale verso il suolo.
Il calcolo viene eseguito mediante la formula:

$$Step_z = \omega/(2\pi/2048)$$

Il posizionamento verticale, consente di capire approssimativamente se la rilevazione e il calcolo siano avvenuti correttamente. Successivamente il sensore si posiziona ad un angolo

minimo di partenza che è stato impostato manualmente pari a 45° così da essere sicuri che il contenitore del circuito non interferisca con le rilevazioni, essendo questa relativamente vicina al sensore, restituendo con ciò dati falsati.
Prima di cominciare la scansione avviene un controllo della presenza della scheda Micro SD e in caso non fosse presente, viene restituito un messaggio di errore indicativo della mancanza. Se invece è presente, viene generato il file txt dove verranno salvati i dati. Questa fase conclude il macro gruppo riguardante la configurazione. Questo può essere riassunto e illustrato sequenzialmente nei seguenti sottoprogrammi del codice:

1. Impostazione manuale dell'altezza ($hight()$);

2. Impostazione manuale del livello di dettaglio della scansione ($definition()$);

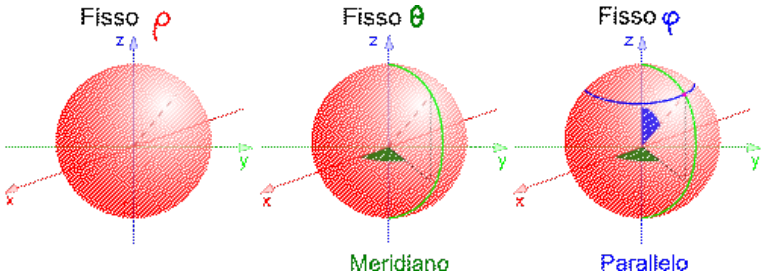
3. Posizionamento manuale del sensore ($positioning()$);

4. Centratura automatica verticale e posizionamento iniziale ($centering()$);

5. Controllo e configurazione Micro SD ($sd\_setting()$).

L'altro macro gruppo è quello che si occupa di eseguire ciclicamente le scansioni lungo i paralleli e i meridiani.
La scrittura sul file inizia e termina ad ogni meridiano, ovvero il controllore abilita la scrittura sul file subito prima dell'inizio della rilevazione del primo punto del meridiano, dopo la rilevazione dell'ultimo punto dello stesso meridiano viene interrotta la scrittura e il file viene salvato; successivamente il sensore si sposterà al meridiano successivo e ripeterà il ciclo aprendo nuovamente il medesimo file e aggiungendo i nuovi dati.
La scansione del singolo meridiano parte dall'angolo minimo impostato (45° ma può essere ridotto); il TFmini-s esegue la rilevazione e trasmette i dati ad Arduino. I valori delle distanze vengono correlate ai rispettivi angoli creati dalla posizione del sensore rispetto alla posizione di partenza; gli angoli sono ricavati dalla conversione dei passi che sono serviti ai motori per portarsi in quella posizione.
Le terne di valori costituiscono un sistema di coordinate sferiche.
L'insieme di tutte queste terne espresse in coordinate sferiche può essere già utilizzato per ricreare il modello virtuale in un ambiente grafico.
Essendo però stata utilizzata una funzione particolare di Matlab che richiede che le terne siano espresse in formato cartesiano, si è prima eseguita una conversione da coordinate

sferiche a coordinate cartesiane mediante le funzioni:



$$x = dist \cdot sin(\varphi) \cdot cos(\theta)$$
$$y = dist \cdot sin(\varphi) \cdot sin(\theta)$$
$$z = dist \cdot cos(\varphi)$$

Figura 4.2: Coordinate sferiche [18]

Come illustrato dalla figura il parametro φ è l'angolo formato con l'asse verticale positivo, mentre l'angolo ω è l'angolo formato con l'asse verticale negativo, ovvero il supplementare di φ; ai fini della rilevazione delle coordinate sferiche viene utilizzato l'angolo φ.
E' necessaria pertanto la conversione di ω in φ per i meridiani dove la scansione avviene dal basso verso l'alto. I meridiani adiacenti vengono scansionati in verso opposto così da sfruttare la posizione di termine del sensore che viene usata come posizione di partenza della scansione del nuovo meridiano; pertanto la rilevazione partendo dall'alto, considera direttamente il parametro φ.
La rilevazione degli angoli avviene mediante la conversione del numero di passi eseguiti dai motori.
Le terne in formato cartesiano vengono infine caricate nel file txt; ogni terna occupa una riga e ogni coordinata è separata da uno spazio (carattere della tastiera).
La chiusura del file, terminata la procedura per ciascun meridiano, consente di non perdere la totalità dei dati in caso di arresto anomalo o malfunzionamento.
Durante la rilevazione di ogni punto del meridiano vengono "chiamati" altri due sottoprogrammi che si occupano del controllo della temperatura del sensore e della richiesta di interruzione manuale della scansione.
Se la temperatura del sensore supera i 55°C la scansione si interrompe e viene attivato il relè per la disattivazione della linea di alimentazione del Lidar, per un minuto, trascorso il quale viene ripristinata l'alimentazione e, se la temperatura è tornata a valori standard, la scansione riprende da dove si era interrotta.
Manualmente, mediante la pressione del joystick, si può far terminare la scansione conservando tutti i dati scritti fino a quel momento nel file.
Finita la scansione di tutti i punti di un meridiano, si passa a quello successivo e verrà ripetuto ciclicamente tutto il processo fino a quando sarà "invocata" la fine, determinata o dall'avvenuta rilevazione di tutti i punti o dall'interruzione manuale.
Il sottoprogramma per il passaggio al meridiano successivo presenta anche un'importante correzione per le rilevazioni che vengono eseguite con livello di dettaglio a uno o due passi.

Essendo i motori controllati in Full-stepping, le posizioni dei rotori sono quattro e questo non crea problemi se il livello di dettaglio è tarato su quattro passi o su suoi multipli che coincidono con potenze di due superiori o uguali a quattro $(2^2, 2^3, etc)$ perché ciò implica che le bobine attive di partenza coincidono con quelle attive di fine ovvero i rotori interni tornano nella stessa posizione da dove sono partiti.
Il problema si pone negli altri casi dato che a seguito di uno spostamento di uno o di due passi per passare al meridiano successivo, le bobine attive risulteranno essere diverse da quelle di partenza, ed essendo i due motori in multiplexing, alla commutazione dell'alimentazione all'altro motore (per il suo controllo), questo si troverà ad avere delle bobine già eccitate, diverse da quelle di partenza, comportando così uno spostamento indesiderato prima della scansione del meridiano. Se questo errore non venisse corretto a ogni meridiano scansionato verrebbe introdotto un nuovo errore che modificherebbe l'angolo di partenza della rilevazione e il modello virtuale risulterebbe tutto scalato e deformato. Per correggere lo sfasamento, prima di cominciare la rilevazione, il sensore viene fatto ruotare di tanti passi quanti quelli necessari per portarlo nella posizione corretta.
Il macro gruppo relativo all'esecuzione delle scansioni può essere identificato dalla sequenza ciclica di questi sottoprogrammi:

1. Apertura file per scrittura ($sd\_open()$);
2. Rilevamento distanze punti del meridiano ($parallel()$), costituito da:
   (a) Rilevamento distanza ($lidar()$);
   (b) Conversione coordinate($conversion()$);
   (c) Controllo interruzione manuale($break()$);
   (d) Controllo temperatura ($temp()$);
3. Prossimo meridiano ($meridian()$), che svolge anche le operazioni di:
   (a) Controllo errore di multiplexing;
   (b) Chiusura file txt.

La suddivisione in tante funzioni separate (o sottoprogrammi) è stata necessaria per la stabilità del sistema.
Suddividendo il programma si sono potute utilizzare più variabili locali e meno globali mantenendo in memoria così solo i parametri finali. Nelle prime prove pratiche, infatti, nonostante il codice fosse corretto e ciclico, i dati trasmessi serialmente al computer (per il controllo), che dovevano essere sempre presenti (anche in caso di errore), a volte non erano presenti; solo dopo alcune modifiche, l'IDE di Arduino durante il debug del codice ha rilevato e riportato che il sistema poteva essere instabile a causa dello spazio insufficiente di memoria.

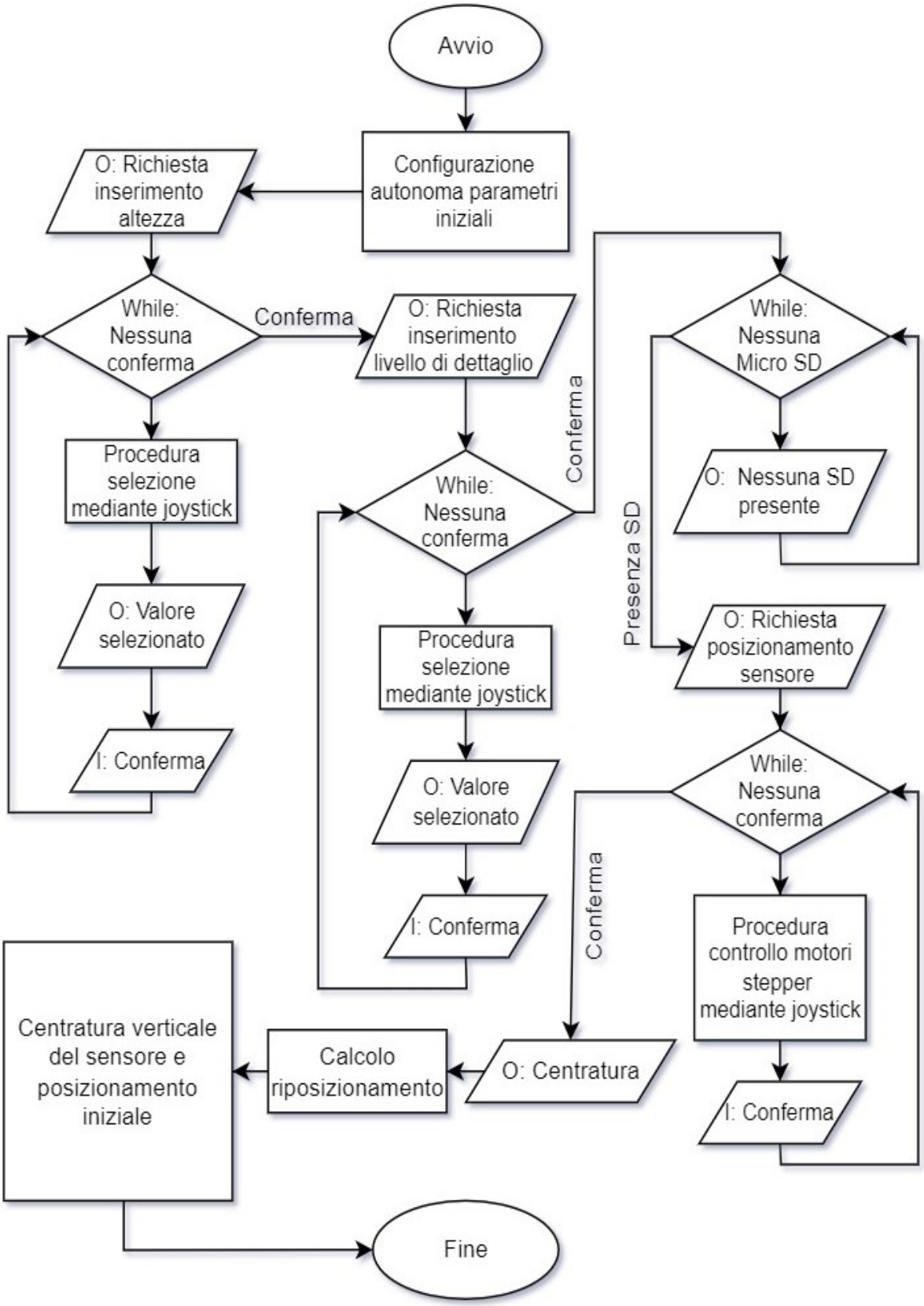


Figura 4.3: Diagramma di flusso relativo alla configurazione

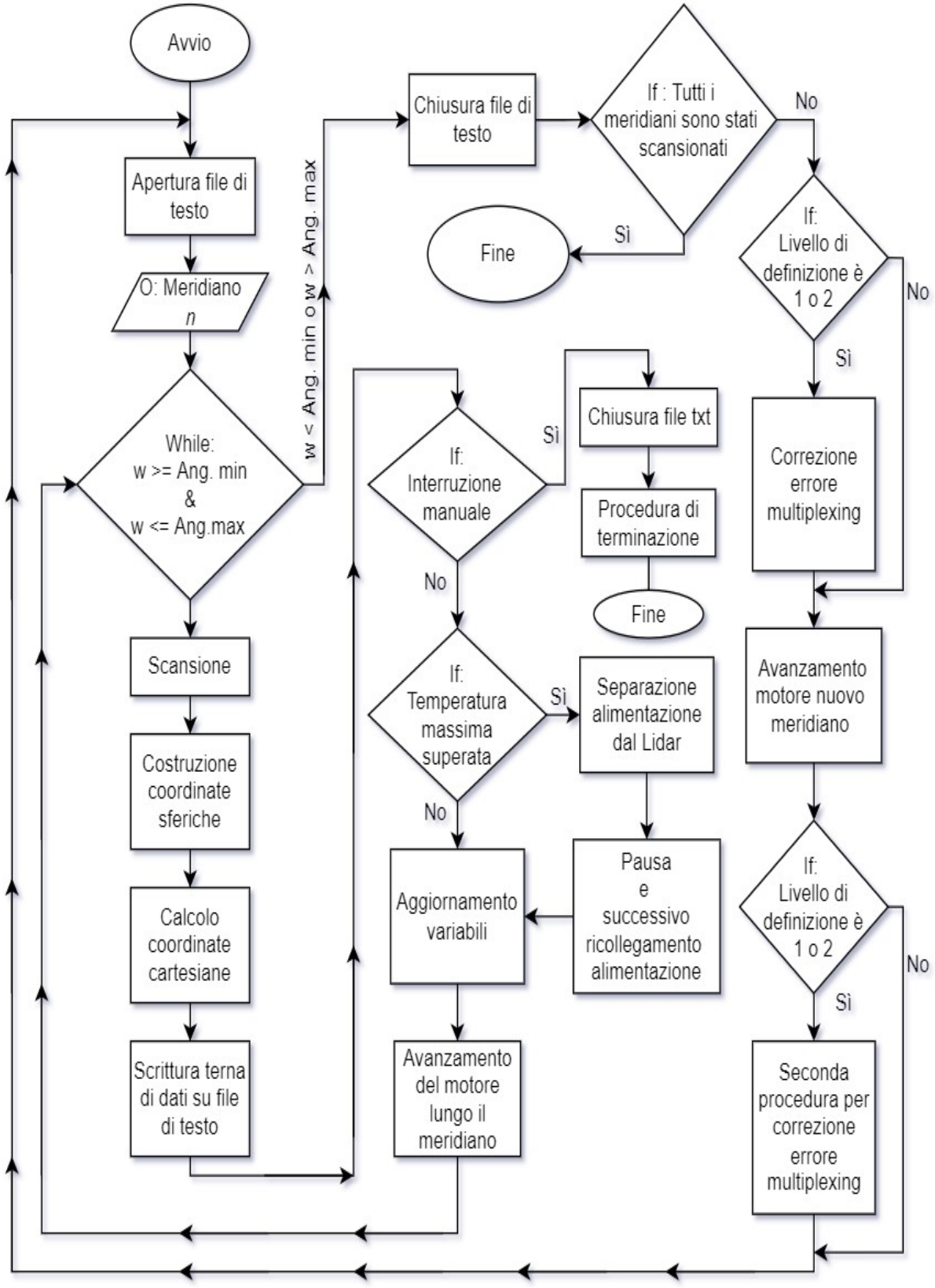


Figura 4.4: Diagramma di flusso relativo alla scansione

4.2 Programma Matlab

Il programma scritto con Matlab si occupa dell' elaborazione dei dati raccolti dal sensore. Una volta collegata la scheda Micro SD al computer, il file txt deve essere trasferito nella stessa cartella dove è presente il programma Matlab e, una volta avviata l'esecuzione del programma scritto, questo è in grado di recuperare e caricare i dati presenti nel file di testo senza bisogno di trasferirli manualmente; per fare ciò nel codice è stato inserito il nome del file che ad ogni scansione rimane lo stesso.
I dati presenti all'interno del file di testo vengono caricati in una matrice denominata: *data*, a tre colonne e k righe, dove k rappresenta il numero di rilevazioni salvate nel documento.
Le tre colonne rappresentano i valori delle coordinate cartesiane del punto rilevato.
Oltre a queste variabili che identificano il punto nello spazio, è stato utilizzato un altro parametro che rappresenta il valore della distanza del medesimo punto rispetto alla posizione del sensore che combacia con il punto di coordinate cartesiane $(0, 0, 0)$.
Questo parametro consente di associare un determinato colore al relativo punto così da avere un'informazione visiva aggiuntiva relativamente alla profondità aiutando così la comprensione del modello. Più il punto è distante dal centro più il suo colore sarà scuro (blu), più è vicino più sarà chiaro (giallo).
Le seguenti immagini raffigurano i dati salvati nel file di testo e successivamente come questi vengono acquisiti dal programma Matlab.

```
1039.7412000 0.0000000 -1332.3055000
1097.5004000 0.0000000 -1337.3081000
1136.5205000 0.0000000 -1317.5437000
1228.9528000 0.0000000 -1355.9407000
1296.3362000 0.0000000 -1361.5846000
1371.7871000 0.0000000 -1371.7871000
1448.4941000 0.0000000 -1379.0811000
1481.9021000 0.0000000 -1343.1180000
```

Figura 4.5: File txt

1.0397e+03	0	-1.3323e+03
1.0975e+03	0	-1.3373e+03
1.1365e+03	0	-1.3175e+03
1.2290e+03	0	-1.3559e+03
1.2963e+03	0	-1.3616e+03
1.3718e+03	0	-1.3718e+03
1.4485e+03	0	-1.3791e+03
1.4819e+03	0	-1.3431e+03

Figura 4.6: Matrice 'data' Matlab

Name	Value
c	1311408x1 double
data	1311408x3 double
x	1311408x1 double
y	1311408x1 double
z	1311408x1 double

Figura 4.7: Variabili Matlab

Le quaterne di dati vengono poi inserite nella funzione Matlab *pcshow* nel seguente modo:

$$pcshow([x(:), y(:), z(:)], c)$$

Questa funzione genera un ambiente virtuale tridimensionale nel quale disporre ordinatamente i punti con coordinate cartesiane: x, y, z, associandovi anche il colore c. [19]
Il grafico tridimensionale inoltre dispone di assi cartesiani per illustrare le dimensioni effettive del modello; inoltre è stato aggiunto un punto rosso in corrispondenza del punto $(0, 0, 0)$ rappresentativo della posizione del Lidar.
I grafici realizzati verranno discussi e commentati più dettagliatamente nel capitolo riguardante i risultati ottenuti ma di seguito vengono già forniti alcuni esempi.

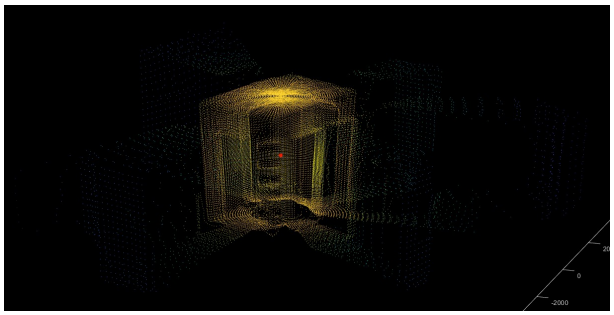


Figura 4.8: Corridoio vista laterale con livello di dettaglio otto

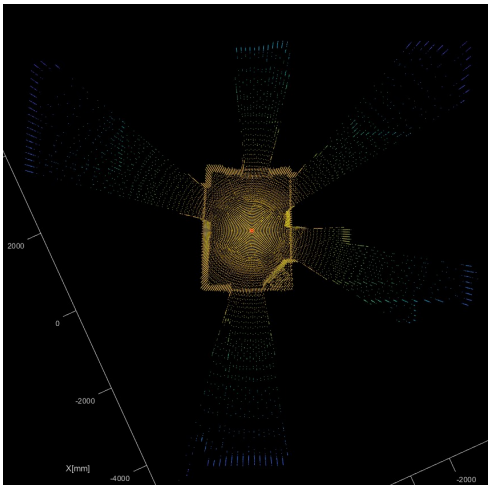


Figura 4.9: Corridoio vista dall'alto con livello di dettaglio otto

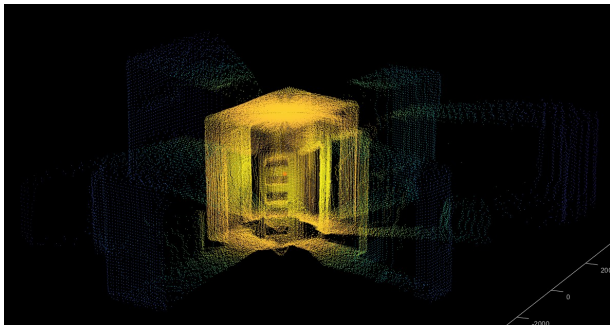


Figura 4.10: Corridoio vista laterale con livello di dettaglio quattro

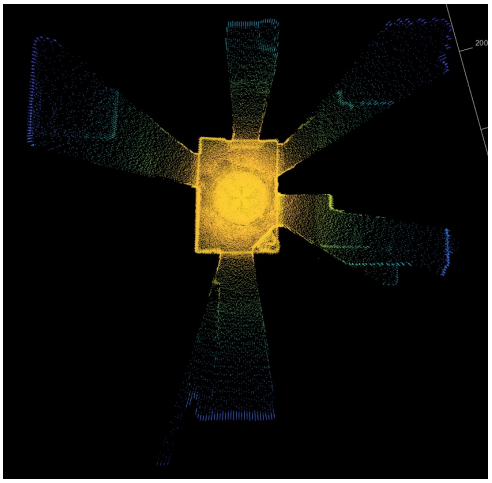


Figura 4.11: Corridoio vista dall'alto con livello di dettaglio quattro

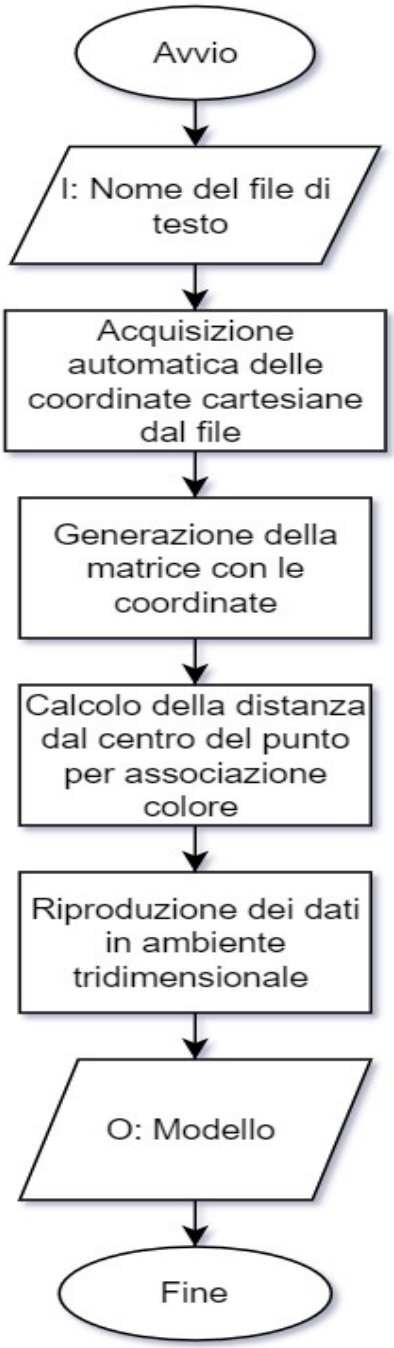


Figura 4.12: Diagramma di flusso della procedura di virtualizzazione

Capitolo 5

Funzionamento e risultati

In questo capitolo verrà prima illustrata in maniera sintetica l'interazione con il sistema, il suo funzionamento pratico e infine verranno presentati e discussi i risultati ottenuti.

5.1 Funzionamento pratico

Per una scansione corretta innanzitutto il dispositivo va posto al centro della stanza e alzato ad un livello adeguato alle esigenze; per una comprensione visiva del modello l'altezza dovrebbe essere approssimativamente tra 1.40m e 1.70m.

Successivamente questo va messo in posizione perfettamente orizzontale mediante una livella per consentire che il piano "xy" virtuale possa essere quanto più rappresentativo del piano orizzontale reale.

L'immagine a fianco raffigura in rosso il piano xy che si può usare come riferimento per tarare la centratura del sensore, questo combacia con la latra di supporto del secondo motore.

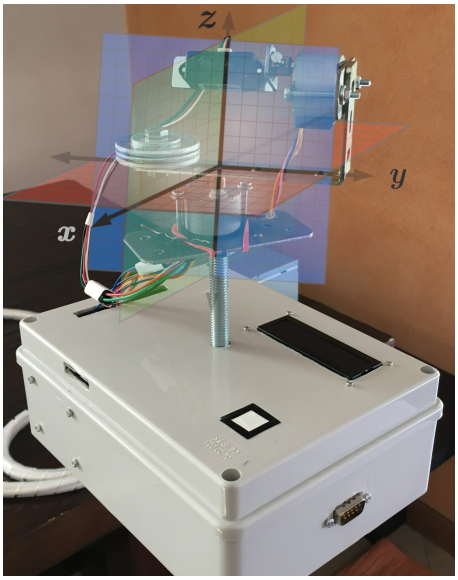


Figura 5.1: Piano xy

Una volta posizionato correttamente il dispositivo può essere acceso grazie al collegamento dell'apposito alimentatore.

Se non già presente, andrà inserita la memoria Micro SD; sul display verrà indicata la mancanza della stessa se non presente.

Il display comunicherà all'operatore di inserire i dati relativi all'altezza del sensore Lidar e il livello di dettaglio con il quale eseguire la scansione.

Impostati questi dati verrà richiesto il posizionamento del sensore in direzione del suolo

per formare l'ipotenusa del triangolo teorico già discusso in precedenza.
Confermato il posizionamento il sensore prima si porterà autonomamente in posizione verticale e poi comincerà la scansione. Una volta cominciata la rilevazione non servirà fare altro; il sistema si gestirà in autonomia fino a scansione completata.

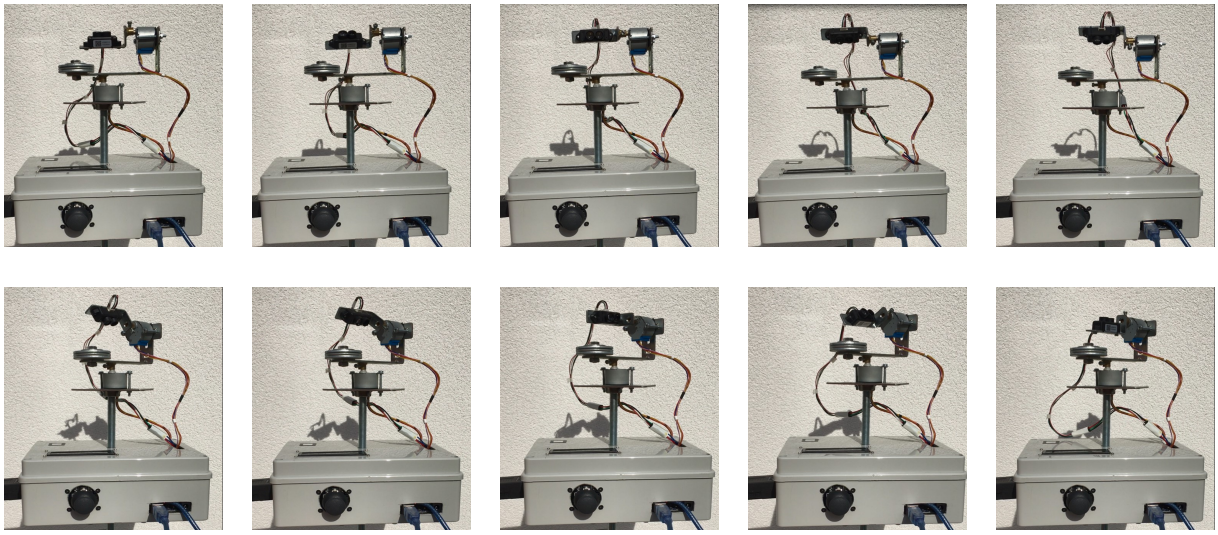


Figura 5.2: Sequenza cronologica movimenti

Durante tutto il procedimento sul display verrà indicato quale meridiano è in corso di rilevazione e in caso fosse raggiunta la temperatura critica del sensore, sarà presente un avviso di segnalazione di sospensione e successivamente di ripresa.
Come già illustrato si può interagire in qualsiasi momento con il dispositivo interrompendo manualmente la scansione mediante pressione del joystick, al quale seguirà un avviso di interruzione e di fine scansione.
Finita la scansione si può procedere con la rimozione della memoria e l'inserimento della stessa nel computer.
Trasferito il file nella stessa cartella del programma Matlab, si procede alla costruzione del modello avviando il programma.

5.2 Risultati

I risultati sono rappresentati dalla correttezza e dal dettaglio del modello ricostruito.
I vari ambienti ricostruiti sono molto dettagliati, soprattutto se eseguiti con livello di dettaglio "elevato" come uno o due passi. Le scansioni fatte con livello di dettaglio a otto o più passi non forniscono sufficienti dettagli a meno di ambienti con dimensioni ridotte.
La seguente immagine raffigura un modello ricostruito da una scansione del soggiorno eseguita con livello di dettaglio uno.

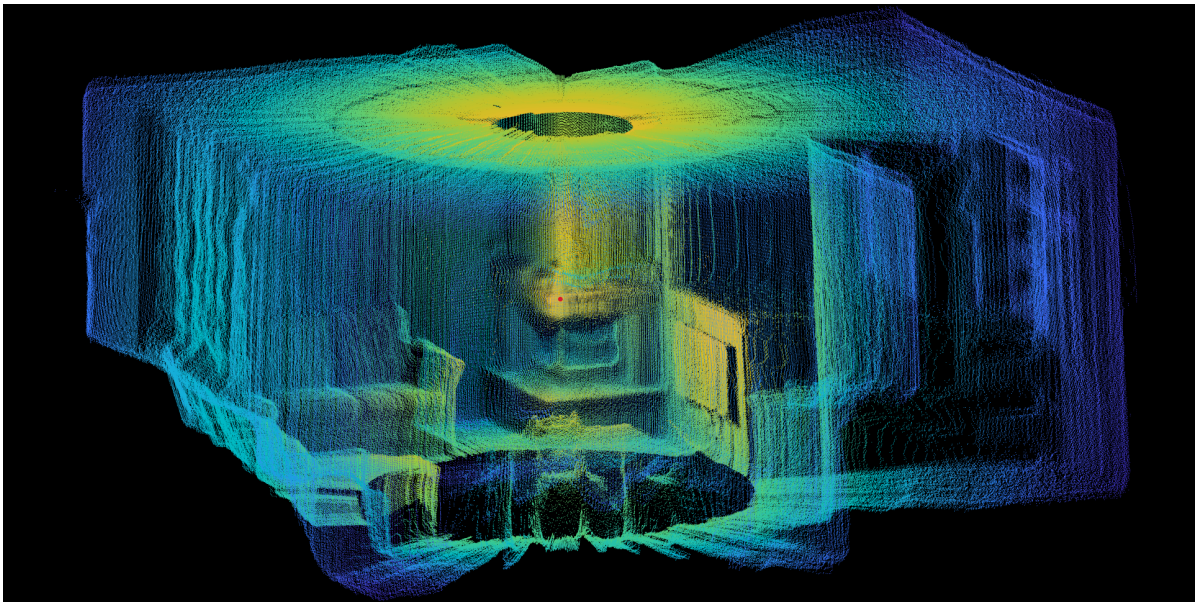


Figura 5.3: Modello salotto intero a definizione uno

Come si può notare dalla figura le superfici sono ben delineate e il dettaglio consente di interpretare i diversi oggetti nella stanza (divano, caminetto, televisore, ecc.). Questa particolare scansione ha richiesto un tempo di scansione di circa 17 ore per un totale di 1.311.408 punti utilizzati; per velocizzare il processo, sono state rimosse due sezioni circolari immediatamente sopra e sotto il dispositivo, come si può vedere dalla figura.
La tabella a seguito illustra il tempo approssimativo richiesto per le diverse scansioni complete con livello di dettaglio diverso:

Livello di dettaglio	Punti totali	Tempo richiesto
1	2.097.152	27h
2	524.288	8h
4	131.072	3h 40’
8	32.768	50’

Tabella 5.1: Descrizione temporale rapportata al livello di dettaglio

Per un corretto confronto qualitativo del modello sopra presentato, si è frammentato lo stesso suddividendolo in tre sezioni e ponendo il punto di osservazione in coincidenza del sensore, rappresentato dal punto rosso in figura 5.3.
Di seguito sono rappresentati i frammenti del modello e le relative immagini reali.

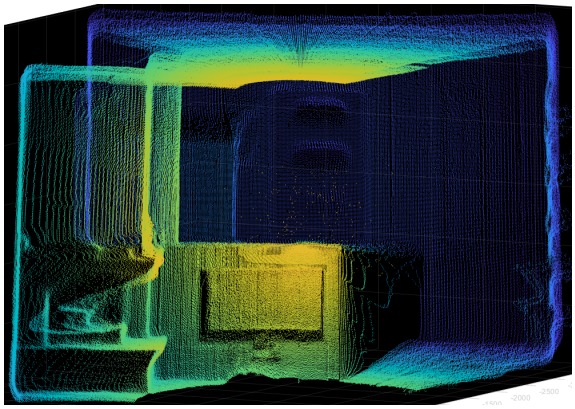


Figura 5.4: Modello parte I

Figura 5.5: Foto parte I

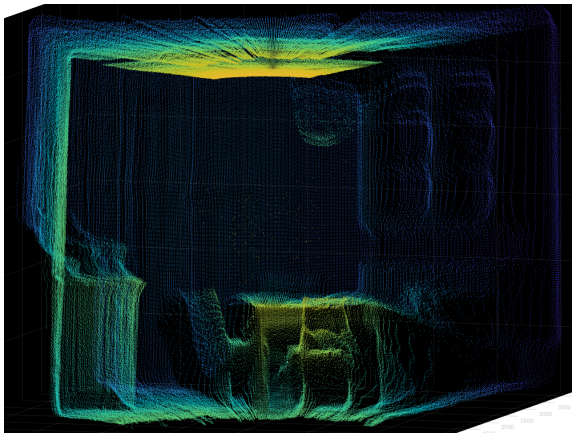


Figura 5.6: Modello parte II

Figura 5.7: Foto parte II

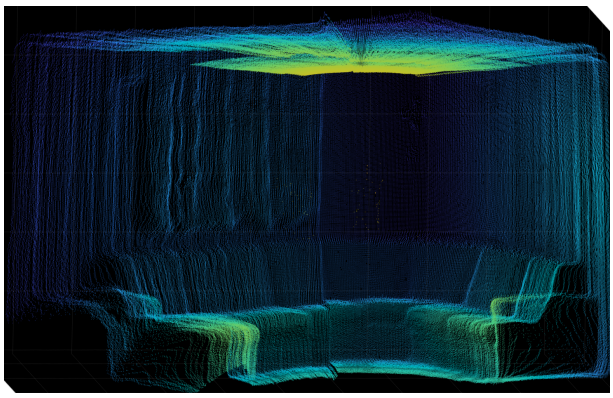


Figura 5.8: Modello parte III

Figura 5.9: Foto parte III

Oltre al modello sopra riportato, ne sono stati creati altri come quelli già illustrati nelle figure 4.6 e 4.8.
Tutti i modelli rispecchiano le dimensioni e le caratteristiche degli ambienti rilevati.
I modelli costruiti con scansioni ad alta densità di punti, come nel caso di rilevazioni con livello di dettaglio uno o due, sono in grado di restituire più caratteristiche dell'ambiente analizzato, ma allo stesso tempo vengono maggiormente evidenziate le discontinuità tra i punti vicini, dovute agli errori di misura.
Utilizzando modelli meno dettagliati, queste discontinuità si percepiscono in maniera ridotta facendo sembrare le rappresentazioni migliori, più omogenee, anche se in realtà non lo sono.
Considerando il livello di dettaglio e i tempi necessari per le acquisizioni dei punti per una scansione completa, un giusto bilanciamento lo si può trovare nelle scansioni eseguite con livello di dettaglio quattro. Queste forniscono sufficienti informazioni pur mantenendo comunque un tempo di rilevazione contenuto (circa 3h 40').
Mentre, una scansione eseguita con livello di dettaglio otto potrebbe essere utilizzata ai fini di acquisizione delle dimensioni dell'ambiente, in quanto non fornisce sufficienti informazioni per identificare gli oggetti presenti nella stanza.

Capitolo 6

Considerazioni finali

In fase di progettazione, il sistema inizialmente ideato prevedeva l'utilizzo di componenti differenti come: servo motori, un'alimentazione portatile, un microcontrollore più avanzato e un supporto fisico diverso.
Durante la fase di sviluppo si è preferito modificare certi elementi che hanno condotto alla realizzazione del dispositivo sopra illustrato.
Queste modifiche sono state effettuate per garantire un miglior funzionamento di tutto il sistema.
La sostituzione dei servo motori con dei motori stepper ha aumentato notevolmente la definizione raggiungibile in fase di scansione.
La struttura in metallo, creata appositamente per il sostegno del sensore, è stata fondamentale per poter porre il sensore al centro delle rilevazioni, consentendo di identificarlo come punto materiale in fase di ricostruzione; la precedente struttura introduceva numerose criticità a riguardo e sarebbe stato necessario modificare tutte le rilevazione per poter consentire il riposizionamento virtuale del sensore al centro del modello, introducendo così ulteriori errori dovuti ad approssimazioni; con la nuova struttura si sono potuti eliminare tutte queste rielaborazioni.
Un'importante modifica è stata fatta a livello di interazione manuale con il sistema che prima prevedeva la modifica dei parametri unicamente a livello di codice attraverso il computer, questo è stato fatto mediante l'inserimento del joystick e del display LCD.
Aggiungendo i due moduli sopra riportati, i due relè e sostituendo i motori, è risultato necessario cambiare il tipo di alimentazione, non più adeguata al carico richiesto.
Inizialmente la sostituzione è stata effettuata mediante l'impiego di due alimentatori distinti che fornivano contemporaneamente la potenza necessaria al circuito; questi però non si sono rivelati essere adeguati per il tipo di carico, introducendo diversi problemi sopratutto nell'utilizzo del joystick, il quale restituiva valori falsati.
Infine l'implementazione dell'alimentatore PS-5251-08 unito al bilanciamento tutti i carichi distribuiti sulle schede di alimentazione hanno risolto i problemi riguardanti l'alimen-

tazione, garantendo anche una load regulation sufficientemente elevata per la corretta lettura di tutte le posizioni del joystick.
Il microcontrollore inizialmente predisposto al controllo del sistema era un clone di un Arduino Mega modificato con l'aggiunta di moduli integrati per il Wi-Fi e il bluetooth.
Lo scopo primario era quello di utilizzare per l'appunto queste due tipologie di connessioni per la condivisione in tempo reale dei dati. Questo però non è stato possibile in quanto la stabilità delle connessioni non permetteva una trasmissione sicura dei dati, inoltre la stessa velocità di rilevazione ne avrebbe risentito. A seguito di questi problemi, è stato preferibile utilizzare un microcontrollore standard (Arduino Uno) e una trasmissione dei dati mediante Micro SD.
Nella realizzazione della versione finale del dispositivo si sono riscontrate comunque delle difficoltà correlate a difetti di tipo hardware riguardanti alcuni moduli e alcuni cavi, che sono stati successivamente sostituiti con componenti funzionanti.
Il sistema finale, come visto nel capitolo precedente è in grado di scansionare e ricostruire molto dettagliatamente l'ambiente; ciò non toglie che per poter generare un modello, il tempo necessario è elevato. Questo infatti rappresenta uno dei maggiori limiti all'utilizzo, rendendolo inadatto a scansioni e rappresentazioni in tempo reale.
Diversi sensori Lidar oggi presenti in commercio, consentono di eseguire scansioni in tempo reale ma molti si focalizzano su un numero ristretto di elementi. Alcuni di questi eseguono la rilevazione di un'unico parallelo, altri invece emettono un griglia di punti in un'unica direzione, e solo i più sofisticati ed elaborati consentono di eseguire contemporaneamente scansioni in tempo reale, complete come quelle realizzate dal dispositivo costruito.
Ognuna di queste tipologie è focalizzata su un determinato ambito di utilizzo.
Il progetto realizzato infatti non è stato ideato con l'obbiettivo di restituire rappresentazioni in tempo reale, ma è stato sviluppato per la rilevazione e la riproduzione virtuale di ambienti, operazione che svolge correttamente.
Molti limiti del dispositivo come: la precisione, il range di misura, le velocità di acquisizione e di spostamento, possono essere ricondotte alle singole componenti utilizzate, essendo queste molto economiche. Una maggiore ottimizzazione del codice potrebbe portare ad un'ulteriore riduzione dei tempi ma paragonata alle prestazioni dei componenti, questa risulterebbe essere irrilevante. Questo sistema, come esposto nel capitolo introduttivo, consente la rilevazione e la rappresentazione virtuale di ambienti, di cui i modelli rappresentano proprio il risultato finale.
Grazie alle modifiche apportate in fase di sviluppo oltre ad avere raggiunto gli obiettivi iniziali del progetto, si sono ottenuti risultati migliori delle stesse previsioni.
I risultati discussi nel capitolo cinque illustrano infatti la fedeltà dei modelli alla realtà che il sistema è stato in grado di ricostruire nonostante le componentistiche non fossero all'avanguardia.

Questi modelli possono essere successivamente utilizzati per la riproduzione effettiva dell'ambiente mediante l'unione delle immagini reali e delle superfici virtuali associate, Matlab stesso implementa delle funzioni specifiche in grado di eseguire quanto illustrato; oppure possono essere sfruttati per una prima interazione con la stanza reale da parte di dispositivi autonomi; oppure per effettuare una rilevazione di forma e dimensioni di ambienti, nonché degli elementi che li compongono (porte, finestre, arredamento, etc.) per poter poi essere implementati su programmi finalizzati alla progettazione architettonica.

Bibliografia

[1] Benewake (Beijing) Co. Ltd. *TFmini-S_ Product Manual*. URL: http://en.benewake.com/support_data?catename=03c54a91-8f9a-409f-985d-37acc68f7bf4&supportid=5253ad27-fad0-41ca-b0ee-89e01af31a7c. (accessed: 19.08.2022).

[2] Benewake (Beijing) Co. Ltd. *TFmini-S_ Datasheet*. URL: http://en.benewake.com/support_data?catename=03c54a91-8f9a-409f-985d-37acc68f7bf4&supportid=5253ad27-fad0-41ca-b0ee-89e01af31a7c. (accessed: 19.08.2022).

[3] Arduino. *Arduino Uno R3 Reference Manual*. URL: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf. (accessed: 19.08.2022).

[4] Electronics Hub Ravi Teja. *Arduino Pinout*. URL: https://www.electronicshub.org/arduino-uno-pinout/. (accessed: 19.08.2022).

[5] Last Minute Engineers. *Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino*. URL: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/. (accessed: 19.08.2022).

[6] Rorirobot. *ULN2003A Stepper Motor Controller*. URL: https://win.adrirobot.it/motor_driver/uln2003_stepper/uln2003_stepper_Motor_Controller.htm. (accessed: 19.08.2022).

[7] Arduino Facile. *Micro SD Card con Arduino*. URL: https://arduinofacile.altervista.org/progetti/moduli/micro-sd-card-con-arduino/. (accessed: 19.08.2022).

[8] Sparkfun Jim Blom. *Bi-Directional Logic Level Converter Hookup Guide*. URL: https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide/all. (accessed: 19.08.2022).

[9] Rorirobot. *Display LCD I2C 16X2 con retroilluminazione Blu*. URL: https://win.adrirobot.it/display_lcd/display-lcd-i2c-16x2-con-retroilumminazione-blu.htm. (accessed: 19.08.2022).

[10] AZ-Delivery. *Modulo a 2 relè 5V con innesto di basso livello*. URL: https://www.az-delivery.de/it/products/2-relais-modul. (accessed: 19.08.2022).

[11] AZ-Delivery. *Modulo Joystick KY-023 per UNO R3*. URL: https://www.az-delivery.de/it/products/joystick-modul. (accessed: 19.08.2022).

[12] Amazon. *Alimentazione PC Liteon PS-5251-08 250W*. URL: https://www.amazon.it/Alimentazione-Liteon-PS-5251-08-DX2300-440569-001/dp/B07TL12P1K. (accessed: 19.08.2022).

[13] Elcoteam. *Modulo Alimentazione per Breadboard a 830 Contatti*. URL: https://www.elcoteam.com/modulo-alimentazione-per-breadboard-a-830-contatti. (accessed: 19.08.2022).

[14] Progetti Arduino. *Guida al software di Arduino IDE*. URL: https://www.progettiarduino.com/guida-al-software-di-arduino-ide-sketch.html. (accessed: 19.08.2022).

[15] Arduino. *Arduino IDE 1.8.19*. URL: https://www.arduino.cc/en/software. (accessed: 19.08.2022).

[16] Matlab. *Matematica. Grafica. Programmazione.* URL: https://it.mathworks.com/products/matlab.html. (accessed: 19.08.2022).

[17] Luigi Morelli. *Protocolli di comunicazione su SBC: UART, I2C, SPI*. URL: https://www.moreware.org/wp/blog/2020/07/08/protocolli-di-comunicazione-su-sbc-uart-i2c-spi/. (accessed: 24.08.2022).

[18] YouMath Salvatore Zungri. *COORDINATE SFERICHE*. URL: https://www.youmath.it/lezioni/analisi-due/varie/2278-coordinate-sferiche.html. (accessed: 03.09.2022).

[19] Matlab. *pcshow Plot 3-D point cloud*. URL: https://it.mathworks.com/help/vision/ref/pcshow.html. (accessed: 19.08.2022).

Ringraziamenti

Vorrei ringraziare tutte le persone che mi hanno permesso di arrivare fin qui e di portare a termine questo percorso.
Un sentito grazie al mio relatore Meneghini Matteo per la sua disponibilità e tempestività ad ogni mia richiesta. Grazie per avermi fornito ogni indicazione e consiglio utile alla realizzazione del progetto e dell'elaborato finale.
Un ringraziamento speciale va fatto ai miei genitori che da sempre mi sostengono e supportano nella realizzazione dei miei progetti.
Grazie a Martina per essere sempre stata presente e avermi aiutato durante tutti questi anni.