



UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Sviluppo e porting di firmware per drone di tipo Ducted Fan

Sottotitolo della Tesi

Candidato:

Robert Laurentiu Mincu

Relatore:

Andrea Bonci

Correlatore:

Prof. Nome Correlatore 1

Prof. Nome Correlatore 2

Anno Accademico 2024-2025



UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Sviluppo e porting di firmware per drone di tipo Ducted Fan

Sottotitolo della Tesi

Candidato:

Robert Laurentiu Mincu

Relatore:

Andrea Bonci

Correlatore:

Prof. Nome Correlatore 1

Prof. Nome Correlatore 2

Anno Accademico 2024-2025

UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE
Via Brecce Bianche – 60131 Ancona (AN), Italy

La dedico a Giada, Alina e Paolo

Ringraziamenti

Grazie a Tizio e Caio

Ancona, Dal 1/2/2025 al 1/10/2025

Robert Laurentiu Mincu

Abstract

The `univpmthesis` class produces a template for the Bachelor and Master thesis manuscripts for students at UNIVPM. The class allows to write your manuscript using either italian or english; so, if you are an international student, feel free to use it!

Sommario

Indice

1. Hardware	1
1.1. STM32 NUCLEO-H745ZI-Q. La scheda di controllo	1
1.1.1. STM32H745ZI-TQ6. Architettura	1
1.1.2. STM32H745ZI-TQ6. Memoria	1
1.1.3. STM32H745ZI-TQ6. Convertitori	3
1.1.4. STM32H745ZI-TQ6. Timer	3
1.1.5. STM32H745ZI-TQ6. Periferiche di comunicazione	3
1.1.6. STM32H745ZI-TQ6. Eventi asincroni	4
1.1.7. NUCLEO-H745ZI-Q. Caratteristiche della scheda	4
1.1.8. NUCLEO-H745ZI-Q. Configurazione ed impiego delle periferiche	5
1.2. TATTU LiPo. Batteria ricaricabile	6
1.2.1. TATTU LiPo. Caratteristiche fisiche.	6
1.2.2. TATTU LiPo. Caratteristiche tecniche e considerazione operative.	6
1.3. PowerSafe Twin ADV. MNR-electronics	9
1.3.1. PowerSafe Twin ADV. Caratteristiche fisiche e tecniche . . .	10
1.4. DFRobot Power Module. Convertitore di tensione	11
1.5. Power Distribution Board	12
1.6. Turnigy AereoDrive SK3-3536 1400KV	12
1.6.1. Turnigy AereoDrive SK3-3536 1400 KV. Caratteristiche fisiche e tecniche:	13
1.7. Turnigy Plush 40A. <i>Electronic Speed Controller</i>	14
1.7.1. Turnigy Plush 40A. Caratteristiche fisiche:	15
1.7.2. Turnigy Plush 40A. Programmazione	15
1.7.3. Turnigy Plush 40A. Risoluzione e controllo dei motori	18
1.8. Eliche	20
1.8.1. GEMFAN GF 9060. Caratteristiche fisiche	21
1.9. Hitec HS-82MG Gear micro servo	21
1.9.1. HS-82 MG Hitec. Caratteristiche fisiche	22
1.9.2. HS-82 MG Hitec. Caratteristiche tecniche e controllo	24
1.10. BNO-055 Bosch Sensortec. Unità di misura inerziale e magnetometro	24
1.10.1. BNO-055 Bosch Sensortec. Caratteristiche fisiche	25
1.10.2. BNO-055 Bosch Sensortec. Infrastruttura di alimentazione, comunicazione ed integrazione del senore	25
1.10.3. Principio di misura dell'accelerometro, giroscopio e magnetometro	26

Indice

1.10.4. BNO-055 Bosch Sensortec. Accelerometro, giroscopio e magnetometro integrati	28
1.10.5. BNO-055 Bosch Sensortec. Modalità operative	30
1.11. VL53L1X ST. Sensore Time-of-Flight per la misurazione a lunga distanza	31
1.11.1. Il principio di misura di un sensore <i>Time-of-flight, ToF</i>	32
1.11.2. VL53L1X STMicroelectronics. Caratteristiche fisiche	32
1.11.3. VL53L1X STMicroelectronics. Catatteristiche tecniche	33
2. Firmware	37
2.1. Descrizione del firmware di gestione del VL53L1X STMicroelectronics. STSW-IMG007 FULL API	37
2.1.1. STSW-IMG007. <i>vl53l1_platform.c</i> , il <i>source file</i> di adattamento	38
2.1.2. STSW-IMG007. Procedure di adattamento	40
3. Sistema	48
3.1. Modello matematico	48
3.2. Controllo	48
4. Hardware	49
4.1. Componente1	49
4.2. Componente2	49
4.3. Schema dei collegamenti	49
5. Software	51
5.1. Diagramma di flusso	51
5.2. Gestione singoli componenti	51
5.2.1. Gestione componente1	53
5.2.2. Gestione componente2	53
5.3. Funzionamento complessivo	53
6. Test e risultati	54
6.1. Test1	54
6.2. Test2	55
Conclusioni e sviluppi futuri	56
Appendici	56
A. Appendix1	57
B. Il mio primo capitolo con L^AT_EX	58
B.1. Introduzione	58
B.2. Organizzazione dei files	58
B.2.1. Opzioni della classe	59

Indice

B.2.2. Pacchetti di supporto	59
B.3. Brevissimi esempi sull'inserimento degli oggetti di testo	59
B.3.1. Equazioni	59
B.3.2. Tabelle	60
B.3.3. Figure	60
B.4. Inserimento della bibliografia	60
B.5. Esempio di codice in python	61
B.6. Esempio di codice in C	61

Elenco delle figure

1.1.	Il microcontrollore STM32H745ZI-TQ6	4
1.2.	NUCLEO-H745ZI-Q. Dettaglio esplicativo dei <i>jumper</i>	6
1.3.	La scheda di sviluppo NUCLEO-H745ZI-Q	7
1.4.	TATTU LiPo 4S 1300mAh 75C	9
1.5.	PowerSafe Twin ADV MNR-electronics. Schema dei collegamenti . .	10
1.6.	PowerSafe Twin ADV MNR-electronics	11
1.7.	DFRobot Power Module. Disposizione dei terminali di uscita	11
1.8.	DFRobot Power Module	12
1.9.	Power Distribution Board	13
1.10.	Turnigy Aereo Drive SK3 3536 1400 KV	14
1.11.	Turnigy Plush 40A	16
1.12.	Schema dei collegamenti per la programmazione dell'E.S.C	16
1.13.	Turnigy Programming Card	19
1.14.	HS-82 MG Hitec. Misure	22
1.15.	HS-82 MG	23
1.16.	HS-82 MG Hitec. Disposizione dei terminali	24
1.17.	BNO-055 Bosch Sensortec. Configuraizone dei pin e breakout board	26
1.18.	BNO-055 Bosch Sensortec. Architettura di sistema	30
1.19.	BNO-055 Bosch Sensortec. Architettura di sistema	31
1.20.	P.C	33
1.21.	S.B.D	33
1.22.	VL53L1X, dettaglio sull'emettitore	34
1.23.	Field of View del VL53L1X	34
1.24.	VL53L1X, dettaglio sull'emettitore	36
4.1.	Schema dei collegamenti	50
5.1.	Diagramma di flusso del codice sviluppato	52
6.1.	Esempio di grafico	54

Elenco delle tabelle

1.1. VL53L1X. Modalità di misura	35
B.1. Esempio Tabella.	60

Capitolo 1.

Hardware

MANCA LA DESCRIZIONE INIZIALE DEL CAPITOLO

1.1. STM32 NUCLEO-H745ZI-Q. La scheda di controllo

L'STM32 NUCLEO-H745ZI-Q è una scheda di sviluppo prodotta da STMicroelectronics, basata sul microcontrollore STM32H745ZI-TQ6, appartenente alla famiglia ad alte prestazioni STM32H7. Il dispositivo è stato progettato per facilitare lo sviluppo, il *debug*, e la prototipazione di applicazioni *embedded* complesse.

La NUCLEO-H745ZI-Q è stata utilizzata per sviluppo e *testing* dei *firmware* di gestione di tutte le componenti del *D.P.D.F.* Su di essa sono state caricate tutte le librerie software sviluppate con l'ausilio di STMCubeIDE.

A seguire, una descrizione dettagliata delle *features* del microcontrollore STM32H745ZI-TQ6.

1.1.1. STM32H745ZI-TQ6. Architettura

Il microcontrollore STM32H745ZI-TQ6 presenta un'architettura *dual-core* a 32-bit composta da un processore ad alte prestazioni, il **Cortex-M7**, e da un processore il cui utilizzo è consigliato per applicazioni *real-time* e *low-power tasks*. L'architettura consente l'implementazione di applicazioni *multithread*.

In questo progetto di tesi, l'esecuzione delle librerie *software* di gestione dei dispositivi interessati è affidata al **Cortex-M4**.

A seguire un elenco delle caratteristiche concernenti all'architettura *dual-core*:

- Il **Cortex-M7** opera ad una frequenza di 480 MHz.
- Il **Cortex-M4** opera ad una frequenza di 240 MHz.

1.1.2. STM32H745ZI-TQ6. Memoria

Il microcontrollore possiede una struttura di memoria complessa e stratificata. Le memorie interne principali sono la memoria *Flash* e la *SRAM*. Inoltre, il *core M7* monta una *cache L1* e una piccola *backup SRAM* alimentabile a batteria.

Flash

La Flash interna da **2 MByte** è l'area di memoria su cui risiede il *firmware*. la memoria in questione è organizzata in *dual-bank*, che permette la programmazione/-cancellazione di una banca dati mentre si esegue sull'altra.

SRAM

Il dispositivo dispone di circa **1 MByte** di memoria dati volatile o SRAM, distribuita in più banchi con caratteristiche e destinazioni d'uso differenziate. In particolare:

- La SRAM1 e la SRAM2, rispettivamente da 320 KByte e 384 KByte, costituiscono i blocchi principali ad alta velocità, accessibili da entrambi i *core*. Rappresentano l'area privilegiata per l'allocazione dei dati che richiedono tempo di accesso ridotti e condivisibilità tra i due processori.
- Il blocco di SRAM3, da 128 KByte, è tipicamente dedicata al **Cortex-M4**, favorendo così una separazione logica delle risorse riducendo anche i conflitti di accesso.
- SRAM4, SRAM5 e SRAM6, in ordine, 128 KByte, 64 KByte e 64 KByte completano la struttura, fornendo ulteriori spazi di memoria che possono essere destinati a funzioni specifiche o a buffer ad alte prestazioni, secondo esigenze applicative.

Cache L1

Il **Cortex-M7**, a supporto dell'efficienza complessiva, integra una *cache L1* per istruzioni e dati, riducendo la letenza di accesso alla memoria non volatile e minimizzando i colli di bottiglia dovuti alla differenza tra la velocità del processore e quella della *Flash*. In tal modo le prestazioni globali del sistema risultano incrementate.

Flexible Memory Controller e Dual-mode Quad-SPI

Il *Flexible Memory Controller* è un'unità hardware che consente l'interfacciamento diretto con memorie parallele esterne al dispositivo. Il *FMC* supporta: SRAM, PSRAM, SDRAM, NOR Flash e NAND Flash. L'integrazione del *FMC* comporta un incremento nell'efficienza delle comunicazioni. Una volta configurato il *controller*, la memoria esterna entra a far parte dello spazio di indirizzi del microcontrollore mascherando la complessità del protocollo.

Il Dual Mode Quad-SPI è un'altra interfaccia dedicata alle *Flash* seriali esterne ad alta velocità. La soluzione poco fa citata permette di espandere la memoria dedicata al *firmware* o archiviare dati.

Memory Protection Unit

La *Memory Protection Unit* è una componente hardware integrata in ciascun *core* del microcontrollore. La mansione del circuito integrato in questione è quella di controllare e regolare l'accesso alla memoria.

Unità di elaborazione

L'architettura dell'STM32H745ZI-TQ6 integra anche unità di elaborazione dedicate ai calcoli numerici e al processamento dei segnali.

Floating Point Unit

La *FPU* è un'unità hardware integrata nei due *core* di elaborazione che consente l'esecuzione diretta di operazioni in virgola mobile. La presenza di questo circuito integrato riduce drasticamente la latenza di calcolo in applicazioni che richiedono elaborazioni numeriche complesse.

Nel caso del **Cortex-M7** la FPU supporta sia la *single precision* che la *double precision*, mentre il **Cortex-M4** solo una FPU a precisione singola.

Digital Signal Processing

Il *Digital Signal Processing* è un supporto hardware per istruzioni dedicate all'elaborazione numerica di segnali digitali.

1.1.3. STM32H745ZI-TQ6. Convertitori

Convertitori Analogici-Digitali

L'STM32H745ZI-TQ6 integra tre convertitori analogici digitali **SAR ADC** (*successive approximation analog-to-digital converter*) a 16-bit: ADC1, ADC2 e ADC3. I primi due, l'ADC1 e l'ADC2, sono "*tightly coupled*", cioè strettamente accoppiati, consentendo una sincronizzazione precisa delle conversioni. Questo approccio permette di ottenere alte frequenze di campionamento, precisione temporale e riduzione del carico computazione del core. L'ADC3 invece è istanziato separatamente.

La risoluzione di questi può essere configurata a 16, 14, 12, 10 o 8 bit.

Convertitori Digitali-Analogici

Il microcontrollore possiede 2 **DAC** indipendenti la cui risoluzione è a 12 bit.

1.1.4. STM32H745ZI-TQ6. Timer

I *timers* sono periferiche hardware fornite dal microcontrollore per svolgere attività correlate al tempo. L'STM32H745ZI-TQ6 fornisce 15 *timer* di diverso tipo e 5 *low power timers*. Alcuni di questi possono essere utilizzati per la genesi di segnali PWM.

1.1.5. STM32H745ZI-TQ6. Periferiche di comunicazione

L'STM32H745ZI-TQ6 integra un insieme ampio e variegato di protocolli di comunicazione. Le periferiche possono essere raggruppate in tre categorie principali: comunicazione seriale, comunicazione veloce e interfacce multimediali speciali. Per quanto riguarda la comunicazione seriale, il microcontrollore supporta i protocolli: I^2C ,

USART/UART e SPI. Nell’ambito della comunicazione veloce e avanzata il dispositivo supporta i protocolli: CAN (*Controller Area Network*), USB OTG, Ethernet MAC, SD/SDIO/MMC. L’integrazione delle interfacce multimediali consentono la gestione di applicazioni audio, grafiche e visione artificiale: SAI (*Serial Audio Interface*), DFSDM (*Digital Filter for Sigma-Delta Modulators*) e DCMI (*Digital Camera Interface*).

1.1.6. STM32H745ZI-TQ6. Eventi asincroni

Il microcontrollore offre la possibilità di gestire con efficacia ogni genere di *interrupt* (evento asincrono) grazie all'integrazione di un'unità di elaborazione dedicata: il NVIC (*Nested Vector Interrupt Controller*).



Figure 1.1.: Il microcontrollore STM32H745ZI-TQ6

Nel contesto applicativo sono state utilizzate molte delle periferiche poc'anzi citate. Nella successiva sessione si riserverà spazio per approfondire quali di queste e le modalità di utilizzo.

1.1.7. NUCLEO-H745ZI-Q. Caratteristiche della scheda

La NUCLEO-H745ZI-Q è la scheda di sviluppo (*development board*) che proietta all'esterno tutte le funzionalità del microcontrollore poco fa descritto.

Caratteristiche fisiche

- Dimensioni: $133.34 \times 70,00$ [mm].
 - Peso: 120 [g].

ST-LINK/V3E

L'ST-LINK/V3E è un'interfaccia hardware, gestita dal microcontrollore STM32F723, dedicata al collegamento tra elaboratore esterno e microcontrollore. Il modulo permette

Capitolo 1. Hardware

di caricare il compilato *firmware* sul microcontrollore direttamente dalla porta USB Micro-B collegata al sistema di elaborazione. Il modulo, permette, inoltre, di eseguire il *debug* in tempo reale del codice caricato sull'*MCU*.

L'ST-LINK/V3E fornisce anche interfacce ausiliarie utili ai fini dello sviluppo.

A seguire un breve elenco esplicativo di queste funzionalità aggiuntive:

- VCP (*Virtual COM Port*): lo ST-LINK crea una porta seriale virtuale via USB, in tal modo è possibile collegarsi al microcontrollore come se fosse collegato via UART classica.
- MSC (*Mass Storage*): la scheda di sviluppo appare al sistema di elaborazione come una memoria esterna.

Alimentazione

La scheda di sviluppo offre diversi approcci per l'alimentazione:

- Alimentazione via USB Micro-B ST-LINK a 5V.
- Alimentazione via Jack esterno da 7-12V.
- Alimentazione tramite *pin Vin* esterno a 3.3V.

Nonostante la presenza di diversi approcci di alimentazione e i differenti voltaggi, la tensione di alimentazione del microcontrollore è 3.3V, ottenuta attraverso regolatori interni.

Sulla scheda di sviluppo sono presenti dei *jumper*, ovvero dei piccoli interruttori meccanici che permettono di riconfigurare l'*hardware* della scheda evitando di intaccare il *firmware*. Si suggerisce di prestare attenzione alla configurazione dei *jumper* qualora si volesse alimentare la scheda da ingressi diversi dal **COM1** o **MICRO-B ST-LINK**.

Formato fisico

La scheda di sviluppo in esame appartiene alla famiglia **Nucleo-144** di STMicroelectronics e, dunque, mette a disposizione 2 connettori **Morpho** che espongono i 144 *pin* del microcontrollore. Il sistema di espansione poco fa citato viene chiamato ST Zio.

La NUCLEO-H745ZI-Q offre, inoltre, la compatibilità hardware e pinout con gli *Shield Arduino Uno R3*.

1.1.8. NUCLEO-H745ZI-Q. Configurazione ed impiego delle periferiche

Da inserire una volta sicuri ottenuta la sicurezza delle periferiche hardware.

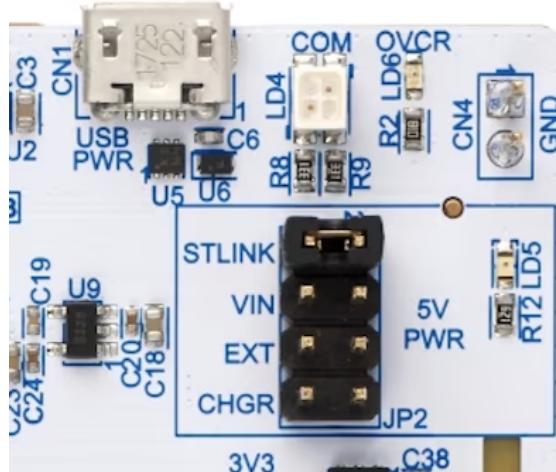


Figure 1.2.: NUCLEO-H745ZI-Q. Dettaglio esplicativo dei jumper

1.2. TATTU LiPo. Batteria ricaricabile

Motori, Servomotori e E.S.C richiedono specifici livelli di tensione per il loro funzionamento, tensioni che non possono essere fornite dalla NUCLEO H745ZI-Q. Si rende, quindi, necessario l'inserimento nel progetto di batterie ricaricabili.

La selezione delle batterie disponibili sul mercato è stata condotta a seguito di un'analisi approfondita delle caratteristiche elettriche di motori, servomotori ed E.S.C, unitamente ad uno studio dettagliato delle esigenze di autonomia necessarie per le prove di volo, rapportando il tutto al *budget* disponibile.

Le **TATTU LiPo** possiedono le prestazioni richieste dal progetto.

1.2.1. TATTU LiPo. Caratteristiche fisiche.

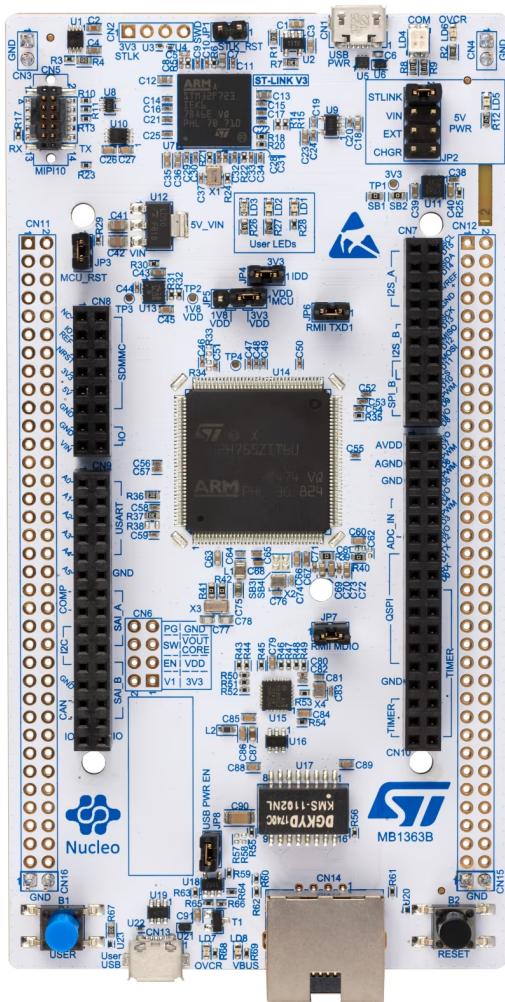
- Dimensioni: $74 \times 35.5 \times 29$ [mm].
 - Peso: 155 [g].
 - Numero di celle: 4.

1.2.2. TATTU LiPo. Caratteristiche tecniche e considerazione operative.

LiPo

La sigla *LiPo*, sta ad indicare la tecnologia elettrochimica utilizzata nella costruzione della batteria. *LiPo* nasce da *Lithium-Polymer*, ovvero polimero di litio. La particolare nominazione deriva dal componente chimico utilizzato come elttrolita, che, nel caso in esame, non è liquido, ma un polimero gelificato. Malgrado nelle fonti ufficiali non venga specificato il polimero usato nel progetto delle *TATTU LiPo*, con il fine di informazione, è stato deciso di riportare quello più comunemente utilizzato per la tipologia di batterie in esame, il *polivinilidenuoro esafluoropropilene* (PVDF-HFP).

Capitolo 1. Hardware



Rispetto ad altri sistemi elettrochimici presenti sul mercato (come ad esempio: Li-ion, *LiFePo₄*, NiMH ecc..), le batterie LiPo presentano un'elevata densità di potenza, una bassa resistenza interna e una notevole flessibilità nelle geometrie costruttive, oltre ad un contenuto peso. Tali attributi risultano vantaggiosi nell'ambito della progettazione di UAV, come il *D.P.D.F.*

Tensione

La *TATTU LiPo* possiede **4** celle da **3.7 [V]**. Le celle sono collegate in serie, perciò la tensione nominale è circa **14.8 [V]**.

A carica completa, ogni cella può raggiungere i **4.2 [V]**, emettendo un'uscita di **16.8 [V]**. Il mantenimento della tensione delle celle nell'intorno dei 3.7 [V], contribuisce a preservare la stabilità chimica interna delle batterie, riducendone l'erosione delle celle e prolungandone la vita utile. Si consiglia di non mantenere le celle a 4.2 [V] per periodi prolungati di tempo, evitandone così il danneggiamento.

Nella fase iniziale e di sviluppo *firmware*, le batterie sono state tenute in bilanciamento alla tensione nominale di 14.8 [V], mentre, nella fase finale, in cui i *test* di volo sono stati protagonisti, al fine di ottenere un'aumento dell'autonomia delle batterie e della prestazione dei motori, queste venivano caricate, mantenendo il bilanciamento tra celle, fino a 4.2 [V] per cella. Infine, è necessario adottare un'ulteriore accortezza per evitare danni. È consigliabile non far scendere la tensione della batteria al di sotto dei **12 [V]**, mantendo così il valore di tensione di ogni cella sopra i **3 [V]**.

Capacità

La *capacità* di una batteria, misurata in [mAh], *milliAmpere-ora*, definisce la quantità di carica elettrica che la batteira può immagazzinare e fornire nel tempo. Nel caso in esame, la capacità è di **1300 [mAh]**, vale a dire che la batteria può fornire **1.3 [A]** per un'ora. In generale si può utilizzare la seguente formula approssimata per il calcolo del tempo di funzionamento:

$$t_h = \frac{C \text{ [mAh]}}{A \text{ [mA]}} \quad (1.1)$$

dove t_h rappresenta il tempo di funzionamento in ore, C la capacità della batteria espressa in milliAmpere-ora e A la corrente assorbita espressa in milliAmpere. Con il fine di aumentare l'autonomia di volo, si è optato per l'accoppiamento in parallelo di **due TATTU LiPo**, così da ottenere una capacità totale di **2600 [mAh]**. L'accoppiamento in sicurezza è stato effettuato utilizzando il ***PowerSafe Twin della MNR-electronics***, il quale verrà approfondito nella successiva sezione.

Capacità di scarica

Con il termine *capacità di scarica* ci si riferisce alla quantità di corrente massima che la batteria può fornire, in sicurezza e in maniera continuativa, senza danneggiarsi. La *TATTU LiPo* possiede una capacità di scarica pari a **75C**, vale a dire che può

erogare:

$$A_{max} = C_s \cdot C \implies A_{max} = 75 [C] \cdot 1.3 [A] = 97.5 [A] \quad (1.2)$$

senza surriscaldarsi o danneggiarsi.

Nella precedente formula, A_{max} si riferisce alla corrente massima erogabile dalla batteria in sicurezza, espressa in [A], Ampère. C_s e C , invece, si riferiscono rispettivamente alla capacità di scarica e alla capacità di erogazione di corrente in'ora, espressa sempre in [A].



Figure 1.4.: TATTU LiPo 4S 1300mAh 75C

1.3. PowerSafe Twin ADV. MNR-electronics

Per garantire la sicurezza e l'affidabilità dell'alimentazione a due pacchi batteria, si è deciso per l'implementazione del **modulo di ridondanza, PowerSafe Twin ADV di MNR-electronics**.

Il dispositivo incarna un'avanzata soluzione di *ridondanza attiva*. Quest'ultima fa riferimento alla capacità del sistema di intervenire attivamente nel funzionamento del circuito con il fine di garantire continuità di alimentazione. Il modulo, di fatti, monitora continuamente alcune variabili di stato, come tensione e corrente, selezionando, in tempo reale, il pacco alimentante il carico. Per esempio, se dovesse abbassarsi la tensione di una batteria sotto una certa soglia critica, il dispositivo commuterebbe automaticamente sull'altra, evitando così, una significativa interruzione della potenza fornita.

Nell'architettura del sistema è possibile distinguere tre moduli distinti.

Modulo di potenza

Il modulo di potenza è il circuito elettrico responsabile: della commutazione tra i pacchi batteria, del bilanciamento e della distribuzione di corrente verso il carico.

Inoltre, il dispositivo implementa circuiti di protezione, isolando la sorgente guasta, qualora si dovesse presentare un malfunzionamento, ed evitando il trasferimento di corrente fra i pacchi batteria.

Modulo di controllo

Il modulo di controllo rappresenta il "centro di comando" del dispositivo. L'implementazione di un microprocessore permette la continua misura di parametri decisionali fondamentali, quali: la tensione delle batterie, la corrente di carico e l'eventuale squilibrio. L'inserimento del modulo di controllo nell'architettura circuitale, permette l'implementazione della, precedentemente citata, tecnologia di *ridondanza attiva*.

Modulo di allarme

Il modulo di allarme provvede alla segnalazione di eventuali guasti mediante emissioni luminose (LED) ed acustiche (Buzzer).

Schema di collegamento e montaggio

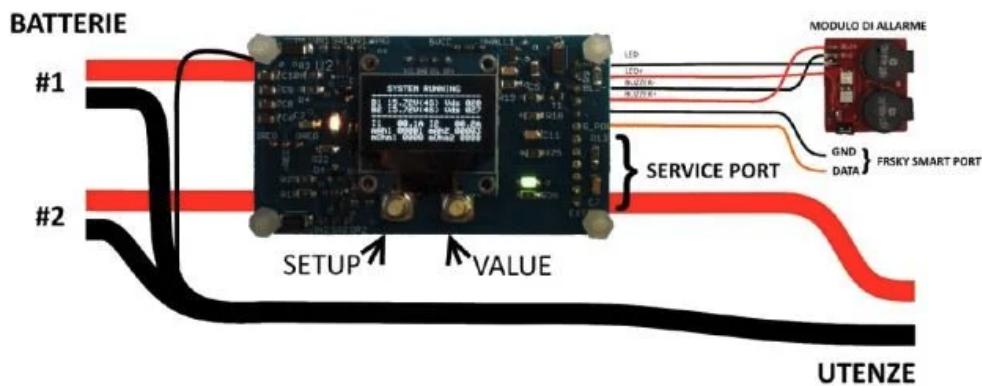


Figure 1.5.: PowerSafe Twin ADV MNR-electronics. Schema dei collegamenti

1.3.1. PowerSafe Twin ADV. Caratteristiche fisiche e tecniche

A seguire, un breve elenco delle caratteristiche fisiche e tecniche del dispositivo:

- Dimensioni con involucro protettivo: $80 \times 59 \times 26$ [mm].
- Peso, considerando sia l'involucro protettivo che i cablaggi (connettori XT60): 100 [g].
- Corrente massima gestibile: 100 [A] (in continua).
- Numero di celle per pacco batteria gestibili: $[3S - 8S]$
- Tensione massima ammessa in ingresso: ≈ 33.6 [V]. Considerando il minimo 3.7 [V] e il massimo 4.2 [V] per cella: $[11,1 \div 33,6]$ [V].



Figure 1.6.: PowerSafe Twin ADV MNR-electronics

1.4. DFRobot Power Module. Convertitore di tensione

Ai fini progettuali sono stati implementati diversi dispositivi, ciascuno dei quali richiedente di una specifica tensione di alimentazione. Si è reso necessario ricorrere a convertitori di tensione per distribuire nella maniera corretta le tensioni di alimentazione ai vari dispositivi.

Il **DFRobot Power Module**, riceve in ingresso la tensione proveniente dal pacco batterie, erogando quattro diversi valori di tensione a seconda del terminale che si sceglie di utilizzare.

La prima soluzione offerta dal dispositivo, fornisce in uscita la stessa tensione che si ha in ingresso, diventando così, un ponte per il passaggio della tensione.

Un'altra soluzione consiste nell'utilizzo degli appositi *pin* sul dispositivo, dai quali può essere erogata una tensione di **5 [V]**, selezionabile mediante un apposito *switch*. Infine, il dispositivo presenta un terminale che eroga una tensione variabile il cui valore è controllato da un *dimmer*. Il *dimmer* permette di selezionare valori intermedi fino ad un massimo pari alla tensione di ingresso. Quest'ultima è stata utilizzata nell'alimentazione dei servomotori. La presenza di due servomotori ha richiesto l'integrazione di due convertitori di tensione.

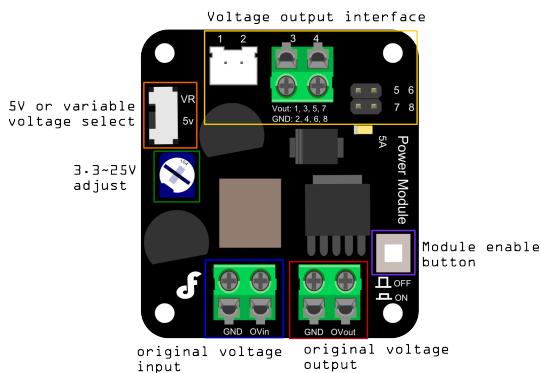


Figure 1.7.: DFRobot Power Module. Disposizione dei terminali di uscita

Caratteristiche fisiche

- Dimensioni: $46 \times 50 \times 20$ [mm].
- Peso: 35 [g].

Caratteristiche tecniche

- Intervallo di tensione in ingresso: **[3.6÷25]** [V].
- Intervallo di tensione esprimibile in uscita: **[3.3÷25]** [V].
- Massima potenza erogabile: **25** [W].
- Frequenza di variazione: **350** [kHz].



Figure 1.8.: DFRobot Power Module

1.5. Power Distribution Board

Con *Power Distribution Board* si fa riferimento ad un sottosistema elettrico la cui principale funzione è la **distribuzione in modo sicuro ed efficiente dell'energia fornita dal pacco batterie** agli *E.S.C* e, conseguentemente, ai motori. Ai fini progettuali, è stata selezionata una *P.D.B*, il cui punto di prelievo del segnale presenta un connettore **XT60**, adatto al pacco batterie.

Il *P.B.C* integrato nel contesto operativo, possiede quattro uscite eroganti al massimo una corrente di **20** [A] ed un peso di: **27.3** [g].

1.6. Turnigy AeroDrive SK3-3536 1400KV

Il Turnigy AeroDrive SK3 3536 1400kv è un motore *Brushless* alimentato a corrente continua e pilotato a trifase con correnti alternate.

Il rotore contiene magneti permanenti disposti con polarità alternata lungo la sua circonferenza. Lo statore invece, ospita tre avvolgimenti elettrici (fasi) disposti

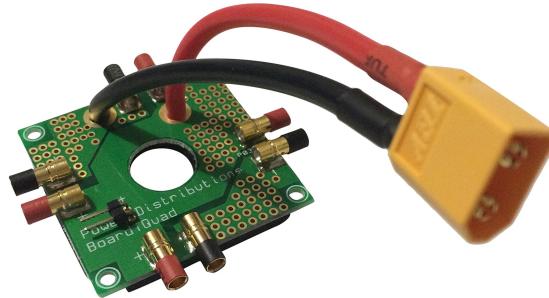


Figure 1.9.: Power Distribution Board

generalmente a 120 gradi elettrici l’uno rispetto l’altro, connessi ai tre terminali d’uscita che si identificano come A,B e C.

Il rotore, essendo formato da magneti permanenti, non necessita di alimentazione: è lo statore a generare il campo magnetico rotante che induce il moto.

Il cambio di polarità degli avvolgimenti deve essere gestito da un sistema elettronico esterno, *l’Electronic Speed Controller (ESC)*, che nella successiva sezione verrà approfondito.

I tre conduttori che fuoriescono dal motore rappresentano i terminali delle tre fasi dello statore. Ciascun filo è collegato a uno degli avvolgimenti, e nessuno di essi costituisce un riferimento comune. Per far ruotare il motore è necessario che tali fasi vengano alimentate secondo una precisa sequenza temporale, generando un campo magnetico rotante. Se si dovesse scambiare due dei tre cavi, l’ordine di commutazione si invertirebbe con conseguente variazione del senso di rotazione.

Nel progetto sono stati impiegati due motori del tipo precedentemente citato, disposti in configurazione **controrotante**, con il fine di compensare le rotazioni indesiderate di **yaw**. Tale scelta consente di bilanciare le coppie di reazione generate dalle eliche, riducendo il momento torcente sul velivolo. Inoltre, l’adozione di due motori si è resa necessaria in quanto un singolo attuatore non sarebbe stato in grado di generare una spinta sufficiente a garantire la portanza richiesta dal sistema nelle condizioni operative previste.

1.6.1. Turnigy AeroDrive SK3-3536 1400 KV. Caratteristiche fisiche e tecniche:

- Tensione operativa: [11.1÷16.8] [V].
- Valore KV: **1400** ($\frac{RPM}{V}$).
- Potenza massima: **590** [W].
- Corrente massima: **40** [A].
- Resistenza interna: [0.021÷0.025] [Ohm].

- Corrente a vuoto: **0.021 [A]**.
- Diametro dell'albero motore: **5 [mm]**.
- Peso: **111 [g]**.
- Spaziatura fori di montaggio: **25×25 [mm]**.
- Connettori *bullet* da **3.5 [mm]**.
- Numero dei poli: **12**.

Il valore "KV" è comune nel contesto del modellismo ed indica il numero di giri al minuto (RPM) che il motore compie per ogni *volt* applicato a vuoto, in formula:

$$RPM = KV \cdot V \quad (1.3)$$

A tensioni di alimentazione maggiori rispetto a quelle specificate il motore rischia di assorbire troppa corrente e surriscaldarsi.

I giri al minuto massimi dipenderanno dalla tensione applicata e dalle dimensioni delle eliche.

Nel presente progetto, il motore in interesse deve essere alimentato a batteria, vale a dire in corrente continua, lasciando spazio alla necessità di un dispositivo che trasformi la tensione continua erogata dalle batterie in un segnale trifase a corrente alternata. Il dispositivo in questione è il precedentemente citato *Electronic Speed Controller*.



Figure 1.10.: Turnigy Aereo Drive SK3 3536 1400 KV

1.7. Turnigy Plush 40A. *Electronic Speed Controller*

Il Turnigy Plush 40A è un controllore elettronico di velocità per motori *brushless* a corrente continua (BLDC).

Trasforma la tensione continua in ingresso in tensioni alternate sfasate di 120 gradi

elettrici.

L'applicazione della corretta sequenza di correnti alle fasi dello statore richiede la conoscenza della posizione angolare in ogni istante di tempo. Il Turnigy Plush 40A stima tale informazione in maniera indiretta, misurando la forza controelettromotrice indotta dal rotore, evitando così l'uso di sensori di posizione; il dispositivo in interesse può dunque essere definito *sensorless*. L'E.S.C in questione può essere concettualmente suddiviso in due sezioni principali: la sezione di potenza e la sezione di controllo. La **sezione di potenza** è responsabile della conversione *continua-trifase*. Tale conversione è mediata da un ponte trifase a sei MOSFET, suddiviso a sua volta in tre mezzi ponti, ciascuno associato ad una fase del motore. Ogni mezzo ponte è composto da un tranzistor di alto lato e da un transistor di basso lato, collegati rispettivamente al potenziale positivo della batteria e a massa. L'opportuna commutazione di apertura e chiusura dei sei MOSFET comporta la generazione di un segnale di tensione alternata per ogni fase.

La **sezione di controllo** ha come protagonista un microcontrollore che si occupa della generazione di segnali di pilotaggio e della stima della posizione del rotore. Il microcontrollore riceve in ingresso il comando di velocità di rotazione del motore sotto forma di segnale PWM. Sulla base di tale comando, il microcontrollore determina il *duty cycle* del PWM e calcola la sequenza di commutazione dei sei MOSFET, con conseguente generazione del campo magnetico rotante dello statore.

Il *Double Propeller Ducted Fan* monta due Turnigy Plush 40A, uno per ogni motore.

1.7.1. Turnigy Plush 40A. Caratteristiche fisiche:

- Corrente nominale: 40 [A].
- Corrente di picco: 55 [A].
- Presenza di un *Battery Eliminator Circuit* (BEC): 5 [V] a 3 [A].
- Celle di batteria necessarie per il corretto funzionamento del dispositivo: [2 ÷ 6].
- Peso: 39 [g].
- Dimensioni in [mm]: $60 \times 24 \times 15$.

1.7.2. Turnigy Plush 40A. Programmazione

Il Turnigy Plush 40A offre la possibilità di programmare il suo comportamento operativo, soddisfando così un vasto spettro di possibili esigenze progettuali.

Al fine di ottenere uno specifico comportamento dei motori, tale da adeguarsi alle particolari esigenze di un UAV come il *Double Propeller Ducted Fan*, entrambi gli E.S.C sono stati minuziosamente riprogrammati.

L'accesso alla *Programming Mode* e l'effettiva programmazione sono state effettuate



Figure 1.11.: Turnigy Plush 40A

mediante l'apposita *Turnigy Programming card*, acquistata separatamente. Per effettuare la programmazione, è necessario disporre i collegamenti elettrici come in figura:



Figure 1.12.: Schema dei collegamenti per la programmazione dell'E.S.C

La *Turnigy Programming Card* richiede di essere alimentata da un segnale continuo di tensione appartenente al seguente intervallo: $[4.8 \div 6][V]$. Per la navigazione fra i parametri è necessario utilizzare il pulsante *up/down*, mentre, per la selezione dei corrispondenti valori, deve essere utilizzato il pulsante *left/right*.

Nel momento in cui verrà selezionato il valore desiderato del parametro in programmazione, il LED blu, che risponde al nome di *connecting*, lampeggerà, segnalando così il successo dell'operazione. A seguire, una piccola spiegazione del significato dei parametri e i valori attualmente selezionati:

Brake

Il parametro *Brake* si riferisce alla modalità con cui viene gestita la mancanza di segnale PWM di controllo. Nel caso in cui dovesse essere impostato ad "ON" il motore verrà immediatamente frenato.

Il parametro in analisi non ha grande valenza ai fini progettuali dal momento che qualsiasi prova di volo è stata effettuata nell'apposita gabbia, per cui qualunque

combinazione del parametro è accettabile. Nello svolgersi del progetto il parametro è stato impostato ad "OFF".

Battery Type

Il *Battery Type* riferisce al dispositivo la tipologia di batteria in utilizzo, questo perché L'E.S.C monitora costantemente la tensione della batteria, quando questa scende sotto una certa soglia, entra in azione il *Low Voltage Cut-off*, riducendo o interrompendo la potenza fornita al motore con il fine di evitare il danneggiamento delle celle. Diverse tipologie di batterie hanno diverse curve di scarica. L'E.S.C può gestire solamente due tipologie di batterie, LiPo e NiHM. Il progetto prevede l'utilizzo delle **Tattu LiPo** perciò il parametro è stato impostato ad Li-xx.

Low Voltage Protection Mode (Cut-off Type)

Come già riferito in precedenza, la *Low Voltage Cut Off* evita il danneggiamento delle celle delle batterie quando queste scendono sotto una certa soglia impostabile. Il dispositivo permette di scegliere tra *Soft Cut-Off* e *Cut-Off*. Se si dovesse scegliere la modalità *Soft*, al superamento della soglia, l'E.S.C ridurrà gradualmente la potenza in uscita. Se si dovesse, invece, scegliere l'altra modalità, il dispositivo interromperà immediatamente la linea di fornitura di potenza al motore.

Al fine di evitare il danneggiamento del sistema, dovuto a un'interruzione improvvisa della fornitura di potenza ai motori durante il volo, seppur all'interno della gabbia, è stata impostata la modalità *Soft Cut-Off*.

Cut Off Voltage

Il parametro permette di impostare la soglia al di sotto della quale entra in azione la *Low Voltage Protection Mode*.

- *Low*: 2.6 [V].
- *Medium*: 2.85 [V].
- *High*: 3.1 [V].

Con il fine di limitare il danneggiamento delle batterie, ottenendo contemporaneamente una discreta autonomia di volo, il parametro è stato impostato a **Medium**.

Numericamente, considerando le batterie **Tattu LiPo** a 4 celle, la soglia di attivazione del L.V.C è: $2.85V \cdot 4 = 11.4V$.

Start Mode

la *Start Mode* gestisce la modalità di accelerazione del motore da fermo. Sono disponibili tre diverse modalità di partenza, pensate per adattarsi a qualsiasi applicazione. La *Normal Start Mode* costituisce la modalità di accelerazione più reattiva. L'E.S.C applica una rampa di potenza relativamente breve, nel giro di $[0.3 \div 0.5][s]$ il motore raggiunge i primi giri utili. L'appena descritta modalità è stata scartata a priori

Capitolo 1. Hardware

dal momento che, come conseguenza di un'accelerazione repentina, si ha forte stress meccanico sull'albero motore, inoltre l'improvvisa accelerazione delle eliche può generare forze indesiderate.

Nella *Soft Start* la rampa di ingresso è più dolce, di fatti il motore, sotto il controllo dell'E.S.C, impiega $[1 \div 1.5][s]$ per arrivare a regime. Infine, nella *Very Soft Start* la potenza viene incrementata nell'arco di $[2 \div 3][s]$.

Qualsiasi prova di volo del *D.P.D.F* è stata eseguita nell'apposita gabbia di contenimento, in cui il sistema è stato sospeso mediante funi di sostegno, perciò, con il fine di evitare movimenti indesiderati, stress meccanico ed errori di misurazione è stata impostata la *Very Soft Start*.

Timing Mode

L'E.S.C, con il fine di genesi di rotazione, genera campi magnetici rotanti commutando le tre fasi in sequenza. Le commutazioni devono avvenire in sincronia con la posizione del rotore. Il *Timing* è l'anticipo angolare con cui l'E.S.C commuta le fasi rispetto alla posizione stimata del rotore.

Se il *Timing* dovesse essere basso, il campo magnetico rotante sarebbe quasi allineato con il rotore, disposizione che aumenterebbe l'**efficienza ellettrico-meccanica** complessiva del sistema ma diminuirebbe coppia e velocità. Al contrario, se dovesse essere alto, si avrebbe più potenza e velocità ma anche più dispersione di calore e meno efficienza.

L'*High Timing Mode*, a cui corrisponde un anticipo angolare di $[16 \div 30]^\circ$, favorisce potenza e velocità, utile per dispositivi ad alto numero di giri o da corsa. La *Low Timing Mode*, con un'anticipo angolare di $[0 \div 7]^\circ$, garantisce una coppia stabile con minima dispersione di calore con una leggera diminuzione della velocità massima.

La *Medium Timing Mode* ($[8 \div 15]^\circ$) costituisce un buon compromesso fra efficienza e prestazioni, dunque è stata scelta ai fini progettuali.

Music Li-Po Cells

La funzionalità è puramente estetica e non influisce sulle prestazioni, per cui non è stata impostata alcuna musica.

Governor Mode

La *Governor Mode* ha lo scopo di mantenere la velocità di rotazione del motore costante indipendentemente dalle variazioni di carico o tensione della batteria. Non essendo utile ai fini progettuali, il parametro è stato impostato su "OFF".

1.7.3. Turnigy Plush 40A. Risoluzione e controllo dei motori

Il *Turnigy Plush 40A*, al fine di controllare la velocità di rotazione dei motori, richiede un segnale del tipo PWM alla frequenza tipica del modellismo, vale a dire, $50[Hz]$.

Capitolo 1. Hardware



Figure 1.13.: Turnigy Programming Card

L'immagine riportata sopra non è riferita alla configurazione attuale degli ESC, ma ha esclusivamente scopo illustrativo.

In primo luogo, è necessaria una procedura di armamento dei motori, che consiste nell'invio del segnale PWM con *duty cycle* del 4.75%. L'esito dell'operazione di armamento può essere stabilito mediante delle segnalazioni acustiche emesse dall'E.S.C. A seguire, le emissioni acustiche più comuni:

Attesa di un segnale di armamento valido

Un'emissione ogni due secondi, segnala l'attesa di un segnale PWM idoneo all'armamento iniziale dei motori. L'emissione viene attivata dal dispositivo immediatamente dopo la connessione di questo all'alimentazione.

Esito positivo dell'operazione di armamento

Tre emissioni sonore consecutive seguite da un'emissione prolungata, indicano l'esito positivo dell'operazione di armamento.

Esito negativo dell'operazione di armamento

Una rapida ripetizione di emissioni sonore (una ogni 0.25 [s]) segnala il mancato armamento dei motori, dovuto ad un segnale di ingresso errato. Comunemente l'esito negativo di armamento si può presentare nel caso in cui il segnale PWM dovesse avere un *duty cycle* superiore al 4.75% oppure nel caso in cui il *duty cycle* dovesse repentinamente variare dal 4.75% a valori superiori. Si consiglia di attendere [2 ÷ 6] [s] prima di iniziare la variazione del *duty cycle* del segnale per il controllo della velocità dei motori.

Al seguito di vari accertamenti empirici, è stato stabilito l'intervallo di lavoro dei motori espresso in *duty cycle*: [6% ÷ 12%].

Risoluzione

Quando si tratta di E.S.C, con il termine "risoluzione" non ci si riferisce al numero di campioni di tensione logica distinguibili dal dispositivo, bensì allo "*span*" di valori del *duty cycle* distinguibili.

La conoscenza della risoluzione del dispositivo è utile nella determinazione della variazione minima del *duty cycle* del segnale PWM di controllo, che a sua volta permette lo sviluppo di un algoritmo di controllo più efficace. Per quanto riguarda il *Turnigy Plush 40A*, la documentazione ufficiale non specifica la risoluzione del dispositivo; pertanto, si è fatto riferimento a fonti non ufficiali, che indicano una risoluzione compresa tra i 8 e 12 bit.

Si è optato per uno sviluppo del controllore capace di trascurare la risoluzione del dispositivo, a causa dell'ambiguità di tale informazione.

1.8. Eliche

Al fine di ottenere una corretta realizzazione progettuale, si rende necessario un minuzioso studio circa le eliche da utilizzare. Le eliche e i motori, costituiscono un perfetto connubio, di fatti, se si dovessero scegliere le eliche sbagliate, potrebbe non verificarsi la generazione della portanza necessaria per il sollevamento del *D.P.D.F.* Per la selezione, sono stati presi in considerazione i due parametri fondamentali usati nella caratterizzazione delle eliche, il **diametro** e il **passo**.

Con il primo si indica il diametro della circonferenza che l'elica produce quando è in rotazione. Con il secondo, invece, si fa riferimento alla *distanza teorica* che l'elica percorrerebbe lungo il suo asse di rotazione in una singola rotazione completa, assumendo che si muova in un fluido perfettamente solido e senza alcun slittamento. Per capire al meglio questo concetto, si suggerisce di pensare ad una vite che compie un giro completo mentre sta venendo inserita nel legno.

Il **diametro** di un'elica ne influenza:

- **Spinta.** Un aumento del diametro comporta un aumento della spinta a parità di RPM.
- **Efficienza.** Un aumento del diametro suggerisce un, seppur leggero, aumento dell'efficienza, dal momento che viene mosso un maggiore volume di aria con meno energia.
- **Coppia del motore richiesta.** Maggiore è il diametro dell'elica, maggiore è la coppia richiesta dai motori per muoverle.
- **Reattività.** Più è alto il valore del diametro e più è alta l'inerzia rotazionale, diminuendo la reattività del sistema.

Un elevato diametro è consigliato per sistemi che hanno l'obiettivo di stabilità. Per quanto riguarda il **passo**, il parametro ha prestigio su:

- **Velocità in volo orizzontale.** Un aumento del passo costituisce un aumento della velocità espressa orizzontalmente.
- **Spinta.** La spinta espressa dall'elica aumenta se ne si aumenta il passo, tuttavia il diametro dell'elica ne ha più influeza.
- **Coppia del motore richiesta.** Un maggior passo richiede una maggior coppia indotta dai motori, con conseguente aumento dello stress meccanico.
- **Bassa efficienza di Hovering.** Un aumento del passo comporta un aumento della velocità dell'aria mossa dall'elica, con conseguente diminuzione della stabilità.

Al contrario, un elevato passo è tipico dei sistemi focalizzati sulla velocità orizzontale e verticale.

1.8.1. GEMFAN GF 9060. Caratteristiche fisiche

La denominazione "9060" fa riferimento ai due parametri fondamentali dell'elica, il diametro e il passo, espressi in pollici

- Diametro: 9 pollici che sono all'incirca 228,9 [mm].
- Passo: 6 pollici, all'incirca 152,4 [mm].

1.9. Hitec HS-82MG Gear micro servo

Il disegno strutturale rende il *D.P.D.F* suscettibile a rotazioni indesiderate sul piano definito dagli assi X ed Y. Tali perturbazioni possono compromettere la stabilità del sistema, determinandone così, una completa perdita di controllo del mezzo.

Al fine di limitare tali perturbazioni, sono stati integrati due *flap*, uno relativo al controllo lungo l'asse delle ascisse e uno lungo l'asse delle ordinate.

I *flap* operano come superfici aerodinamiche di controllo dedicate alla compensazione dei distrubbi di *rollio* e *beccheggio*. Sono posizionati sulla parte terminale del condotto ed interagiscono con il getto generato dai motori controrotanti. L'intervento si basa sulla modifica della direzione e della velocità del flusso d'aria spinto verso il basso. Il *flap*, ruotando di un piccolo angolo, genera una deviazione del getto che lo investe. Tale deviazione produce una variazione della distribuzione di pressione tra il lato esposto al flusso (lato di pressione) e il lato opposto (lato di depressione). La differenza di pressione genera una forza aerodinamica risultante che produce un momento correttivo sull'assetto del velivolo. Nella prima approssimazione utile al controllo, la forza può essere considerata perpendicolare alla superficie del *flap*.

La direzione e il verso della forza devono essere tali da contrastare il momento perturbativo.

Per il controllo del movimento dei *flap*, sono stati selezionati sei servomotori sulla base di alcuni parametri fondamentali, tra cui, coppia di stasi e peso. Dopo un'attenta ricerca, è stato selezionato l'**HS-82MG Gear Micro Servo** della casa **Hitec**. Nel progetto, sono stati integrati due servomotori del tipo sopra citato, uno per ogni *flap*.

1.9.1. HS-82 MG Hitec. Caratteristiche fisiche

- Dimensioni: $29.8 \times 12.0 \times 29.6$ [mm].
- Peso: 19.0 [g].
-

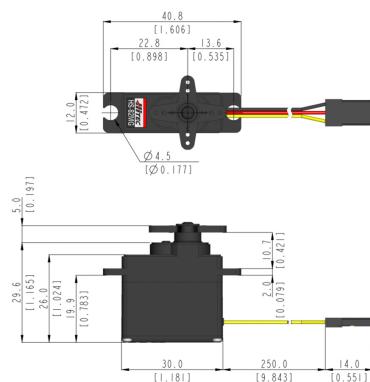


Figure 1.14.: HS-82 MG Hitec. Misure

Coppia di stasi

Con *coppia di stasi* si fa riferimento alla coppia massima che il servomotore può mantenere in posizione statica mentre oppone resistenza ad una forza esterna.

l'*HS-82* offre una coppia di stasi pari a: **2.2 [Kg·cm]** con **4.8 [V]** in ingresso e **2.7 [Kg·cm]** a **6.0 [V]**.

Coppia di stallo

La coppia di stallo è la massima coppia teorica che il servomotore può generare a velocità angolare nulla, vale a dire con carico irremovibile.

Per il dispositivo, la coppia di stallo assume i seguenti valori: **2.8 [Kg·cm]** a **4.8 [V]** e **3.4 [Kg·cm]** a **6.0 [V]**.

Velocità angolare

La velocità angolare del dispositivo senza carico è di **60° in 0.12 [s]** a **4.8 [V]**,

Capitolo 1. Hardware

mentre, è di **60° in 0.10 [s]** a **6.0 [V]**. Rapportando ad un’unico grado di variazione, si ha: **1° in 0.0020 [s]** a **4.8 [V]** e **1° in 0.0017 [s]** a **6.0 [V]**.

Corrente operativa

La corrente operativa è la corrente assorbita dal servomotore mentre questo è in movimento. Le informazioni qui sotto riportate si riferiscono al servomotore privo di carico.

Si ha un assorbimento di **220 [mA] per 60°** a **4.8 [V]** e un assorbimento di **280 [mA] per 60°** a **6.0 [V]**.

Corrente di stallo

Con *corrente di stallo* si fa riferimento alla massima corrente che il dispositivo assorbe quando è alimentato in condizione di blocco meccanico. Con questa configurazione, l’assorbimento di corrente ammonta a **1450 [mA]** a **4.8 [V]** e **1800 [mA]** a **6.0 [V]**.

Corsa operativa

La *corsa operativa* fa riferimento all’intervallo angolare che il servomotore copre in risposta ad un segnale PWM **standard**. Nel caso del *HS-82*, l’intervallo è pari a **120°**.



Figure 1.15.: HS-82 MG

1.9.2. HS-82 MG Hitec. Caratteristiche tecniche e controllo

Alimentazione

L'intervallo di tensione operativa è il seguente: **[4.8÷6.0] [V]**.

Per valori di tensione prossimi all'estremo superiore dell'intervallo si hanno dei miglioramenti sulle prestazioni generali del dispositivo. Tuttavia, aumentano anche gli effetti dei comportamenti indesiderati, come un aumento della dispersione di calore per effetto joule o dello stress sull'elettronica interna di controllo.

Per evitare complicazioni, il valore di tensione in ingresso è stato impostato a **5.5 [V]**.

Controllo

Il controllo della posizione angolare del rotore avviene tramite segnale PWM. Analogamente al *Turnigy Plush 40A*, si è optato per una frequenza di **50 [Hz]**, mentre l'ampiezza è pari a **3.3 [V]**. Con il fine di ottenere il posizionamento dei *flap* in configurazione perpendicolare al terreno, è necessario fornire un impulso di attivazione pari a **1.5 [ms]**. Con la frequenza scelta, tale valore corrisponde ad un **duty cycle del 7.5%**, che rappresenta la posizione centrale del servomotore.

La documentazione indica una *Dead Band Width* pari a **5 [μs]**, ovvero la minima variazione di impulso che il servomotore è in grado di rilevare. Da ciò si deduce che la minima variazione angolare attuabile col servomotore è di **0.5°**. Nel contesto applicativo ciò equivale ad una variazione minima del *duty cycle* pari a **0.025%**. Per ottenere una rotazione oraria del rotore, il comando deve operare nel seguente intervallo di impulsi: **1.5÷2.1 [ms]**. L'effetto di rotazione antioraria si ha per: **0.9÷1.5 [ms]**.

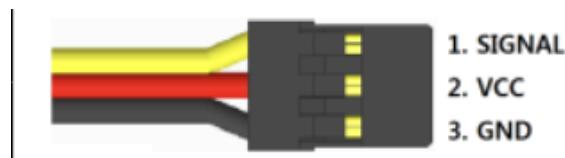


Figure 1.16.: HS-82 MG Hitec. Disposizione dei terminali

1.10. BNO-055 Bosch Sensortec. Unità di misura inerziale e magnetometro

Al fine di controllare l'orientamento nello spazio del *D.P.D.F* mediante servomotori, si necessita dell'informazione di rotazione del sistema nello spazio di riferimento. Si richiede, dunque, un dispositivo capace di ottenere tale informazione. Il **BNO-055** della casa **Bosch Sensortec**, con la sua accuratezza, è il miglior candidato per tale responsabilità. Il **BNO-055** è un sensore di orientamento assoluto a 9 assi sviluppato da **Bosch Sensortec**. Il dispositivo è un *System in Package[A]* che

integra un accelerometro triassiale a 14 bit, un giroscopio triassiale a 16 bit, un sensore geomagnetico triassiale e un microcontrollore Cortex $M0^+$ a 32 bit incaricato nell'eseguire il *software di sensor fusion* integrato.

Il *software*, poc'anzi citato, combina i dati provenienti da accelerometro, giroscopio e magnetometro, fornendo: **Quaternioni, Angoli di Eulero come pitch, roll e yaw e Vettori di orientamento lineari e gravitazionali**. Nel presente progetto, il *chip* BNO-055 è stato acquistato con montaggio su *scheda di breakout* incluso, dunque nel paragrafo sottostante sono state aggiunte sia le dimensioni/peso del modulo che le dimensioni/peso del chip.

1.10.1. BNO-055 Bosch Sensortec. Catatteristiche fisiche

- dimensioni del modulo: $20 \times 24 \times 2$ mm.
- dimensioni del chip: $3.8 \times 5.2 \times 1.1$ mm.
- massa del modulo: 3 g.
- massa del chip: $\approx 150mg$.

1.10.2. BNO-055 Bosch Sensortec. Infrastruttura di alimentazione, comunicazione ed integrazione del senore

Alimentazione

Il sensore ha due distinti ingressi di alimentazione: Il V_{DD} e il V_{DDIO} .

Il V_{DD} è il principale terminale di alimentazione dei sensori. Il valore di tensione iniettabile nell'ingresso in analisi appartiene al seguente intervallo: **[2.4÷3.6] [V]**. Il V_{DDIO} , invece, è il distinto ingresso di alimentazione del μC e delle interfacce digitali. In questo caso, il valore di tensione può assumere valori di questo intervallo: **[1.7÷3.6] [V]**.

Tuttavia, il dispositivo è montato su di una *breakout board* che offre a disposizione solamente un terminale di alimentazione, chiamato V_{IN} , il quale fornisce, mediante un micro regolatore di tensione, l'alimentazione sia al V_{DD} che al V_{DDIO} . Il valore di tensione in ingresso al terminale V_{IN} appartiene all'intervallo: **[2.4÷3.6] [V]**. Il BNO-055, inoltre, supporta tre differenti modalità di alimentazione: *Normal mode*, *Low power mode* and *Suspend mode*. Per lo sviluppo è stata scelta la **Normal mode**.

Protocollo di comunicazione

Il *BNO-055* offre la possibilità di utilizzare due distinti protocolli di comunicazione seriale, *I²C* e *UART*. Nel presente progetto è stato utilizzato esclusivamente il protocollo *I²C*. Come per il *VL53L1X* di *STMicroelectronics*, precedentemente trattato, il sensore dispone della capacità di comunicare in *Fast Mode*, scambiando byte a 400 [kHz] (oltre che in *Standard Mode* a 100 [kHz]). Anche in questo caso,

sia la *breakout board* che il sensore forniscono i terminali di **Serial DAta (SDA)** e **Serial CLock (SCL)**.

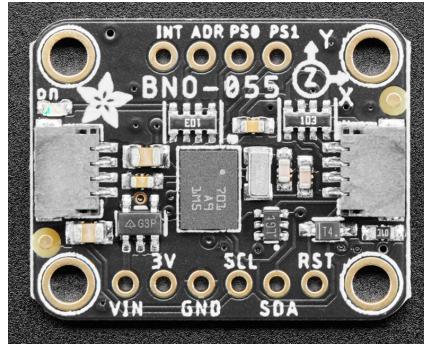


Figure 1.17.: BNO-055 Bosch Sensortec. Configuraione dei pin e breakout board

1.10.3. Principio di misura dell'accelerometro, giroscopio e magnetometro

Prima di descrivere le *features* dell'accelerometro, giroscopio e magnetometro installati nel *BNO-055*, si ritiene necessario dedicare una sezione di approfondimento circa i principi di funzionamento dei sensori precedentemente citati.

Accelerometro

L'accelerometro integrato nel BNO-055 Bosch Sensortec è di tipo capacitivo M.E.M.S, *Micro-Eletro-Mechanical System*. Il dispositivo consente la misura dell'accelerazione lungo i tre assi cartesiani.

Il principio di funzionamento si basa sulla rilevazione delle variazioni di capacità tra microstrutture mobili e fisse realizzate su un substrato di silicio.

L'elemento sensibile di ciascun asse è costituito da una massa sospesa, *proof mass*, vincolata da microtravi elastiche ad una cornice ancorata al substrato. In condizioni di quiete, la massa è in equilibrio e la capacità tra le piastre interdigitate[A] rimangono simmetriche.

Quando il dispositivo è sottoposto ad un'accelerazione lungo uno degli assi sensibili, la massa inerziale si sposta in direzione opposta a quella dell'accelerazione, generando una variazione differenziale delle capacità tra le piastre.

La variazione di capacità viene rilevata da un circuito integrato, che la converte in un segnale elettrico, proporzionale alla velocità applicata.

Il segnale analogico prodotto, viene successivamente digitalizzato da un convertitore analogico digitale integrato nel chip.

Giroscopio

Il giroscopio integrato nel dispositivo fa parte della categoria M.E.M.S di tipo vibrante,

vibrating structure gyroscope, e consente la misura della velocità angolare lungo i tre assi cartesiani.

Il principio di funzionamento si basa sull'effetto Coriolis, che si manifesta quando una massa in moto oscillatorio subisce una rotazione rispetto ad un sistema di riferimento inerziale. All'interno del sensore, ciascun asse dispone di una o più masse vibranti, le quali vengono mantenute in oscillazione a frequenza costante, mediante un circuito di attuazione elettrostatica. Quando il dispositivo ruota attorno ad uno degli assi la massa subisce una forza di Coriolis data da:

$$\vec{F}_c = 2m(\vec{v} \times \vec{\omega}) \quad (1.4)$$

dove m è la massa oscillante, \vec{v} è la velocità della massa nella sua traiettoria vibrante, $\vec{\omega}$ è la velocità angolare del corpo.

Questa forza induce uno spostamento trasversale rispetto alla direzione di vibrazione, che viene rilevato attraverso variazioni di capacità tra elettrodi fissi e mobili, similmente a quanto avviene nell'accelerometro.

Tali variazioni, proporzionali alla velocità angolare, vengono convertite in un segnale elettrico mediante un circuito di lettura differenziale e successivamente digitalizzate tramite un convertitore analogico digitale integrato.

Magnetometro

Il sensore integra inoltre un **magnetometro ad effetto Hall**, sfruttante il fenomeno fisico della tensione di Hall per misurare l'orientamento del dispositivo rispetto al campo magnetico terrestre.

Nella sua forma più semplice, un elemento di Hall è una sottilissima lamina di materiale conduttore o semiconduttore attraversata da una corrente continua controllata. Se su questa lamina agisce un campo magnetico con componente perpendicolare alla direzione della corrente, i portatori di carica vengono deviati lateralmente della **forza di Lorentz**:

$$F = q(\vec{v} \times \vec{B}) \quad (1.5)$$

con \vec{v} vettore velocità della carica e \vec{B} vettore campo magnetico. Questa deviazione procude un accumulo di carica ai bordi opposti della lamina, e, in regime stazionario, un campo elettrico trasversale che equilibria la forza magnetica. Il risultato è una differenza di potenziale trasversale, la **tensione di Hall**: V_H che risulta proporzionale alla componente di campo magnetico normale alla superficie del sensore e alla corrente che lo attraversa. Per un elmento *Hall* omogeneo abbiamo:

$$V_H = \frac{IB_{\perp}}{nqs} \quad (1.6)$$

dove B_{\perp} è la componente del campo perpendicolare al piano della lamina, n la densità di portatori, q il valore di carica elementare e s lo spessore del *film* attivo.

Nel magnetometro ogni elemento di Hall è disposto in modo da avere la propria normale allineata con uno degli assi cartesiani del sistema di riferimento del sensore. Quando il campo magnetico terrestre attraversa il dispositivo, ciascun elemento rileva la proiezione del vettore \vec{B} lungo il proprio asse, trasformandola in una tensione di Hall proporzionale.

Dopo un'opportuna amplificazione della tensione di Hall percepita e una conversione analogico-digitale si ottiene un tripletto di valori numerici che rappresentano le tre componenti cartesiane del campo magnetico locale.

Il vettore tridimensionale ottenuto viene ricostruito e poi confrontato con il valore atteso del campo geomagnetico.

Attraverso il *sensor fusion*, il magnetometro fornisce quindi il riferimento assoluto per l'*azimut*, vale a dire, la direzione del Nord magnetico rispetto al sistema di riferimento solidale del sensore.

1.10.4. BNO-055 Bosch Sensortec. Accelerometro, giroscopio e magnetometro integrati

L'accelerometro, il giroscopio e il magnetometro sono tutti prodotti da Bosch Sensortec.

Accelerometro

L'accelerometro installato sul *BNO-055* appartiene alla famiglia *Bosch Sensortec*. La sua **risoluzione** è di **14 bit**. L'accelerometro, per adattarsi ad una vasta gamma di esigenze progettuali, mette a disposizione diversi intervalli di misura:

- $[\pm 2]$ [g]. Modalità utile nella rilevazione di piccolissime accelerazioni.
- $[\pm 4]$ [g]. Rappresenta un compromesso equilibrato. Adatto alla maggior parte delle applicazioni, di fatti, quando attiva la *sensor fusion*, viene impostato tale intervallo di misura.
- $[\pm 8]$ [g] e $[\pm 16]$ [g]. Le modalità vengono utilizzate in applicazioni dinamiche estremamente veloci. Questi intervalli difficilmente vengono saturati. Nel presente progetto non sono stati nemmeno presi in considerazioni.

L'accelerometro, con attiva la *sensor fusion*, ha una banda passante di **62.5 Hz**.

Giroscopio

Come per l'accelerometro, anche il giroscopio appartiene alla casa *Bosch Sensortec*. La sua **risoluzione**, però, è di **16 bit**. Anche in questo caso si dispone di diversi intervalli di misura:

- $[\pm 125]^\circ/\text{s}$. Modalità pensata per movimenti lenti e precisi, di fatti, possono essere rilevate minime variazioni d'angolo.

- $[\pm 250]^\circ/\text{s}$. Viene conservata la precisione a fronte di un intervallo più permissivo rispetto al precedente.
- $[\pm 500]^\circ/\text{s}$. Questa modalità è adottata dalla maggioranza delle applicazioni.
- $[\pm 1000]^\circ/\text{s}$. La modalità è stata pensata per azionamenti energici.
- $[\pm 2000]^\circ/\text{s}$. Quest'ampio intervallo è adatto ad applicazioni estremamente veloci. I dati in questa modalità sono estremamente rumorosi. Nonostante questo, è la modalità operativa impostata, quando la *sensor fusion* è attiva. L'estremo rumore di questa modalità, viene attenuato dalla *sensor fusion*.

Ad un'aumento dell'ampiezza dell'intervallo, corrisponde una diminuzione della qualità del dato. Il giroscopio, nel mentre che la *sensor fusion* è attiva, ha una banda passante di **32 Hz**.

Magnetometro

Anche il magnetometro porta il marchio *Bosh Sensortec*. Il dispositivo misura campi nell'ordine dei **$\pm 1300 \mu\text{T}$** sugli assi X e Y, e **$\pm 2500 \mu\text{T}$** sull'asse Z. Seguendo il precedente ordinamento degli assi, la **risoluzione** è rispettivamente di **13 bit** e **15 bit**. La documentazione suggerisce un'elevata sensibilità ai disturbi (*soft iron, hard iron*). Difetto che può essere corretto attivando la *sensor fusion*.

La banda passante del dispositivo, quando è attivata la *sensor fusion* è di **10 Hz**.

Sensor fusion

Se presi singolarmente, nessuno dei sensori trattati fornisce misure accurate e complete. Il progetto necessita di un sensore che possa fornire la rotazione del sistema intorno agli assi. Questa rotazione è ottenuta dall'integrazione della grandezza estratta dal giroscopio, tuttavia, è intrinsecamente suscettibile ad errori (amplificazione di rumore o *bias*).

L'accelerometro fornisce un'informazione assoluta circa la direzione del vettore gravità, da cui possono essere ricavati gli angoli di *roll* e *pitch*. Tuttavia, il limite principale risiede nella presenza di accelerazioni non dovute alla gravità, di fatti, rapidi movimenti o vibrazioni producono rumore che interferiscono con la determinazione del vettore gravità.

Infine, il magnetometro è suscettibile alla presenza di elementi ferromagnetici o correnti elettriche, determinando, così, errori significativi nella misura.

La *sensor fusion* nasce con l'obiettivo di superare tali limiti, combinando coerentemente le informazioni disponibili. Nei sistemi più evoluti, utilizzati nelle piattaforme di navigazione di veicoli e velivoli, si impiegano filtri come *EKF* (Filtro di Kalman esteso) o *UKF* (Filtro di Kalman a Punti Sigma). La documentazione ufficiale del BNO-055 riporta che il microcontrollore presenta un algoritmo di *sensor fusion* utilizzate il **Filtro di Kalman esteso**.

Nel presente progetto, con il fine di aumentare l'attendibilità, l'accuratezza e la

stabilità della misura, la ***sensor fusion*** è stata sempre tenuta attiva.

A seguire una rappresentazione schematica del *System in package BNO-055 Bosch Sensortec*.

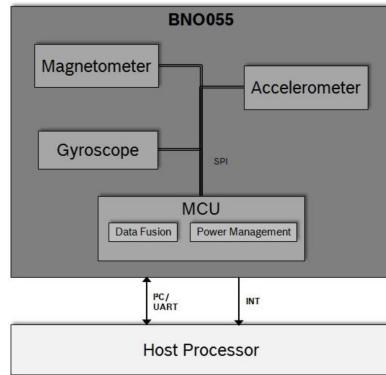


Figure 1.18.: BNO-055 Bosch Sensortec. Architettura di sistema

1.10.5. BNO-055 Bosch Sensortec. Modalità operative

Il dispositivo di misura fornisce una grande varietà di segnali di *output*, che possono essere scelti selezionando l'appropriata modalità operativa.

Le modalità operative vengono classificate in base all'attivazione o meno del *software di sensor fusion*, nello specifico distinguiamo tra le *Non-Fusion modes* e le *Fusion modes*.

Fusion-modes

Nel momento in cui viene impostata una delle seguenti modalità, l'algoritmo di fusione esegue automaticamente, in *background*, la calibrazione dei sensori.

- **IMU (Inertial Measurement Unit)**. In questa modalità di fusione, l'orientamento relativo del *BNO-055* nello spazio, è determinato utilizzando i dati estratti dall'accelerometro e dal giroscopio. La computazione è veloce, di fatti è suggerita nelle applicazioni ad alta velocità.
- **M4G (Magnet for Gyroscope)**. Simile alla precedente modalità, tuttavia, il giroscopio non viene più utilizzato per la rilevazione della rotazione bensì la variazione dell'orientamento del magnetometro nel campo magnetico terrestre.
- **COMPASS**. In questa modalità, il *BNO-055* si comporta come una bussola, determinando l'orientamento del sensore rispetto al polo nord magnetico.
- **NDOF**. In questa modalità, la computazione dell'orientamento del dispositivo, è eseguita servendosi di tutti i dati provenienti dai sensori. La presente modalità è quella con il più alto consumo di corrente.

Nel presente progetto, la modalità **NDOF** è sempre stata scelta come modalità operativa del *BNO-055*. A seguire, un’immagine riportata dalla documentazione ufficiale, specificante la frequenza con cui il sensore produce dati in uscita, nel mentre che la *sensor fusion* è attiva.

BNO055 Operating Mode	Data input rate			Algo calling rate	Data output rate			
	Accel	Mag	Gyro		Accel	Mag	Gyro	Fusion data
IMU	100Hz	NA	100Hz	100Hz	100Hz	NA	100Hz	100Hz
COMPASS	20Hz	20Hz	NA	20Hz	20Hz	20Hz	NA	20Hz
M4G	50Hz	50Hz	NA	50Hz	50Hz	50Hz	NA	50Hz
NDOF_FMC_OFF	100Hz	20Hz	100Hz	100Hz	100Hz	20Hz	100Hz	100Hz
NDOF	100Hz	20Hz	100Hz	100Hz	100Hz	20Hz	100Hz	100Hz

Figure 1.19.: *BNO-055 Bosch Sensortec. Architettura di sistema*

Non-Fusion Modes

- **ACCONLY.** Il dispositivo fornisce solamente dati grezzi di provenienti dall’accelerometro, ponendo magnetometro e giroscopio nella modalità a basso consumo.
- **MAGONLY.** Se impostata, il dispositivo fornisce solamente i dati del magnetometro, sospendendo l’accelerometro e il magnetometro.
- **GYROONLY.** La modalità è simile alle precedenti, però, in questo caso il dispositivo fornisce solo le misure del giroscopio, sospendendo l’accelerometro ed il magnetometro.
- **ACCMAG.** In questa modalità, il sistema fornisce sia le misure dell’accelerometro che quelle del magnetometro. Il giroscopio è sospeso.
- **ACCGYRO.** Simile alla precedente, ma il sensore rende disponibili solo le misure dell’accelerometro e del giroscopio.
- **MAGGYRO.** In questo caso, il sensore fornisce le misure del magnetometro e del giroscopio, sospendendo l’accelerometro.
- **AMG (ACC-MAG-GYRO).** Con la presente modalità, il μC attiva tutti i sensori, fornendo le misure di accelerometro, giroscopio e magnetometro.

1.11. VL53L1X ST. Sensore Time-of-Flight per la misurazione a lunga distanza

Al fine di ottenere il comportamento desiderato del **D.P.D.F**, è necessario conoscere la distanza da terra mentre quest’ultimo è in volo. Bisogna, dunque, includere

un sensore apposito.

Il **VL53L1X** di **STMicroelectronics** è pienamente coerente con i requisiti precedentemente espressi. Tuttavia, non si è optato per l'integrazione diretta del sensore della casa STMicroelectronics, si è preferito acquistare il modulo **IRIS11A0J9776** prodotto da **Pololu**. La *breakout board* di Pololu, contenente il sensore VL53L1X è stata scelta per semplificare la fase di configurazione elettronica del dispositivo. Per comprendere il suo principio di misura, è opportuno dedicare un'intera sezione alla sua descrizione.

1.11.1. Il principio di misura di un sensore *Time-of-flight, ToF*

La modalità di misura conosciuta come *Time-of-Flight* si basa sulla determinazione del tempo impiegato da un segnale, generalmente un impulso luminoso nella banda dell'infrarosso, per compiere un viaggio di andata e ritorno tra un emettitore ed un bersaglio riflettente. In particolare, per il dispositivo in analisi, la sorgente di emissione luminosa è costituita da un laser a cavità verticale (VCSEL), che emette brevi impulsi di luce modulata. Il segnale emesso si propaga nell'ambiente fino ad incontrare un oggetto. Parte della radiazione viene riflessa e raccolta da un rilevatore sensibile alla luce, generalmente un *SPAD array* o "*Single-Photon Avalanche Diode*". Questo rilevatore è in grado di registrare fino all'arrivo di singoli fotoni, consentendo una misura estremamente sensibile del tempo di volo. Una volta noto l'intervallo temporale tra l'istante di emissione dell'impulso e quello di ricezione del suo "*eco*", il calcolo della distanza si ottiene applicando la relazione:

$$d = \frac{c \cdot \Delta t}{2} \quad (1.7)$$

dove d è la distanza dell'oggetto dalla superficie, c è la velocità della luce nel vuoto e Δt è il tempo di volo misurato. Il fattore 2 al denominatore tiene conto che il segnale percorre il tragitto due volte, andata e ritorno. I "ToF" moderni possono utilizzare tecniche avanzate di correlazione temporale o modulazione di fase per migliorare la precisione e la resistenza al rumore ambientale.

Nel caso in esame, il VL53L1X utilizza la tecnologia base ToF o dToF, "*Direct Time-of-Flight*".

1.11.2. VL53L1X STMicroelectronics. Caratteristiche fisiche

A seguire, un breve elenco delle principali caratteristiche fisiche:

- dimensioni del chip: $4.90 \times 1.25 \times 1.56$ [mm].
- dimensioni del modulo: $17.50 \times 12 \times 2.56$ [mm].
- massa del chip: 30 [mg].
- massa del modulo: 0.5 [g] (senza i *pin header*).

1.11.3. VL53L1X STMicroelectronics. Catatteristiche tecniche

Alimentazione

Il sensore presenta un unico ingresso di alimentazione il cui nome è V_{DD} . L'intervallo di tensione di alimentazione è: [2.6÷3.5][V]

Protocollo di comunicazione

Il dispositivo è stato sviluppato per comunicare con il microcontrollore utilizzando esclusivamente il protocollo di comunicazione I^2C . Il sensore dispone della capacità di comunicare in *Fast Mode*, scambiando byte ad una frequenza di 400 [kHz].

Il *VL53L1X* fornisce i pin di **Serial DAta (SDA)** e di **Serial CLock (SCL)**.

XSHUT e GPIO 1 (Interrupt)

Il *VL53L1X* dispone di un ingresso denominato **XSHUT**, che consente di spegnere o riavviare il dispositivo a livello *hardware*, indipendentemente dall'*Host*. Portando l'ingresso al livello logico basso, il sensore entra in modalità di *shutdown*, interrompendo le misurazioni e riducendo il consumo di corrente. Al contrario, appliccando un livello di tensione alto, il sensore viene inizializzato e reso operativo.

Oltre che l'ingresso **XSHUT**, il sensore dispone di un'**uscita** denominata **GPIO1**, spesso indicata anche come **INT (interrupt)**, la cui funzione principale è segnalare al microcontrollore eventi o stati del sensore.

Nel presente progetto, il pin XSHUT è stato utilizzato esclusivamente per il *reset* forzato del sensore, mentre il GPIO1 è stato impiegato per segnalare la disponibilità della misura, semplificando il firmware, riducendo il carico computazionale e consentendo la gestione della temporizzazione, aspetti che verranno approfonditi nella dedicata sezione nel capitolo *Software*.

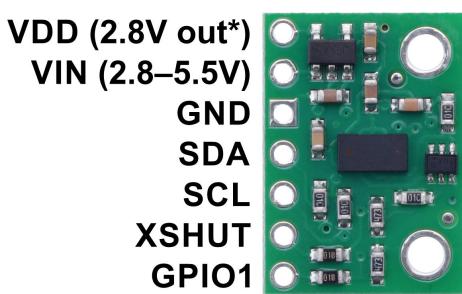


Figure 1.20.: P.C

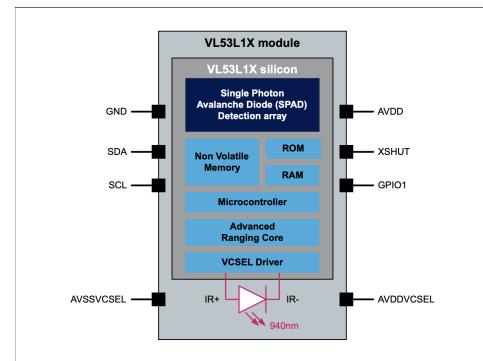


Figure 1.21.: S.B.D

L'immagine a sinistra, specifica la disposizione della piedinatura del modulo *Pololu* (*P.C.*: Pin Configuration), mentre, la figura di destra, rappresenta la suddivisione modulare del *VL53L1X* (*S.B.D.*: System Block Diagram).

Emettitore

Il **VL53L1X** monta un emettitore **laser ad infrarossi con lunghezza d'onda pari a 940 [nm] appartenente alla classe uno**. Quest'ultimo fa riferimento alla classificazione di sicurezza stabilita dalla norma internazionale **IEC 60825-1**. La classificazione ha il dovere di descrivere quanto un'emissione laser possa essere pericolosa per l'interazione umana (occhi, pelle ecc). La normativa organizza gli emettitori in *classi*, dove ogni classe rappresenta un livello crescente di rischio. Per un laser di *classe uno*, come quello emesso dal dispositivo, le condizioni operative normali, sono considerate sicure per occhi e pelle.



Figure 1.22.: VL53L1X, dettaglio sull'emettitore

Field of View (FoV)

Il *Field of View* è l'angolo massimo entro il quale il dispositivo rileva un oggetto. Per quanto riguarda l'emettitore in esame, il **FoV** è all'incirca di **27°**.

A seguire, un'immagine esplicativa del *FoV*:

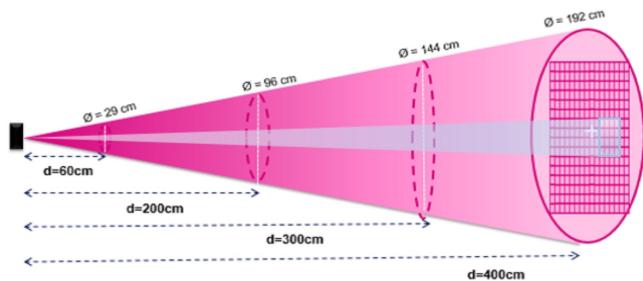


Figure 1.23.: Field of View del VL53L1X

Region-of-interest (R.O.I)

Con *Region-Of-Interest* si fa riferimento ad un'opzione offerta dal **VL53L1X** che permette di restringere il *Field of View* utilizzato nella misura. La funzione è utile

nel caso in cui si voglia migliorare la precisione di misura su oggetti piccoli.

Nel presente progetto, il sensore è responsabile della misura dell'altitudine del *D.P.D.F*, per cui la tecnologia (*R.O.I*) non è mai stata utilizzata, lasciando il *FoV* a 27°.

Prestazioni di misura, Timing Budget e modalità operative di misura

Il *VL53L1X* vanta di ottime prestazioni di misura, di fatti questa è accurata e veloce. Il dispositivo ha la capacità di misurare distanze fino ai 4 [m] ad una frequenza di 50 [Hz]. Con il fine di coprire una più vasta gamma di applicazioni, il dispositivo fornisce la possibilità di selezionare la modalità di misura più adatta alla specifica applicazione, oltre che la velocità e l'accuratezza. A seguire, una tabella esplicante le varie modalità di misura offerte dal dispositivo:

Table 1.1.: VL53L1X. Modalità di misura

Modalità	Range massimo in condizioni di scarsa luce (approssimato)	Range massimo con luce ambientale
<i>Short</i>	136 [cm]	135 [cm]
<i>Medium</i>	290 [cm]	76 [cm]
<i>Long</i>	360 [cm]	73 [cm]

Tabella riportata dal datasheet ufficiale del *VL53L1X*. I valori presentati, sono stati ricavati sotto le seguenti condizioni : timing budget = 100 [ms], bersaglio bianco, riflettanza : 88%, luce ambientale : 200 [kcps/SPAD].

La *Short ranging mode* è stata sviluppata per misure ravvicinate ad alta stabilità e in ambienti luminosi. La *Long range mode* è pensata, invece, per la massimizzazione della portata, al costo di accettare segnali più "nervosi". Oltre a quelle precedenti, il dispositivo offre la possibilità di selezionare la *Medium range mode*, che rappresenta un buon compromesso tra distanza operativa e robustezza al rumore.

L'ambiente in cui sono stati effettuati i test di volo è molto luminoso, inoltre, i fili di sostegno, si sono rilevati un vincolo per l'altitudine massima raggiungibile, così da non superare i 60 [cm].

Dunque, ai fini progettuali, la ***Short ranging mode*** è stata privilegiata, diventando l'unica modalità operativa utilizzata per tutto l'arco del progetto.

Con il fine di minuziosa gestione della temporizzazione e dell'accuratezza, è stata sfottata la funzione di **Timing Budget** del sensore. Con *Timing Budget* si fa riferimento al tempo totale che il sensore impiega per eseguire una singola misurazione di distanza. Il *Timing Budget* è espresso in [ms] e può assumere i seguenti valori: **[20÷1000] [ms]**. Un aumento del *Timing Budget* comporta una riduzione della frequenza di misura ma, al tempo stesso, ne migliora l'accuratezza e la stabilità.

La documentazione, suggerisce che un *Timing Budget* di **33 ms** offre ottime prestazioni in termine di velocità ed accuratezza, per cui, ai fini progettuali, è stato selezionato tale valore. Per la gestione dell'effettiva temporizzazione è stato variato l'*Inter-Measurement Period*, approfondito nella sezione *Software*.

Capitolo 1. Hardware

A seguire, un'immagine riportata direttamente dallo *User Manual* del *VL53L1X*, raffigurante le prestazioni di misura a fronte di diversi valori del *Timing Budget*.

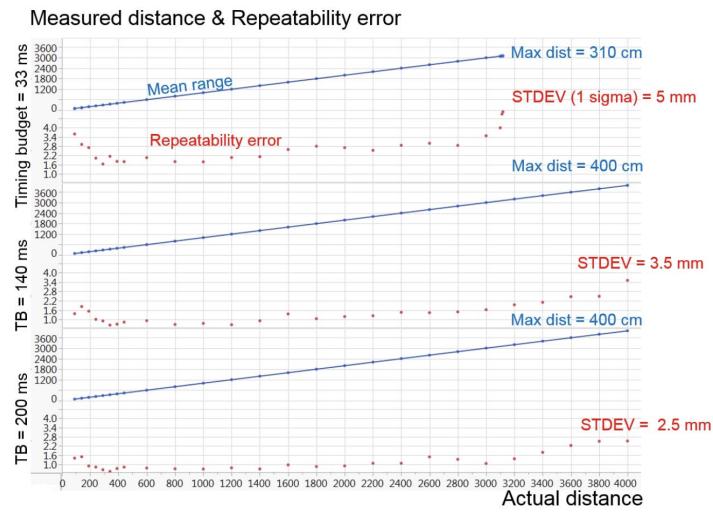


Figure 1.24.: VL53L1X, dettaglio sull'emettitore

Capitolo 2.

Firmware

Il *firmware* di controllo del *D.P.D.F.*, è stato sviluppato con l'obiettivo di ottenere un *hovering* del velivolo sufficientemente stabile.

L'inseguimento dell'obiettivo ha portato alla creazione di una libreria di funzioni utilizzando il linguaggio di programmazione **C**. Il *firmware* è stato interamente creato nell'**STCubeIDE**, l'ambiente di sviluppo integrato offerto da STMicroelectronics per la programmazione dei microcontrollori **STM32** da essa prodotti. La **versione** dell'*IDE* che ha accompagnato tutto lo sviluppo è stata la **1.16.1**.

L'*IDE* di *ST* integra in un'unica piattaforma, software: per la configurazione *hardware*, la generazione di codice basata sulle librerie **H.A.L** (*Hardware abstraction layer*) e strumenti di compilazione e *debug*.

Per la configurazione delle periferiche presenti sul microcontrollore *STM32H745ZI-TQ6* è stata utilizzata l'interfaccia di programmazione grafica messa a disposizione da *STCubeIDE*, *STCubeMX*. Lo sviluppo è stato completato usando la **versione 6.12.1** del *STCubeMX*.

Per la sviluppo *firmware*, invece, sono state implementate molte funzioni appartenenti alla libreria *H.A.L*. In corso d'opera, si è reso necessario uno studio e un successivo sviluppo di *sub-firmware* per ciascuna componente *hardware* richiedente di programmazione, per poi effettuare in un secondo momento, il *merge* del codice completo.

Per la gestione dei sensori **BNO-055 BoshSensortec** e **VL53L1X STMicroelectronics** sono stati integrati i *driver* ufficiali delle aziende produttrici, per cui l'unica azione di sviluppo intrapresa per i queste componenti è stata l'adattamento al contesto **H.A.L** (*Hardware Abstraction Layer*). Nel presente elaborato, i termini "procedura" e "funzione" saranno utilizzati come sinonimi, in quanto indicano la medesima entità.

2.1. Descrizione del firmware di gestione del VL53L1X STMicroelectronics. STSW-IMG007 FULL API

STMicroelectronics, produttore del VL53L1X, fornisce il *sub-firmware* ufficiale di gestione del dispositivo, chiamato **STSW-IMG007 FULL API**.

Con il fine di risparmiare tempo per il completamento del progetto, è stata preferita

l'implementazione del *sub-firmware* ufficiale piuttosto che lo sviluppo in autonomo. Il *sub-firmware* di ST è **multi piattaforma**, vale a dire che è stato concepito, per operare, oltre che su STCubeIDE, anche su altri ambienti di sviluppo. In virtù di ciò, si è reso necessario un adattamento all'ambiente di sviluppo utilizzato.

I produttori del *sub-firmware* hanno ridotto il lavoro necessario all'adattamento, difatti, per raggiungere tal fine è sufficiente metter mano alla cartella *platform*. Nello specifico, è risultata sufficiente la modifica di *vl53l1_platform.c*.

Al fine di garantire la corretta operatività del *sub-firmware*, si è optato per un adattamento tramite funzioni *H.A.L*.

2.1.1. STSW-IMG007. *vl53l1_platform.c*, il *source file* di adattamento

L'**A.P.I** fornisce funzioni, la cui definizione è a carico del consumatore finale, sono state individuate durante l'esecuzione del codice. Le segnalazioni di errore di *STCubeIDE* sono state utili alla localizzazione di tali **indefinite** funzioni.

Lo studio del contesto delle procedure da definire, è stato la chiave del corretto funzionamento.

Con il fine di migliorare la leggibilità del codice, si è optato per la divisione del *file* in due sezioni. La prima corrispondente al *sub-firmware* ufficiale, la seconda legata alla definizione delle funzioni di adattamento. Nel presente elaborato non verranno menzionati altri *file* al di fuori di *vl53l1_platform.c*, inoltre non verranno discusse procedure che non sono state sviluppate al fine dell'adattamento. La comunicazione *I²C* con il dispositivo è gestita dalla periferica I2C2 del'STM32H745ZI-Q-TQ6.

Al fine di verificare la corretta operatività delle funzioni implementate, si è optato per l'utilizzo di un terminale per la stampa a video della informazioni. Il terminale in questione è fornito dal *software* esterno **PuTTy**. Per comunicare i risultati circa l'operatività delle implementazioni, è stata utilizzata la capacità di *PuTTy* di stampare ciò che viene inviato tramite protocollo seriale UART. Nello specifico, è stato utilizzata la periferica USART/UART3 della NUCLEO-H745ZI-Q. Entrambe le periferiche di comunicazione vengono gestite dall'*H.A.L*, tramite le strutture **I2C_HandleTypeDef** e **UART_HandleTypeDef**.

Con il fine di includere la libreria di funzioni *H.A.L*, è stata necessaria l'inclusione di "stm32h7xx_hal.h" in "vl53l1_platform.c".

Per evitare problemi di compilazione, nel *file* "vl53l1_platform.c" sono state ridefinite le variabili di controllo di tali periferiche con direttiva "**extern**". A seguire, le direttive di inclusione ed extern aggiunte nel file.

```

1
2 //direttive di inclusione aggiunte
3 #include "stdio.h"
4 #include "stdint.h"
5 #include "stm32h7xx_hal.h"
6
7 //direttive di inclusione originali
8 #include "vl53l1_platform.h"
```

```

9      #include "vl53l1_platform_log.h"
10
11     //direttive extern
12     extern I2C_HandleTypeDef hi2c2;
13     extern UART_HandleTypeDef huart3;
```

Listing 2.1: *path* delle direttive di inclusione di *vl53l1_platform.c*, in aggiunta la ridefinizione delle variabili di gestione della comunicazione

Modalità di gestione delle eccezioni

Il *sub-firmware*, per ragioni deducibili, non gestisce le eccezioni mediante implementazione *H.A.L.* Possiede invece un sistema di gestione delle eccezioni originale.

La struttura utilizzata per segnalare l'insorgenza di eccezioni è un *alias* di tipo: **VL53L1_Error** rappresenta infatti il tipo di dato impiegato dal *sub-firmware* a tale scopo. Di conseguenza tutte le funzioni di *"vl53l1_platform.c"* restituiscono il tipo di dato citato in precedenza.

Lo stesso sistema di gestione delle eccezioni, è stato utilizzato durante lo sviluppo, con il fine di verificare il corretto funzionamento delle implementazioni.

In alcune definizioni, è stata forzata la "collaborazione" tra la struttura di gestione delle eccezioni della libreria *H.A.L* e quella presente nel *sub-firmware* sotto analisi. Questa "collaborazione" si è resa necessaria per garantire il corretto funzionamento della libreria all'interno dell'IDE.

```

1
2 typedef int8_t VL53L1_Error;
3
4 #define VL53L1_ERROR_NONE ((VL53L1_Error) 0)
5 #define VL53L1_ERROR_CALIBRATION_WARNING ((VL53L1_Error) - 1)
6     /*!< Warning invalid calibration data may be in used
7         \a VL53L1_InitData()
8         \a VL53L1_GetOffsetCalibrationData
9         \a VL53L1_SetOffsetCalibrationData */
10 #define VL53L1_ERROR_MIN_CLIPPED ((VL53L1_Error) - 2)
11     /*!< Warning parameter passed was clipped to min before to be
12         applied */
13
14 #define VL53L1_ERROR_UNDEFINED ((VL53L1_Error) - 3)
15
16 #define VL53L1_ERROR_INVALID_PARAMS ((VL53L1_Error) - 4)
17     /*!< Parameter passed is invalid or out of range */
18 #define VL53L1_ERROR_NOT_SUPPORTED ((VL53L1_Error) - 5)
19     /*!< Function is not supported in current mode or configuration */
20 #define VL53L1_ERROR_RANGE_ERROR ((VL53L1_Error) - 6)
21     /*!< Device report a ranging error interrupt status */
22 #define VL53L1_ERROR_TIME_OUT ((VL53L1_Error) - 7)
23     /*!< Aborted due to time out */
24 #define VL53L1_ERROR_MODE_NOT_SUPPORTED ((VL53L1_Error) - 8)
25     /*!< Asked mode is not supported by the device */
26 #define VL53L1_ERROR_BUFFER_TOO_SMALL ((VL53L1_Error) - 9)
```

```

26     /*!< ...
27 #define VL53L1_ERROR_COMMs_BUFFER_TOO_SMALL ((VL53L1_Error) - 10)

```

Listing 2.2: Dettaglio sulla struttura di gestione delle eccezioni presente in *vl53l1_error_codes.h* da notare, il cast a dato VL53L1_Error nelle direttive *define*

2.1.2. STSW-IMG007. Procedure di adattamento

A seguire, la descrizione delle procedure implementate con il fine di ottenere il coretto interfacciamento nel contesto *H.A.L.*

Funzione: RANGING_SENSOR_COMMS_Init_CCI

Nell’analisi della funzione ”**VL53L1_CommsInitialise**”, il cui compito è quello di inizializzare le periferiche di comunicazione in utilizzo, è stata individuata la prima procedura utile all’adattamento.

La funzione **RANGING_SENSOR_COMMS_Init_CCI** è invocata dalla precedente procedura, con lo scopo di ottenere la verifica della corretta operatività della periferica di comunicazione in uso. Per lo sviluppo della definizione della funzione è stata necessaria l’integrazione dell’*H.A.L.*.

La componente *H.A.L* utilizzata per l’adattamento è **HAL_I2C_GetState**, che restituisce lo stato interno della periferica I^2C richiesta mediante un tipo di dato enumerato definito nel panorama ”**HAL_I2C_StateTypeDef**”. Se nei tempi pattuiti viene restituito il valore **HAL_I2C_STATE_READY**, allora la periferica di comunicazione è pronta per condurre informazioni. ”**VL53L1_CommsInitialise**” è stata, inoltre, modificata per adattarsi al valore di restituzione di **RANGING_SENSOR_COMMS_Init_CCI**.

Se dovesse verificarsi la restituzione di ”**HAL_I2C_STATE_READY**” da parte di ”**RANGING_SENSOR_COMMS_Init_CCI**”, ”**VL53L1_CommsInitialise**” risulterebbe invocata con successo, altrimenti verrà gestita l’eccezione tramite restituzione di un parametro ”**VL53L1_CommsInitialise**”.

```

1 HAL_I2C_StateTypeDef RANGING_SENSOR_COMMS_Init_CCI(uint8_t param1,
2           uint8_t param2, uint8_t param3){
3
4     (void)param1;
5     (void)param2;
6     (void)param3;
7
8     HAL_I2C_StateTypeDef status_CCI = HAL_I2C_GetState(&hi2c2);
9     return status_CCI;
}

```

Listing 2.3: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_Init_CCI*

```

1      if(RANGING_SENSOR_COMMS_Init_CCI(0, 0, 0) != HAL_I2C_STATE_READY){
2
3          RANGING_SENSOR_COMMS_Get_Error_Text(comms_error_string);
4          status = VL53L1_ERROR_CONTROL_INTERFACE;
5
6      }

```

Listing 2.4: Dettaglio

dell’implementazione di *RANGING_SENSOR_COMMS_Init_CCI* in
VL53L1_CommsInitialise

Funzione: RANGING_SENSOR_COMMS_Get_Error_Text

La funzione intitolante appare nell’ambito contestuale di ”*VL53L1_CommsInitialise*”. Il suo ruolo è quello di restituire al *main flow* il messaggio di errore associato a ”*RANGING_SENSOR_COMMS_Init_CCI*”.

Con il fine di stampare a video, sul terminale offerto da *PuTTy* il messaggio interessato, è stato sviluppato un’adattamento utilizzando la funzione **HAL_UART_Transmit**. La funzione della libreria *H.A.L* gestisce la trasmissione dei dati sulla periferica *UART* in modalità *polling*. Essa blocca l’esecuzione del microcontrollore fino al completamento della trasmissione, monitorandone lo stato della periferica ed interrompendo l’operazione in caso di *timeout* o errore.

La procedura ”*RANGING_SENSOR_COMMS_Get_Error_Text*” non restituisce alcun parametro. A seguire la sua definizione.

```

1 void RANGING_SENSOR_COMMS_Get_Error_Text(char *error_string){
2
3     sprintf(error_string, "VL53L1_CommsInitialise:
4         RANGING_SENSOR_COMMS_Init_CCI() failed\r\n");
5     HAL_UART_Transmit(&huart3, (uint8_t*)error_string, strlen(
6         error_string), HAL_MAX_DELAY);
7 }

```

Listing 2.5: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_Get_Error_Text*

Funzione: RANGING_SENSOR_COMMS_Fini_CCI

Lo ”stile” di collocazione della funzione intitolante, è analogo a quello analizzato poc’anzi in ”*RAGING_SENSOR_COMMS_Init_CCI*”.

La procedura è stata individuata in ”*VL53L1_CommsClose*”, il cui compito è la chiusura della comunicazione con il dispositivo. ”*RANGING_SENSOR_COMMS_Fini_CCI*” è stata sviluppata al fine di deinizializzare la periferica di comunicazione in utilizzo, interrompendo lo scambio di informazioni.

L’operatività della procedura è nelle mani della funzione ”*HAL_I2C_DeInit*”. Quest’ultima disabilita la periferica utilizzata, ripristina i *pins*, cancella eventuali *interrupt* attive e ripristina le configurazioni *hardware* della periferica *I²C*.

”HA_StatusTypeDef” è il valore restituito da ”HAL_I2C_DeInit” e, in questo caso, anche di ”RANGING_SENSOR_COMMS_Fini_CCI”.

In analogia con la precedente implementazione, la funzione invocatrice è stata modificata per adattarsi all’implementazione.

A seguire, la definizione della funzione e un breve frammento specificante il suo ruolo in ”VL53L1_CommsClose”.

```

1 HAL_StatusTypeDef RANGING_SENSOR_COMMS_Fini_CCI(void){
2
3     HAL_StatusTypeDef status_DeInit = HAL_I2C_DeInit(&hi2c2);
4     return status_DeInit;
5 }
```

Listing 2.6: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_Fini_CCI*

```

1
2     if(RANGING_SENSOR_COMMS_Fini_CCI() != HAL_OK){
3
4         RANGING_SENSOR_COMMS_Get_Error_Text(comms_error_string);
5         status = VL53L1_ERROR_CONTROL_INTERFACE;
6
7     }
```

Listing 2.7: Dettaglio sull’invocazione di *RANGING_SENSOR_COMMS_Fini_CCI* in *VL53L1_CommsClose*

Funzione: **RANGING_SENSOR_COMMS_Write_CCI**

La procedura intitolante, è situata nel contesto della funzione **VL53L1_WriteMulti**. ”VL53L1_WriteMulti” è centrale nel corretto esercizio del dispositivo. La funzione, infatti, scrive più **byte** consecutivi in un registro interno del sensore **VL53L1X**, partendo da uno specifico indirizzo.

”RAGING_SENSOR_COMMS_Write_CCI” è stata sviluppata al fine adattare questa necessità alla piattaforma in utilizzo.

Per l’adattamento, si è optato per uno sviluppo integrante la funzione **HAL_I2C_Master_Transmit**. La funzione della libreria *H.A.L*, invia al dispositivo solamente dati, mentre l’altra in aggiunta, anche l’indirizzo interno.

Tornando all’analisi di ”RANGING_SENSOR_COMMS_Writr_CCI”, la funzione costruisce un *buffer* temporaneo di dimensioni pari alla somma del numero di dati da inviare a due **byte** aggiuntivi.

Successivamente, i dati da scrivere, puntati dall’argomento *pData*, vengono copiati nel *buffer* a partire della terza posizione.

Il *buffer* così composto viene infine inviato al dispositivo *slave* specificato.

Il valore di ritorno di ”RANGING_SENSOR_COMMS_Write_CCI” è lo stesso di ”HAL_I2C_Master_Transmit”.

Capitolo 2. Firmware

Al fine di implementare "RANGING_SENSOR_COMMS_Write_CCI" è stata sviluppata una modifica di "VL53L1_WriteMulti".

A seguire, la definizione della funzione.

```
1 HAL_StatusTypeDef RANGING_SENSOR_COMMS_Write_CCI(uint16_t
2     i2c_slave_address, uint8_t dummy, uint16_t index, uint8_t *pdata,
3     uint32_t data_size){
4
5     uint8_t buffer[data_size+2];
6     buffer[0] = (uint8_t)(index>>8);
7     buffer[1] = (uint8_t)(index & 0xFF);
8
9     //copia dei dati nel buffer
10    for(uint32_t i = 0; i < data_size; i++){
11        buffer[i+2] = pdata[i];
12    }
13
14    HAL_StatusTypeDef status_Write_CCI = HAL_I2C_Master_Transmit(&
15        hi2c2, i2c_slave_address, buffer, (uint16_t)(data_size+2),
16        HAL_MAX_DELAY);
17
18    return status_Write_CCI;
19 }
```

Listing 2.8: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_Write_CCI*

```
1
2     if(RANGING_SENSOR_COMMS_Write_CCI(pdev->i2c_slave_address, 0,
3         index+position, pdata+position, data_size) != HAL_OK){
4
5         status = VL53L1_ERROR_CONTROL_INTERFACE;
6     }
```

Listing 2.9: Dettaglio sull'implementazione di *RANGING_SENSOR_COMMS_Write_CCI* in *VL53L1_WriteMulti*

Funzione: **RANGING_SENSOR_COMMS_Read_CCI**

La procedura ha il compito di leggere informazioni da uno specifico registro interno del **VL53L1X**.

Al fine di adempire al compito, è stata integrata la funzione "HAL_I2C_Mem_Read". "RANGING_SENSOR_COMMS_Read_CCI" è invocata internamente alla funzione "VL53L1_ReadMulti".

Anche in questo caso, l'ambiente interno della funzione invocatrice è stato adattato all'implementazione. A seguire, la definizione e il frammento dell'implementazione.

```
1 HAL_StatusTypeDef RANGING_SENSOR_COMMS_Read_CCI(uint16_t
2     i2c_slave_address, uint8_t dummy, uint16_t index, uint8_t *pdata,
3     uint32_t data_size){
```

```

2
3     HAL_StatusTypeDef status = HAL_I2C_Mem_Read(&hi2c2,
4         i2c_slave_address, index, I2C_MEMADD_SIZE_16BIT, pdata,
5         data_size, HAL_MAX_DELAY);
6
7 }

```

Listing 2.10: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_Read_CCI*

```

1
2     if(RANGING_SENSOR_COMMS_Read_CCI(pdev->i2c_slave_address, 0, index
3         +position, pData+position, data_size) != HAL_OK){
4
5         status = VL53L1_ERROR_CONTROL_INTERFACE;
6     }

```

Listing 2.11: Dettaglio sull'implementazione di *RANGING_SENSOR_COMMS_Read_CCI* in *VL53L1_ReadMulti*

Funzione: RANGING_SENSOR_COMMS_GPIO_Set_Mode

Il titolo presenta una procedura situata nella funzione "VL53L1_GpioSetMode", il cui compito è quello di configurare la modalità operativa di uno dei *pin* di GPIO del sensore *VL53L1X*.

La funzione non utilizza procedure *H.A.L* per adempiere al compito, modifica direttamente la struttura *H.A.L* "GPIO_InitStruct." "GPIO_InitStruct" è modificata in modo tale da configurare la modalità operativa dei *pin* **GPIO** utilizzati nel sensore *VL53L1X*.

La funzione riceve come parametri l'identificatore del *pin* e la modalità di configurazione desiderata. In base al *pin* specificato, la funzione configura il relativo registro "GPIO".

Nel caso in cui venga passato un *pin* non riconosciuto, la funzione restituisce un errore di tipo "**HAL_ERROE**", inviando un messaggio di errore sulla porta **USART/TUART3**.

Se la configurazione ha successo, viene eseguita la funzione "HAL_GPIO_Init()" per applicare le modifiche ai *pin*. Anche in questa sede è stata modificata la funzione invocatrice per la corretta implementazione.

A seguire, la definizione della funzione e il frammento dell'implementazione.

```

1 HAL_StatusTypeDef RANGING_SENSOR_COMMS_GPIO_Set_Mode(uint16_t pin,
2             uint32_t mode){
3
4     GPIO_InitTypeDef GPIO_InitStruct ={0};
5     HAL_StatusTypeDef status = HAL_OK;

```

```

5
6     switch(pin){
7
8         case GPIO_XSHUTDOWN:
9             GPIO_InitStruct.Pin = (uint32_t)GPIO_XSHUTDOWN;
10            GPIO_InitStruct.Mode = mode;
11            GPIO_InitStruct.Pull = GPIO_NOPULL;
12            GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
13            break;
14        case GPIO_POWERENABLE:
15            GPIO_InitStruct.Pin= (uint32_t)GPIO_POWERENABLE;
16            GPIO_InitStruct.Mode = mode;
17            GPIO_InitStruct.Pull = GPIO_NOPULL;
18            GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
19            break;
20        case GPIO_INTERRUPT:
21            GPIO_InitStruct.Pin = (uint32_t)GPIO_INTERRUPT;
22            GPIO_InitStruct.Mode = mode;
23            GPIO_InitStruct.Pull = GPIO_NOPULL;
24            GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
25            break;
26        default:
27            status = HAL_ERROR;
28            break;
29    }
30
31    if(status == HAL_OK){
32        HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
33        return status;
34    }else{
35        char error_GPIO_set_string[ERROR_TEXT_LENGTH];
36        sprintf(error_GPIO_set_string, "VL53L1_GPIO_SET_MODE HAL_ERROR
37            : %d\r\n",status );
38        HAL_UART_Transmit(&huart3, (uint8_t*)error_GPIO_set_string,
39                          strlen(error_GPIO_set_string), HAL_MAX_DELAY);
40        return status;
41    }
42 }
```

Listing 2.12: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_GPIO_Set_Value*

```

1
2     if(RANGING_SENSOR_COMMS_GPIO_Set_Mode(pin, value) != HAL_OK){
3
4         status = VL53L1_ERROR_CONTROL_INTERFACE;
5
6     }
```

Listing 2.13: Dettaglio sull'implementazione di *RANGING_SENSOR_COMMS_GPIO_Set_Value* in *VL53L1_GpioSetMode*

Funzione: RANGING_SENSOR_COMMS_GPIO_Set_Value

La funzione è invocata in molte posizioni nel *file*.

Il suo compito è quello di impostare il valore desiderato sul *pin* specificato. A seguire, la sua definizione.

```

1 HAL_StatusTypeDef RANGING_SENSOR_COMMS_GPIO_Set_Value(uint16_t pin,
2             uint8_t value){
3
4     HAL_StatusTypeDef status = HAL_OK;
5     if(value == GPIO_PIN_SET){
6         HAL_GPIO_WritePin(GPIOB, pin, GPIO_PIN_SET);
7         return status;
8     }else{
9         HAL_GPIO_WritePin(GPIOB, pin, GPIO_PIN_RESET);
10    return status;
11 }
```

Listing 2.14: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_GPIO_Set_Value*

Funzione: RANGING_SENSOR_COMMS_GPIO_Get_Value

La funzione è invocata in "VL53L1_GpioGetValue", il cui compito è restituire il valore impostato sul *pin* richiesto.

La procedura intitolante, serve solo all'adattamento con la piattaforma.

La definizione sfrutta la funzione, "HAL_GPIO_ReadPin", che legge lo stato del *pin* passato come parametro. A seguire, la sua definizione.

```

1 HAL_StatusTypeDef RANGING_SENSOR_COMMS_GPIO_Get_Value(uint16_t pin,
2             uint8_t *value){
3
4     HAL_StatusTypeDef status = HAL_OK;
5
6     if(HAL_GPIO_ReadPin(GPIOB, pin) == GPIO_PIN_SET){
7         value = 1;
8     }else{
9         value = 0;
10    }
11 }
```

Listing 2.15: Estratto di codice C dal file *vl53l1_platform.c*: la funzione *RANGING_SENSOR_COMMS_GPIO_Get_Value*

Introduzione

La qualità della vita dipende sempre di più dai sistemi infrastrutturali, i quali influenzano la società contemporanea fornendo ogni tipo di bene. Tuttavia, queste

sono soggette a deterioramento e, per evitare guasti o problemi più gravi, è necessario affrontare diverse problematiche nelle procedure di manutenzione. L'elevato costo, unito alla difficoltà per gli operatori umani di raggiungere i luoghi di intervento, specialmente considerando gli aspetti legati alla sicurezza, costituiscono forti limiti all'applicazione reale. Ragion per cui, i ricercatori hanno esplorato nuove modalità per automatizzare le procedure di ispezione e, attualmente [A1], la maggior parte degli **Unmanned Aerial Vehicles (UAV)** proposti per supportare le ispezioni infrastrutturali sono droni.

Nonostante le innovative soluzioni, persistono ancora importanti limitazioni da superare: queste vanno dal miglioramento delle prestazioni dei sistemi robotici, fino alle problematiche cooperative e di sicurezza. I limiti principali nelle missioni di ispezione riguardano in primo luogo le eliche dei droni, pericolose ed inadatte alle ispezioni non distruttive a contatto. Inoltre, quando questi UAV vengono impiegati in compiti cooperativi con l'essere umano, è necessaria un'interazione sicura, che non può essere garantita da droni convenzionali.

Un'ulteriore sfida consiste nel controllo preciso di questi *robot* a base flottante: in molti casi è richiesta un'elevata precisione e, attualmente, per ottenere una buona stabilità, si ricorre a soluzioni con molteplici eliche, a discapito degli elevati costi. Il **Dipartimento di ingegneria dell'informazione** dell'Università Politecnica delle Marche si è impegnato ad affrontare questi problemi presentando un'architettura alternativa di UAV il *Double Propeller Ducted-Fan*, considerata vantaggiosa per le ispezioni sicure e l'interazione uomo-macchina.

Il progetto, attualmente, è nella mani del laboratorio di automazione nel D.I.I dell'Università Politecnica delle Marche, a portata degli studenti dei corsi di ingegneria. L'architettura *hardware* e *software* del *D.P.D.F* non ha subito variazioni di grande portata nel tempo. Tuttavia, lo sviluppo tecnologico ha registrato notevoli avanzamenti e dunque si rende necessario un aggiornamento significativo del profilo tecnologico del drone. Ergo, il fine ultimo del progetto di tesi è stato lo "*Sviluppo e porting di firmware per un drone di tipo D.P.D.F.*".

Con il termine *porting* si fa riferimento al processo di adattamento di un *firmware* sviluppato per una piattaforma *hardware* specifica, così che possa funzionare su un'altra piattaforma, diversa dalla precedente. Nel presente caso è stato eseguito il *porting* dalla scheda di controllo **Renesas Demonstration Kit YRDKRX63N** alla scheda **STM NUCLEO-H745ZI-Q**. Inoltre, sono stati apportati degli aggiornamenti al comparto servomotori e sensori. I precedenti '*servos*', **DS3115 MG Digital Servo**, sono stati sostituiti con gli **Hitec HS-82MG**, di minor peso. In merito ai sensori, il pregresso modulo di misura inerziale (I.M.U) **MPU6050 InvenSense** è stato aggiornato con il **BNO-055 Bosch Sensortec**, integrante di un microcontrollore deputato all'esecuzione di un algoritmo di fusione sensoriale [A2].

Capitolo 3.

Sistema

NOTA valido per ogni capitolo: inserire all'inizio di ogni capitolo una brevissima descrizione del contenuto del capitolo stesso.

NOTA: capitolo opzionale. Solo se necessario per comprendere quanto svolto, inserite un capitolo dedicato al sistema su cui avete lavorato (es drone quadrirotore, ballbot ..) dove descrivete ad esempio come è fatto il sistema, come si modella matematicamente e come si può controllare. Nel capitolo inserite solo le informazioni che avete effettivamente utilizzato per svolgere il task assegnatovi (**evitate il copia e incolla dalle vecchie relazioni**). Mettete la fonte da cui avete preso le informazioni (es vecchia relazione [?]).

3.1. Modello matematico

3.2. Controllo

Capitolo 4.

Hardware

Inserire sempre un capitolo "Hardware" dove si descrivono tutti i componenti hardware utilizzati (schede, sensori, attuatori..). Dedicare un paragrafo ad ogni componente. Per ogni componente riportare le seguenti informazioni: modello specifico, riferimento bibliografico con link ad un sito web con la documentazione del componente (es [?]), immagine, pinout, solo le informazioni rilevanti per lo svolgimento del task.

4.1. Componente1

4.2. Componente2

4.3. Schema dei collegamenti

Inserire uno schema dei collegamenti analogo a quello in Figura 4.1. Si suggerisce per lo scopo di usare il software "draw.io". Nello schema riportare tutti i collegamenti tra i componenti, specificando quale pin di un componente1 si collega con quale pin di un componente2 (per la maggior parte dei collegamenti dovete cioè avere una linea con 2 label).

Inserire anche una descrizione testuale dello schema.

Capitolo 4. Hardware

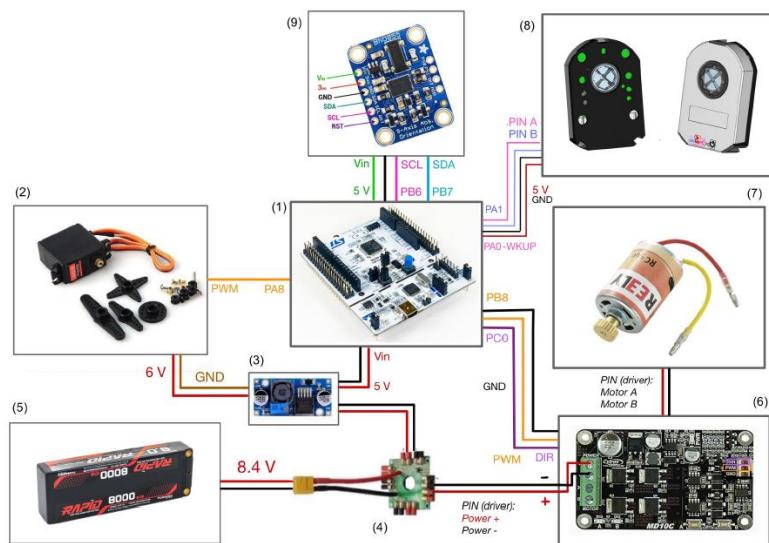


Figure 4.1.: Schema dei collegamenti

Capitolo 5.

Software

Nell'introduzione al capitolo specificate anche la versione dell'STM32CubeIDE e dell'STM32CubeMX che avete usato (se avete cambiato versione nel corso del progetto mettete l'ultima, quella su cui è sviluppato il codice che consegnate).

5.1. Diagramma di flusso

Inserire un diagramma di flusso analogo a quello in Figura 5.1 che spiega il funzionamento generale del codice sviluppato. Inserire anche una descrizione testuale del funzionamento.

5.2. Gestione singoli componenti

Dedicare un paragrafo ad ogni componente in cui viene spiegato il codice che lo gestisce. Inserite prima una descrizione generale di come si gestisce un componente di quel tipo (aiutandovi con schemi o altro). Poi le configurazioni che sono state fatte sul .ioc per poterlo utilizzare (quale periferiche sono state abilitate, come sono state configurate.. mettete degli screen del .ioc). Infine inserite e spiegate le parti di codice che lo gestiscono.

NOTA per il codice: per rendere il codice modulare e riutilizzabile è buona prassi non caricare troppo il main ma creare una libreria per ogni componente. Quindi il componente1 avrà un header file "componente1.h" e un source file "componente1.c", nel source file sono implementate le funzioni che gestiscono il componente, nell'header file ci sono i prototipi di tali funzioni in modo che esse possano essere usate nel main includendolo.

NOTA per il codice: non inserire mai nel codice dei parametri numerici senza contesto ma renderli delle costanti definite (usando la direttiva %define). Se sono delle costanti relative ad uno specifico componente vanno inserite nel relativo header file.

NOTA per il codice: per inserire il codice, anziché utilizzare screenshot utilizzate i seguenti comandi.

Per codice scritto in latex (con linguaggio Python):

Listing 5.1: "Codice in Python"

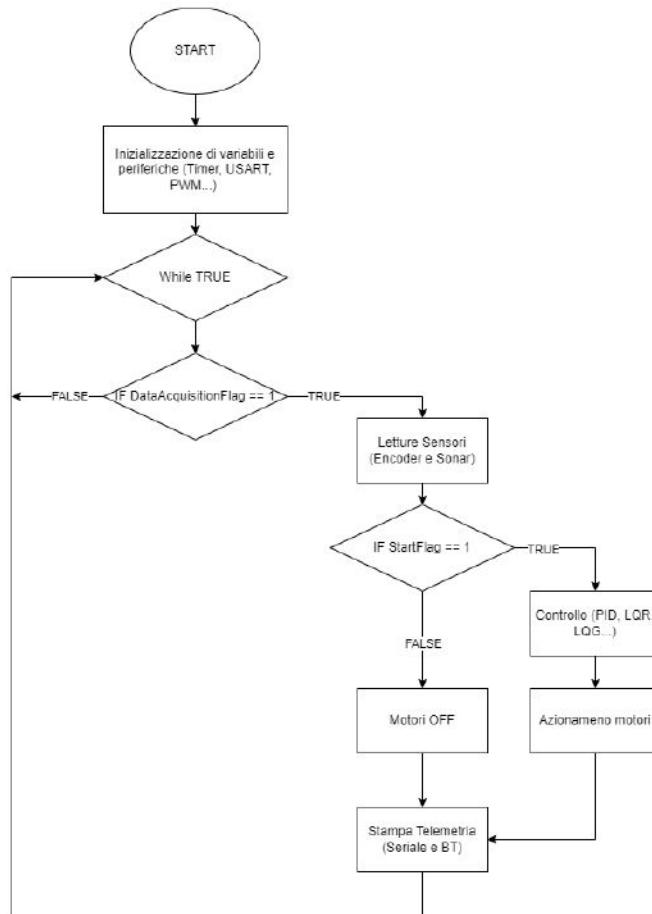


Figure 5.1.: Diagramma di flusso del codice sviluppato

```

import numpy as np

def incmatrix(genl1 ,genl2):
    m = len(genl1) # The length of the first array
    n = len(genl2) # The length of the second array
    sum = 0

    # Compute
    for i in range(0 ,n):
        for j in range(0 , m):
            sum += genl1 [n]* genl2 [m]

    # Print
    print( "The sum is " ,sum)

    return M
  
```

Funziona anche per codice MATLAB:

```
1 %% PREPARE WORKSPACE
2 close all
3 clearvars
4 clc
5
6 %% OPERATIONS
7 sayhello;
8
9 %% FUNCTIONS
10 function sayhello
11     fprintf("Hello world!");
12 end
```

Listing 5.2: "Codice in MATLAB"

5.2.1. Gestione componente1

5.2.2. Gestione componente2

5.3. Funzionamento complessivo

Dopo aver spiegato il codice che gestisce i singoli componenti, inserire e commentare le porzioni di codice relative al funzionamento complessivo del programma.

Capitolo 6.

Test e risultati

Dedicate un capitolo a tutti i test effettuati con relativi risultati, includete sia i test finali relativi al funzionamento complessivo, sia i test delle singole parti (se significativi).

Descrivete in dettaglio le condizioni in cui sono stati svolti i test in modo che siano ripetibili. Durante i test fate dei video e acquisite i dati (es usando la funzione di log di Putty), per i test più rilevanti è opportuno consegnare anche questo materiale per documentare i test effettuati. Nella relazione riportate i risultati con dei grafici come quello in Figura 6.1 (inserite sempre nei grafici le label sugli assi con grandezza e relativa unità di misura). Commentate in modo critico i risultati ottenuti, evidenziando sia quelli positivi sia quelli negativi.

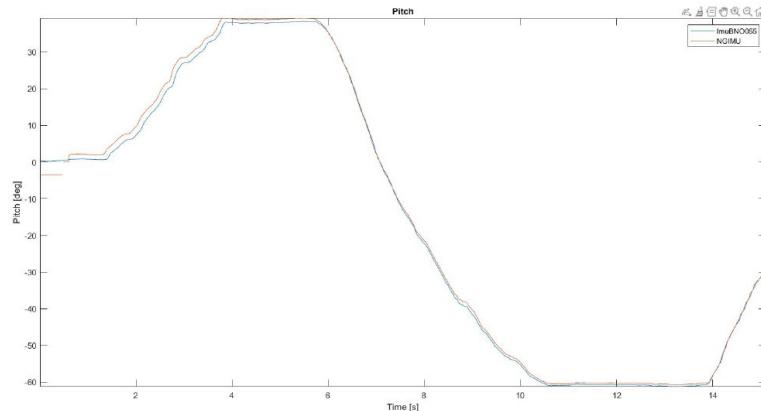


Figure 6.1.: Esempio di grafico

6.1. Test1

Per ogni test riportate: cosa state testando, in che condizioni si è svolto il test, riferimenti a video/file di log che consegnate insieme a relazione e codice, grafici dei risultati, commento dei risultati.

6.2. Test2

Conclusioni e sviluppi futuri

Riassumere brevemente il lavoro svolto rispetto al task assegnato, evidenziando quali risultati sono stati raggiunti e quali no. Se ci sono aspetti del task non completati spiegare quali sono stati i problemi riscontrati in merito.
Inserire considerazioni personali su possibili sviluppi futuri dell'attività svolta (idee per migliorarla che non avete avuto modo di sperimentare, aspetti che suggerite di approfondire, problemi da risolvere..).

Appendice A.

Appendice1

Se necessario ricorrete alle appendici per spiegare le parti "di contorno" dell'attività svolta e/o ciò che non riuscite ad inserire nello schema generale dei capitoli della relazione (es acquisizione dei dati con Matlab).

Appendice B.

Il mio primo capitolo con L^AT_EX

B.1. Introduzione

Benvenuto! Se sei alle prime armi con il linguaggio L^AT_EX, di seguito troverai le informazioni di base per iniziare la scrittura della tua tesi e per capire come organizzarla. Per un maggiore approfondimento, si consiglia la lettura di due guide (in italiano): *L'arte di scrivere con L^AT_EX* e *L^AT_EX per l'Impaziente* di Lorenzo Pantieri. Inoltre, sul web sono presenti un gran numero di forum dedicati: con una semplice ricerca è possibile esaurire ogni curiosità e risolvere qualsiasi problema.

B.2. Organizzazione dei files

Come puoi notare da questa demo, nella cartella di lavoro sono presenti due file: il primo, chiamato `univpmthesis.cls`, definisce tutti i comandi della classe, e gestisce tutte le feature grafiche e di formattazione che caratterizzeranno la tua tesi. Non modificarlo se non sei un utente esperto!

Il file `UNIVPMthesis.tex` invece rappresenta il file principale (in gergo chiamato **main file**) da cui iniziare la scrittura del tuo testo. Fondamentalmente, questo gestisce l'organizzazione del tuo lavoro di tesi e contiene tutte le informazioni per la corretta compilazione del manoscritto. È qui, infatti, che andrai a definire la classe del documento, il suo preambolo, e la successione dei capitoli e del materiale iniziale e finale.

Noterai che nel main file non è esplicitato il contenuto di ogni singolo capitolo, bensì vengono importati i dati di scrittura da files .tex secondari con il comando `input`. Tale approccio risulta particolarmente comodo di fronte alla gestione di elaborati molto lunghi, in cui ciascun capitolo può essere scritto in un file .tex dedicato.

Tutti i capitoli che costituiscono il corpo della dissertatione si trovano nella cartella `chapters`, per ognuno dei quali è stata creata una sottocartella ad hoc. Ciascuna delle sottocartelle conterrà dunque il corrispondente documento del capitolo e le immagini che vengono incluse nello stesso. Questo modo di procedere, sebbene possa sembrare laborioso, è di enorme aiuto qualora ti trovassi a gestire un gran numero di immagini e grafici. L'unica accortezza consta nel dichiarare la locazione dei file immagine

che si vogliono includere nel capitolo all'inizio dello stesso mediante il comando `graphicpath` (un esempio è dato alla primissima riga del file `chapter1.tex`).

Analogamente, la cartella `frontbackmatters` racchiude tutto ciò che non costituisce il corpo del testo, come la Bibliografia ed il capitolo di Appendice.

Poichè il template deriva dalla classe KOMA-script `scrbook`, la dichiarazione del capitolo segue sia il tradizionale comando `chapter` sia il comando `addchap`. Quest'ultimo, infatti, è utile qualora si volesse inserire un capitolo (come l'Introduzione o le Conclusioni) senza la numerazione progressiva automatica.

B.2.1. Opzioni della classe

Poichè `univpmthesis.cls` si basa su `univpmphdthesis.cls`, le opzioni offerte dalla classe sono le seguenti:

a4print definisce le dimensioni del foglio di scrittura in formato A4, permettendo all'utente di controllare meglio l'occupazione degli spazi. Il margine sinistro varia da pagina pari a dispari per tener conto dello spazio extra necessario per la rilegatura;

italian, english definisce la lingua di stampa del documento;

lof include la lista delle figure;

lot include la lista delle tabelle;

oneside, twoside definisce l'impaginazione su singola pagina (`oneside`) o su fronte-retro (`twoside`).

Nota bene: non vi è alcuna disposizione od obbligo su tale impostazione (almeno ad oggi!); è, dunque, fortemente consigliata la selezione del formato `twoside`. *D'altronde, ciò che ha valore del tuo manoscritto sono i contenuti, non il volume della tesi una volta stampata!*

B.2.2. Pacchetti di supporto

Nel preambolo sono stati inclusi i pacchetti più utilizzati per la scrittura di un documento di tesi, puoi aggiungere ulteriori pacchetti in base alle tue esigenze.

B.3. Brevissimi esempi sull'inserimento degli oggetti di testo

B.3.1. Equazioni

Come puoi osservare, Eq. B.1 rappresenta un semplice esempio di come scrivere un'equazione numerata:

$$\sigma = \sigma(\varepsilon, \xi_i) \tag{B.1}$$

B.3.2. Tabelle

Le tabelle costituiscono una categoria di oggetti flottanti, ovvero oggetti di testo che possono essere "spostati" dal compilatore al fine di garantire la massima leggibilità e chiarezza dell'intero testo. Tabella B.1 ne è un rapido esempio, e si raccomanda di posizionare la didascalia della tabella in alto rispetto alla stessa.

Table B.1.: Esempio Tabella.

Test	Campioni	Risultati
a	1	2
b	4	5

B.3.3. Figure

Le figure sono il secondo tipo di oggetto flottante che viene frequentemente utilizzato nella scrittura della tesi; di seguito è ripostato un esempio di figura singola (Figura ??), e di sottofigure (Figura ?? e Figura ??).

Nel caso di figure e sottofigure, la didascalia è posta sotto l'immagine a cui si riferisce.

B.4. Inserimento della bibliografia

L'inserimento della bibliografia rappresenta, con buona probabilità, una delle operazioni più criptiche per un utente alle primissime armi con L^AT_EX, soprattutto se si è prossimi alla cosegna del lavoro di tesi. A tal proposito, il main file qui riportato è già organizzato per accogliere qualsiasi file di bibliografia, l'unica operazione richiesta è la creazione della lista dei riferimenti mediante un file in formato .bib. Per rendere tale compito più semplice possibile, il consiglio è quello di impiegare tool e software dedicati come *JabRef* - gratuito - o *Mendeley* (esistono però molti altri References Manager software!). In questo modo dovrai preoccuparti solamente di inserire correttamente campi relativi alle pubblicazioni che vuoi inserire, il software produrrà in uscita il file già formattato per essere letto dal main file.

A questo punto puoi richiamare nel testo la citazione mediante il comando `cite` facendo riferimento alla bibtexkey che identifica il documento desiderato. In questo modo il file .bib si comporta come un database di riferimenti bibliografici: nel testo verranno inclusi solo quelli effettivamente richiamati con il comando `cite`.

Un esempio: come riportato da Von Mises in [?] ... tale teoria è confermata anche dagli studi in [1, ?].

Un ultimo dettaglio: può capitare che bibliografia e references non vengano compilate immediatamente dopo esser state inserite. Per risolvere questo problema basta compilare il main file due volte consecutive: la prima per far "leggere" al

compilatore le modifiche effettuate e stabilire quali sono i documenti invocati nel testo, la seconda per produrre la modifica sul file .pdf di output.

B.5. Esempio di codice in python

Listing B.1: "Codice in Python"

```
import numpy as np

def incmatrix(genl1, genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m, 1), int) #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2), 1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r, c] = np.where(M2 == M1[i, j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

    if M is None:
        M = np.copy(VT)
    else:
        M = np.concatenate((M, VT), 1)

    VT = np.zeros((n*m, 1), int)

    return M
```

B.6. Esempio di codice in C

```
1
2 void main(int n1){
3     int a = 1;
```

Appendice B. Il mio primo capitolo con L^AT_EX

```
4     int b = 2;  
5     return a+b*n1;  
6 }
```

Listing B.2: "Codice in C"

Buona fortuna per il tuo lavoro!

Bibliografia

- [1] R. Hill. A theory of yielding and plastic flow of anisotropic metals. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 193(1033):281–297, 1948.