

UNIVERSITÀ POLITECNICA DELLE MARCHE  
INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

LIBRERIA BN0055



Corso di  
LABORATORIO DI AUTOMAZIONE

Anno accademico 2023-2024

Studenti:  
Marzio Pieretti  
Giovanni Sarti

Professore:  
Andrea Bonci

Dottorando:  
Alessandro Di Biase



Dipartimento di Ingegneria dell'Informazione



# Indice

<b>Introduzione</b>	<b>2</b>
<b>1 Hardware</b>	<b>4</b>
1.1 BNO055 . . . . .	4
1.2 Nucleo h745zi-q . . . . .	6
1.3 Schema dei collegamenti . . . . .	7
<b>2 Software</b>	<b>10</b>
2.1 STM32CubeIDE . . . . .	10
2.2 New project . . . . .	11
2.3 Configurazione USART . . . . .	15
2.4 Configurazione I2C . . . . .	17
2.5 Configurazione interrupt . . . . .	18
2.6 Configurazione main . . . . .	20
<b>3 Funzionalità</b>	<b>21</b>
3.1 Modalità di alimentazione . . . . .	21
3.1.1 Selezione modalità di alimentazione . . . . .	21
Modalità normale . . . . .	22
Modalità risparmio energetico . . . . .	23
Modalità sospensione . . . . .	24
3.2 Modalità operative . . . . .	25
3.2.1 Config mode . . . . .	26
3.2.2 Non-fusion modes . . . . .	27
ACCONLY . . . . .	27
MAGONLY . . . . .	27
GYROONLY . . . . .	27
ACCMAG . . . . .	27
ACCGYRO . . . . .	27
MAGGYRO . . . . .	27
AMG . . . . .	27
3.2.3 Fusion modes . . . . .	29
IMU . . . . .	30
COMPASS . . . . .	30
M4G . . . . .	31
NDOF_FMC_OFF . . . . .	32
NDOF . . . . .	32
3.3 Gestione degli angoli . . . . .	33
3.3.1 Quaternioni . . . . .	33
3.3.2 Angoli di Eulero . . . . .	38

3.3.3	Conversione QUAT to EUL . . . . .	42
3.4	Rimappatura degli assi . . . . .	45
3.4.1	AXIS_MAP_CONFIG . . . . .	45
3.4.2	AXIS_MAP_SIGN . . . . .	45
3.5	Configurazione sensori . . . . .	47
3.5.1	Configurazione predefinita . . . . .	48
3.5.2	Configurazione accelerometro . . . . .	49
3.5.3	Configurazione giroscopio . . . . .	52
3.5.4	Configurazione magnetometro . . . . .	54
3.6	Offsets . . . . .	56
3.6.1	Configurazione dell'Offset dell'Accelerometro . . . . .	56
3.6.2	Configurazione dell'Offset del Magnetometro . . . . .	57
3.6.3	Configurazione dell'Offset del Giroscopio . . . . .	57
3.6.4	Configurazione del Raggio . . . . .	58
3.7	Calibrazione . . . . .	59
3.7.1	Calibrazione accelerometro . . . . .	59
3.7.2	Calibrazione giroscopio . . . . .	59
3.7.3	Calibrazione magnetometro . . . . .	59
3.7.4	Calibrazione del Compasso, Modalità M4G e Modalità NDOF_FMC_OFF . . . . .	60
3.7.5	Calibrazione in Modalità NDOF . . . . .	60
3.7.6	Calibrazione del Soft-Iron (SIC) . . . . .	63
3.7.7	Profilo di Calibrazione . . . . .	64
3.7.8	Test calibrazione manuale . . . . .	65
3.8	Dati in uscita . . . . .	71
3.8.1	Selezione dell'unità di misura . . . . .	71
3.8.2	Formato dei dati di uscita . . . . .	71
3.8.3	UNIT_SEL . . . . .	72
3.8.4	Tassi di Dati in Uscita per la Fusione . . . . .	74
3.9	Interrupt . . . . .	75
3.9.1	Interrupt PIN . . . . .	75
3.10	Test di autoverifica . . . . .	81
3.10.1	POST - auto test al momento dell'accensione . . . . .	81
3.10.2	BIST - auto test integrato . . . . .	82
<b>4</b>	<b>Interfacce digitali</b>	<b>86</b>
4.1	Protocollo I2C . . . . .	87
4.1.1	HAL_Error . . . . .	87
4.2	Protocollo UART . . . . .	88
4.3	HID su I2C . . . . .	89
	<b>Conclusioni e sviluppi futuri</b>	<b>90</b>

## Introduzione

Questo progetto consiste nella creazione di una libreria per l'IMU BNO055, in modo da poter essere utilizzato per acquisire i dati richiesti. Per poter controllare l'IMU è stata utilizzata una board nucleo h745zi-q. Per quanto riguarda la documentazione iniziale, sono stati forniti 2 datasheets per l'IMU e per il microcontrollore le informazioni necessarie sono state trovate nelle slide del corso di Laboratorio di Automazione dell'Università Politecnica delle Marche [1]. Una volta letti i datasheets, per iniziare ad utilizzare il BNO055 ci siamo serviti di una libreria trovata su GitHub dell'utente Ivy Knob [2], che però risulta incompleta, ne abbiamo quindi modificato la struttura, qualche funzione ma soprattutto ne sono state aggiunte di nuove e verranno spiegate in questa relazione.

Prima però si vuole sottolineare che questo documento non è sostitutivo dei datasheets dei componenti utilizzati, i quali possono essere trovati qui:

- Datasheet Bosch del BNO055 [3]
- Datasheet Adafruit del BNO055 [4]

### Struttura del codice:

In questa relazione il codice sarà scritto dentro dei riquadri colorati in **blu** se il codice contenuto deve essere scritto nel main ed esternamente al while, in **arancione** se nel main ed internamente al while, in **verde** se si tratta di codice già scritto nelle librerie e in **rosso** se si tratta di codice scritto in Matlab.

### Contenuto dei capitoli

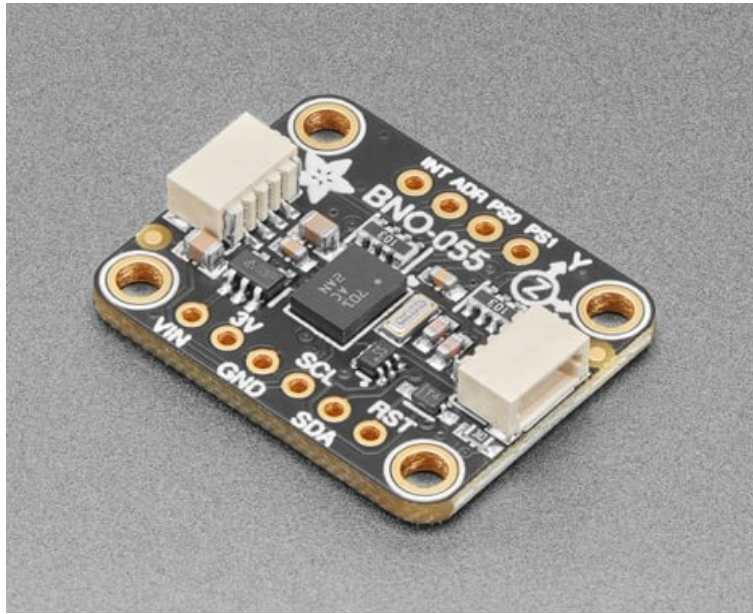
- Hardware: descrive i componenti utilizzati e come collegarli
- Software: descrive le configurazioni effettuate a livello software, dell'USART, dell'I2C, dell'interrupt e del main del progetto.
- Funzionalità: descrive le varie funzionalità disponibili per l'imu e come utilizzarle. Tratta le modalità di alimentazione e operative, fusion e non, viene mostrato il funzionamento dei quaternioni e degli angoli di Eulero, viene poi descritto come rimappare gli assi, come riconfigurare i sensori e i dati in uscita, come abilitare o disabilitare gli interrupt, come effettuare test di autoverifica e come calibrare i sensori.
- Interfacce digitali: aggiunge informazioni utili, come il funzionamento dei protocolli I2C e UART.

# 1 Hardware

Questo capitolo esplora l'hardware fondamentale del nostro progetto: il sensore BNO055 e la scheda Nucleo H745ZI-Q. Il BNO055 offre misurazioni precise di accelerazione, orientamento e rotazione, mentre la scheda Nucleo fornisce una piattaforma avanzata basata sul microcontrollore STM32H745ZI-Q.

## 1.1 BNO055

Il sensore BNO055 è un dispositivo di misurazione inerziale a 9 assi prodotto dalla Bosch Sensortec. Questo componente integra un accelerometro triassiale, un giroscopio triassiale e un magnetometro triassiale, offrendo una soluzione completa per le misurazioni di orientamento e movimento in tre dimensioni.



**Figure 1.1:** *Imu BNO055*

Caratteristiche Principali:

- **Accelerometro Triassiale:** Il BNO055 incorpora un accelerometro triassiale che misura l'accelerazione lineare su ciascun asse. Questa funzionalità è fondamentale per determinare l'orientamento rispetto alla forza di gravità.
- **Giroscopio Triassiale:** Con un giroscopio triassiale, il sensore rileva la velocità angolare intorno a ciascun asse. Ciò consente di monitorare i cambiamenti nell'orientamento con precisione.

- **Magnetometro Triassiale:** Permette al BNO055 di rilevare il campo magnetico terrestre, consentendo così di determinare l'orientamento rispetto al Nord magnetico.
- **Unità di Misura Integrata (IMU):** La fusione di dati provenienti da accelerometro, giroscopio e magnetometro consente di ottenere un'accurata rappresentazione dell'orientamento nello spazio.
- **Sensor Fusion Avanzata:** Il sensore utilizza algoritmi sofisticati di sensor fusion per combinare e filtrare i dati sensoriali, garantendo una misurazione precisa e stabile dell'orientamento.
- **Modalità di Comunicazione Flessibili:** Il BNO055 supporta diverse modalità di comunicazione, inclusi protocolli I2C e UART, per facilitare l'integrazione con una vasta gamma di dispositivi.
- **Calibrazione Automatica:** Il sensore offre una funzione di calibrazione automatica che contribuisce a mantenere la precisione delle misurazioni in vari ambienti.

**Applicazioni:** Il BNO055 trova applicazione in una diversità di settori, tra cui:

- **Robotica:** Utilizzato per il controllo di robot e veicoli autonomi.
- **Realtà Virtuale (VR) e Realtà Aumentata (AR):** Implementato per il tracciamento del movimento in applicazioni immersive.
- **Automotive:** Integrato in veicoli autonomi per la navigazione e il controllo dell'orientamento.
- **Elettronica di Consumo:** Presente in dispositivi indossabili e smartphone per funzioni di rilevamento del movimento.
- **Automazione Industriale:** Utilizzato nei sistemi di automazione per il monitoraggio e il controllo.

## 1.2 Nucleo h745zi-q

La board NUCLEO-144 STM32H745ZI-Q è una piattaforma di sviluppo basata sul microcontrollore STM32H745ZI-Q di STMicroelectronics. Questa board offre un'ampia gamma di funzionalità e una potenza computazionale notevole, ideale per una vasta gamma di applicazioni embedded e IoT.

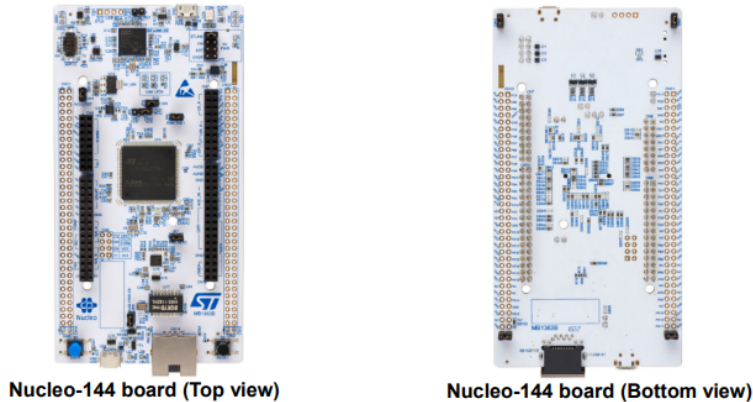


Figure 1.2: Board STM32H745ZI-Q

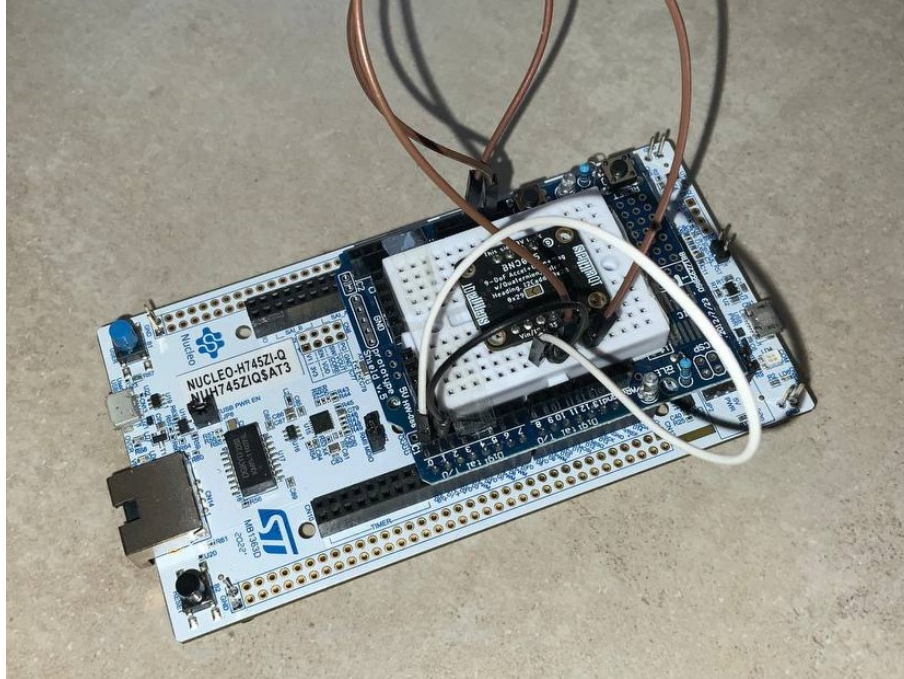
Caratteristiche Principali:

- **Microcontrollore STM32H745ZI-Q:** Cuore della board, questo microcontrollore offre prestazioni elevate grazie al core ARM Cortex-M7 a 480 MHz e ARM Cortex-M4 a 240 MHz. Dispone di ampie risorse di memoria, inclusa la Flash integrata, la RAM e i vari tipi di memoria esterna supportati.
- **Architettura NUCLEO-144:** La board segue lo standard di layout NUCLEO-144, il che significa che offre una vasta gamma di pin GPIO, così come molte risorse aggiuntive come UART, I2C, ADC e altro ancora, rendendo la prototipazione e lo sviluppo molto flessibili.
- **Connettività versatile:** La board include diverse opzioni di connettività, tra cui USB, Ethernet, e una vasta gamma di interfacce seriali. Questo la rende adatta per applicazioni che richiedono comunicazioni veloci e affidabili.
- **Debugger integrato:** La NUCLEO-144 STM32H745ZI-Q è dotata di un debugger integrato ST-Link/V3E che consente la programmazione e il debug del microcontrollore direttamente dalla board, facilitando il processo di sviluppo e riducendo il costo di utilizzo.
- **Compatibilità con l'ecosistema STM32:** Essendo parte della famiglia STM32 di microcontrollori, questa board è completamente compatibile con l'ampio ecosistema di software, strumenti di sviluppo e documentazione offerti da STMicroelectronics, semplificando ulteriormente il processo di sviluppo.



### 1.3 Schema dei collegamenti

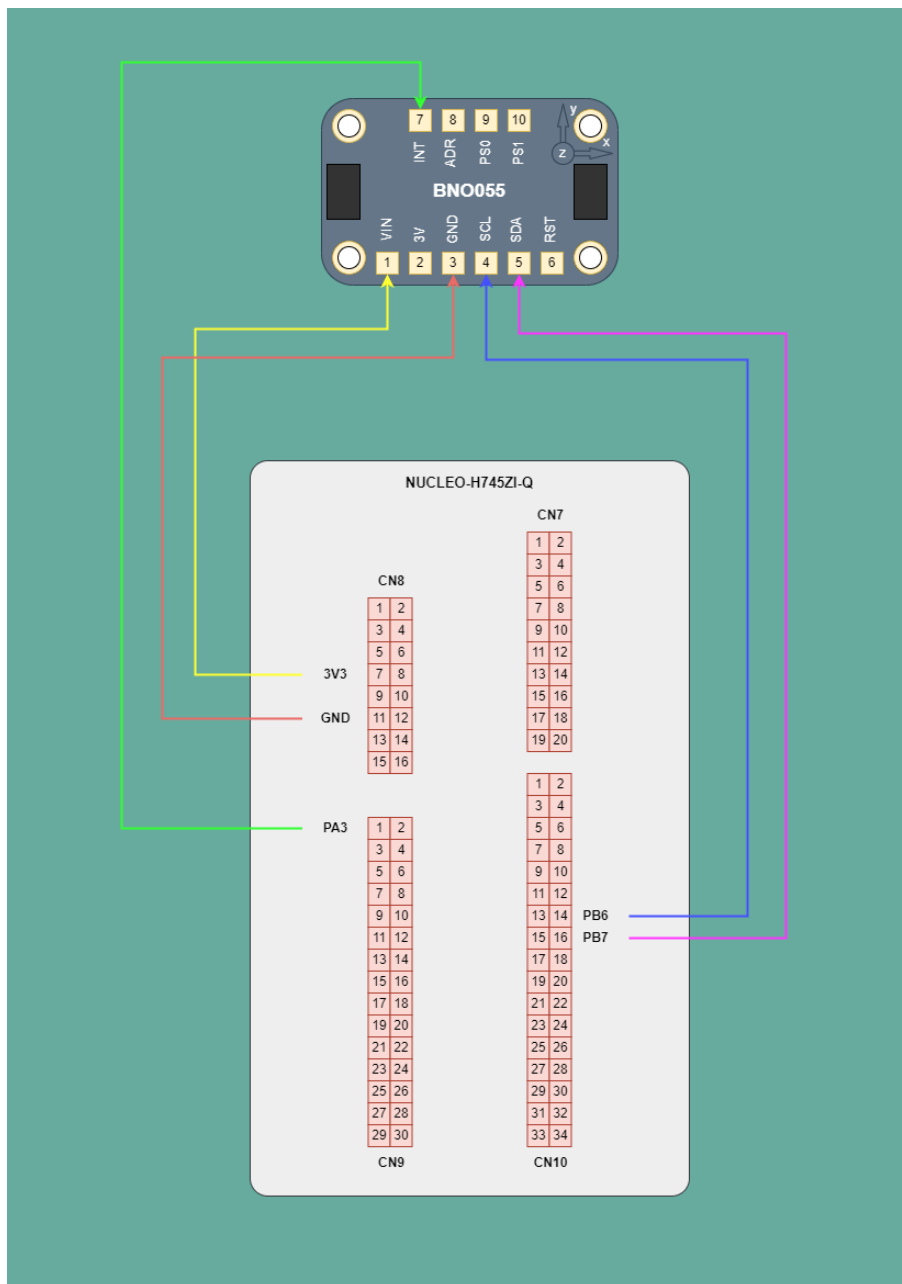
In questa sezione tratteremo del collegamento dei pin tra la board NUCLEO-144 STM32H745ZI-Q e il sensore BNO055. Analizzeremo i dettagli tecnici e le migliori pratiche per garantire una connessione affidabile e efficiente tra questi due dispositivi.



**Figure 1.3:** *Dispositivo completo utilizzato*

Per i test è stato deciso di collegare l'IMU alla board usando come ponte un ProtoShield per Arduino UNO R3 così da avere maggiore stabilità nei movimenti e ridurre una problematica riguardante la precarietà dei collegamenti di cui parleremo nella sezione 4.1.1. Non è però necessario e di conseguenza le connessioni mostrate riguarderanno solo i due componenti principali.

In questa immagine sono stati mostrati solo i pinouts utilizzati, è bene però sapere che la board dispone di alternative che possono essere visualizzate nel suo datasheet di riferimento. La descrizione dei Pin del BNO055 si può trovare nella seguente tabella 1.1.



**Figure 1.4:** *Schema collegamenti*

Inoltre i pin dell'IMU sono descritti dalla seguente tabella:

Pins di alimentazione	VIN [1]: Alimentazione in ingresso da 3.3-5.0V.
	3V [2]: Uscita di 3.3V dal regolatore di tensione lineare integrato sulla scheda; è possibile prelevare fino a circa 50mA secondo necessità.
	GND [3]: Il comune pin GND per alimentazione e logica.
I2C Pins	SCL [4] - I2C clock pin, collegato al I2C clock line. Questo pin può essere utilizzato con logica a 3V o 5V, e su questo pin c'è un pull-up da 10K.
	SDA [5] - I2C data pin, collegato al I2C data line. Questo pin può essere utilizzato con logica a 3V o 5V, e su questo pin c'è un pull-up da 10K.
Altri Pins	RST [6]: Pin di reset hardware. Imposta questo pin basso poi alto per causare un reset sul sensore. Questo pin è compatibile con 5V.
	INT [7]: Il HW interrupt output pin, che può essere configurato per generare un segnale di interruzione quando si verificano determinati eventi come il rilevamento di movimento dall'accelerometro, ecc. Il livello di tensione in uscita è di 3V.
	ADR [8]: Imposta questo pin alto per cambiare l'indirizzo I2C predefinito per il BNO055 se è necessario collegare due IC sulla stessa linea I2C. L'indirizzo predefinito è 0x28. Se questo pin è collegato a 3V, l'indirizzo sarà 0x29.
	PS0 [9] and PS1 [10]: Questi pin possono essere utilizzati per cambiare la modalità del dispositivo (può anche fare HID-I2C e UART) e sono forniti nel caso in cui Bosch fornisca un aggiornamento del firmware in futuro per il microcontrollore ARM Cortex M0 all'interno del sensore. Di norma, dovrebbero essere lasciati scollegati.

**Table 1.1:** *Descrizione PIN BNO055*

## 2 Software

In questa sezione verrà illustrato come configurare il BNO055 a livello software utilizzando STM32IDE. Si è fatto uso di STM32CubeIDE nella versione 1.13.2.

### 2.1 STM32CubeIDE

STM32CubeIDE è un ambiente di sviluppo integrato fornito da STMicroelectronics specificamente progettato per lo sviluppo di software per microcontrollori STM32.

Di seguito, le sue caratteristiche principali:

1. **Basato su Eclipse:** STM32CubeIDE è basato sulla piattaforma Eclipse, il che significa che eredita molte delle sue funzionalità e strumenti di sviluppo già consolidati, come l'editor di codice, il debugger e il sistema di gestione dei plugin.
2. **Integrazione di STM32CubeMX:** STM32CubeIDE è strettamente integrato con STM32CubeMX, un software di configurazione grafica che aiuta gli sviluppatori a generare rapidamente il codice di inizializzazione per i microcontrollori STM32. Questa integrazione semplifica la configurazione hardware e offre un'esperienza di sviluppo più fluida.
3. **Compilatore e toolchain:** STM32CubeIDE include un compilatore GCC per ARM integrato, che consente agli sviluppatori di compilare il codice per i microcontrollori STM32 direttamente dall'IDE. Include anche una toolchain completa per la compilazione, la gestione dei link e la generazione dei file eseguibili.
4. **Debugger avanzato:** STM32CubeIDE offre un debugger avanzato che supporta il debug in-circuit(ICD) e offre funzionalità come il trace degli eventi, il monitoraggio delle variabili in tempo reale, il tracing del codice eseguito e altro ancora. Questo debugger aiuta gli sviluppatori a individuare e risolvere i bug nel codice in modo efficace.
5. **Gestione dei progetti:** STM32CubeIDE offre un sistema di gestione dei progetti flessibile che consente agli sviluppatori di organizzare facilmente i propri progetti e file sorgente. Supporta la gestione di più progetti simultaneamente e offre strumenti per la gestione delle dipendenze e l'integrazione di librerie esterne.
6. **Supporto per le periferiche STM32:** Essendo uno strumento fornito direttamente da STMicroelectronics, STM32CubeIDE offre un ampio supporto per le periferiche STM32, inclusi driver predefiniti, esempi di codice e documentazione integrata. Questo semplifica lo sviluppo di software per le specifiche periferiche STM32.

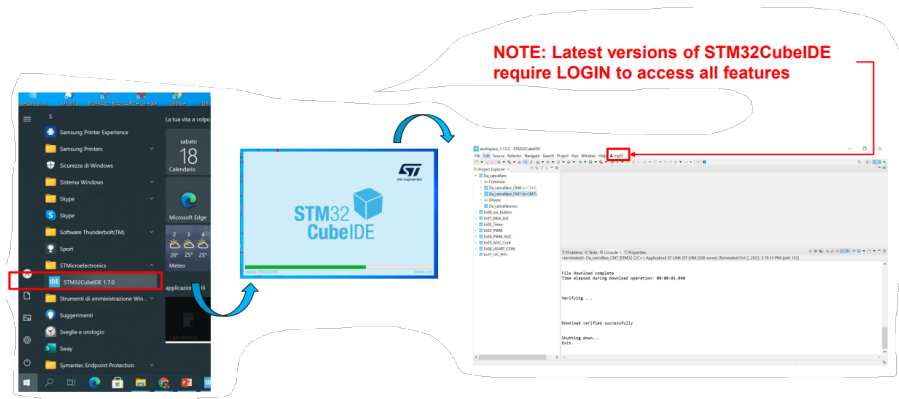
In conclusione, STM32CubeIDE offre agli sviluppatori un ambiente completo per la creazione di software per microcontrollori STM32.

## 2.2 New project

Per creare un nuovo progetto su STM32CubeIDE, dal set-up iniziale alla configurazione del progetto e alla selezione del microcontrollore target, seguire i passaggi fondamentali di seguito.

### Avvio iniziale

Per iniziare, avviare STM32CubeIDE.



**Figure 2.1:** *Passi per avviare STM32CubeIDE*

Il directory di default dei progetti è:

Example - C:\Users\...\STM32CubeIDE\workspace.1.7.0

## Crea un nuovo progetto

Per dare avvio alla creazione di un nuovo progetto, seguire attentamente la sequenza dei seguenti comandi:

→ File → New → *STM32Project*

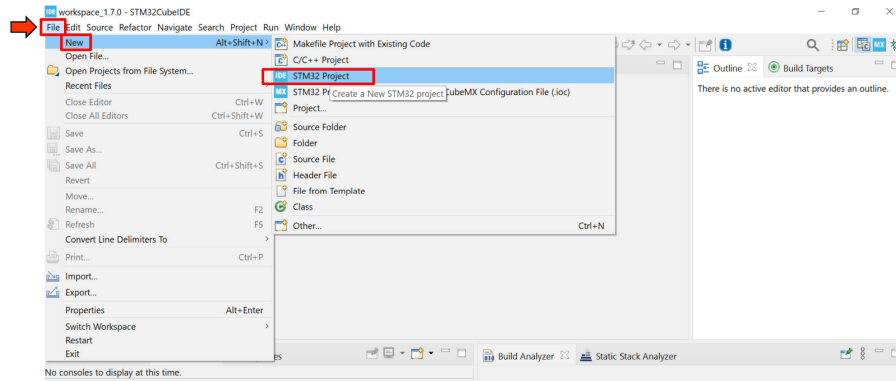


Figure 2.2: Passi per creare un nuovo progetto

In alternativa é possibile iniziare la creazione di un nuovo progetto utilizzando il pulsante 'New' nell'interfaccia utente.

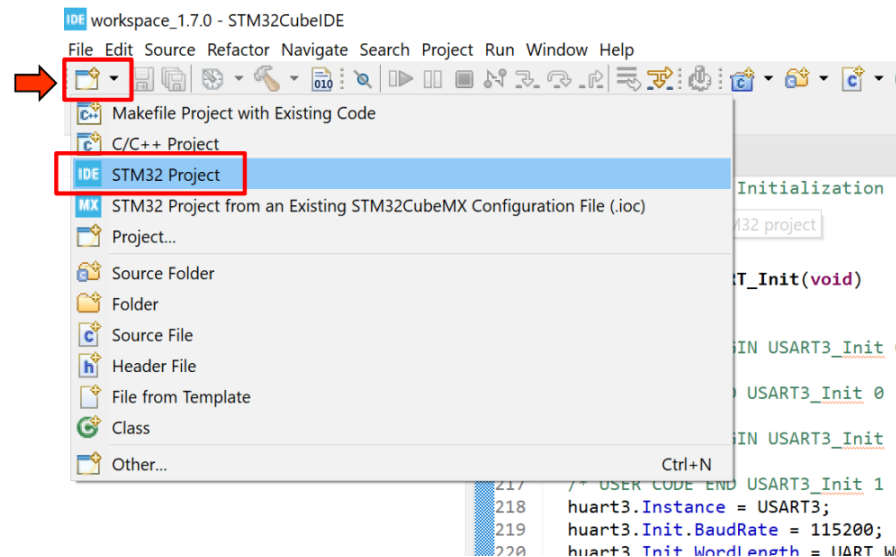


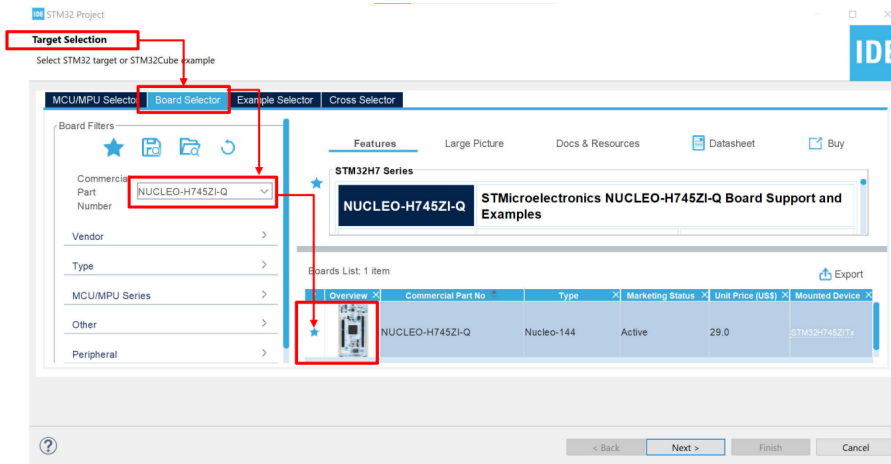
Figure 2.3: Pulsante New

É inoltre possibile usare la scorciatoia 'Alt+Shift+N' per accedere rapidamente alla stessa funzione di creazione del progetto.

## Selezione della Scheda

Dopo aver completato la fase di inizializzazione del nuovo progetto, il processo prevede la selezione del modello di scheda utilizzato. Nel nostro caso, la scheda selezionata è la NUCLEO H745ZI-Q.

→ Target Selection → Board Selector → NUCLEO H745ZI-Q



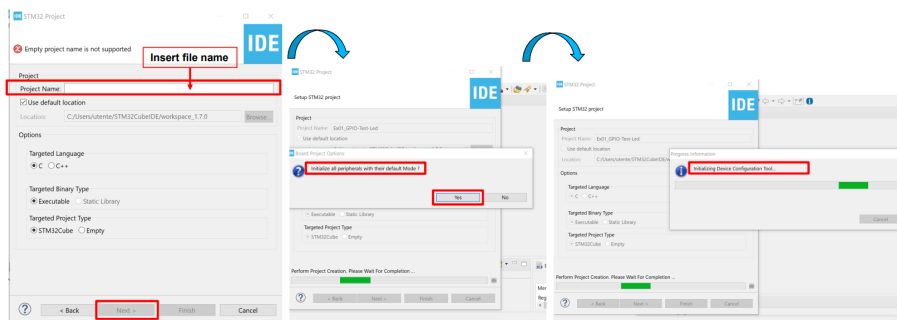
**Figure 2.4:** *Passi per la selezione della Board*

Si consiglia di aggiungere la scheda ai preferiti, facendo clic sull'icona a forma di stella, per semplificare la selezione durante la creazione di nuovi progetti e rendere il processo più veloce.

## Fine della creazione

Adesso, resta solo da assegnare un nome al progetto e avviare l’inizializzazione della configurazione.

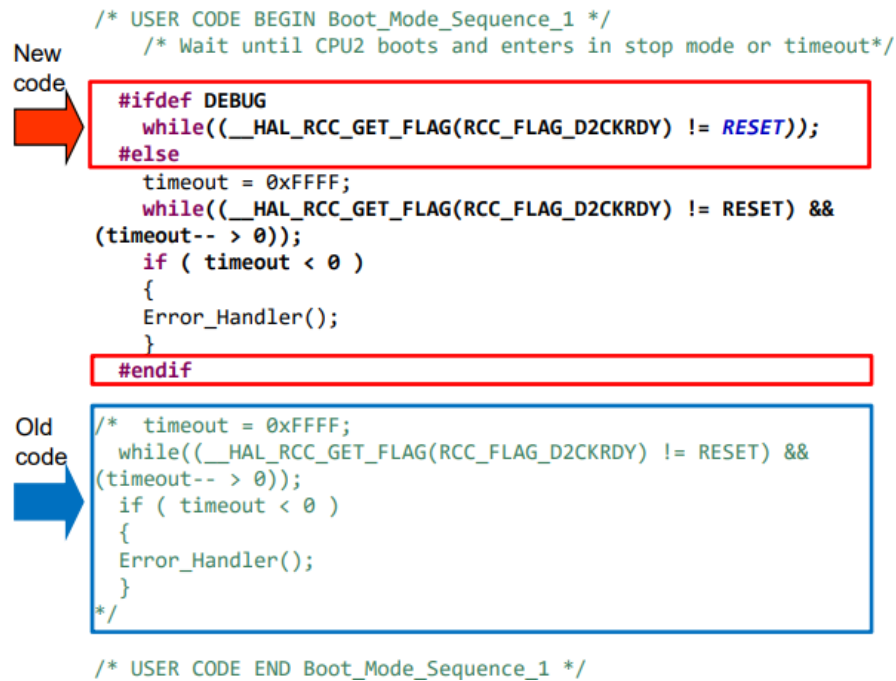
→ Project Name → Initiazlize peripherals default mode (autom) → Initializing device configuration tools (autom)



**Figure 2.5:** *Passi per la conclusione della creazione del nuovo progetto*

## Selezione del core M4

Per il nostro progetto, utilizzeremo il core M4 come core predefinito. Per effettuare questa configurazione, è sufficiente modificare la sezione di codice corrispondente all'interno del file "main.c" del core M7.



```
/* USER CODE BEGIN Boot_Mode_Sequence_1 */
/* Wait until CPU2 boots and enters in stop mode or timeout*/

New code → #ifdef DEBUG
               while((__HAL_RCC_GET_FLAG(RCC_FLAG_D2CKRDY) != RESET));
#else
               timeout = 0xFFFF;
               while((__HAL_RCC_GET_FLAG(RCC_FLAG_D2CKRDY) != RESET) &&
               (timeout-- > 0));
               if ( timeout < 0 )
               {
               Error_Handler();
               }
#endif

Old code → /* timeout = 0xFFFF;
while((__HAL_RCC_GET_FLAG(RCC_FLAG_D2CKRDY) != RESET) &&
(timeout-- > 0));
if ( timeout < 0 )
{
    Error_Handler();
}
*/

/* USER CODE END Boot_Mode_Sequence_1 */
```

**Figure 2.6:** *Modifica del main.c del core M7*



## 2.3 Configurazione USART

Abbiamo scelto di utilizzare la USART3 e come prima cosa la impostiamo nel modo indicato dall'immagine sottostante 2.7. L'USART3 è una periferica di comunicazione seriale sincrona presente nella board nucleo h745zi-q. Ci permette di inviare e ricevere dati a un dispositivo esterno, ad esempio un PC.

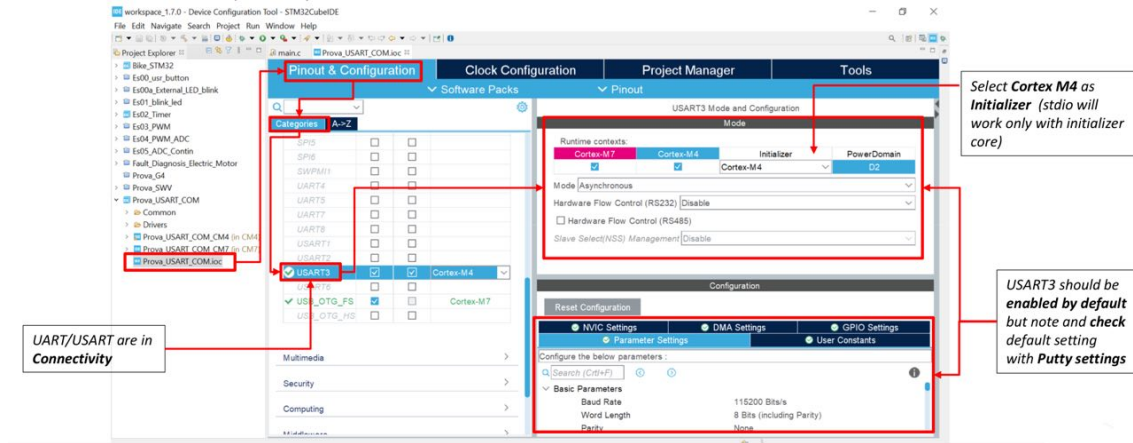


Figure 2.7: Passi per importare l'USART3

Successivamente nella parte finale del main del progetto, precisamente nell' USER CODE 4, scriviamo queste 2 funzioni:

```
int __io_putchar(int ch){
    HAL_UART_Transmit(&huart3, (uint8_t*)&ch, 1, 0xFFFF);
    return ch;
}

int __io_getchar(void){
    uint8_t ch;

    __HAL_UART_CLEAR_OREFLAG(&huart3);

    HAL_UART_Receive(&huart3, (uint8_t *)&ch, 1, 0xFFFF);

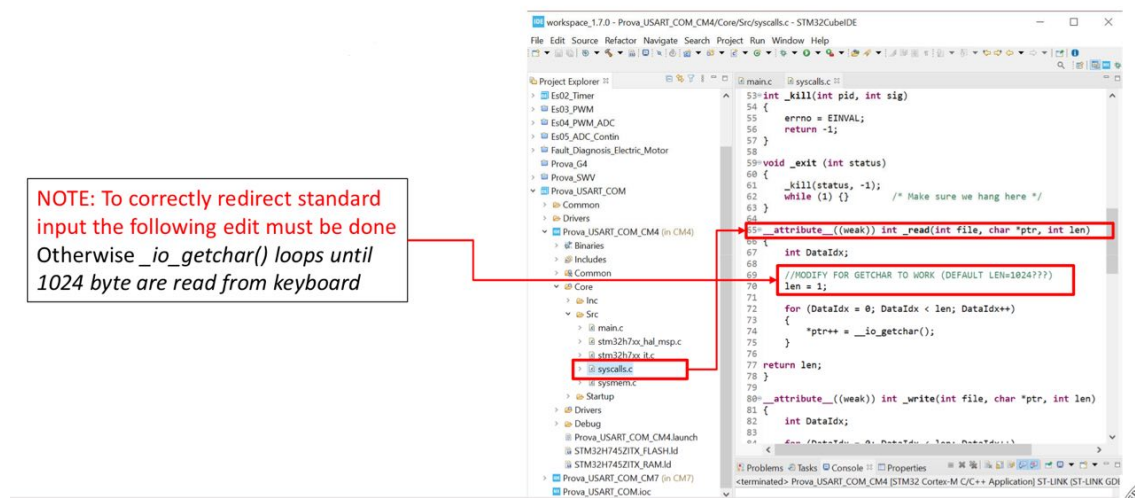
    HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 0xFFFF);

    return ch;
}
```

Listing 2.1: Codice da inserire nel main

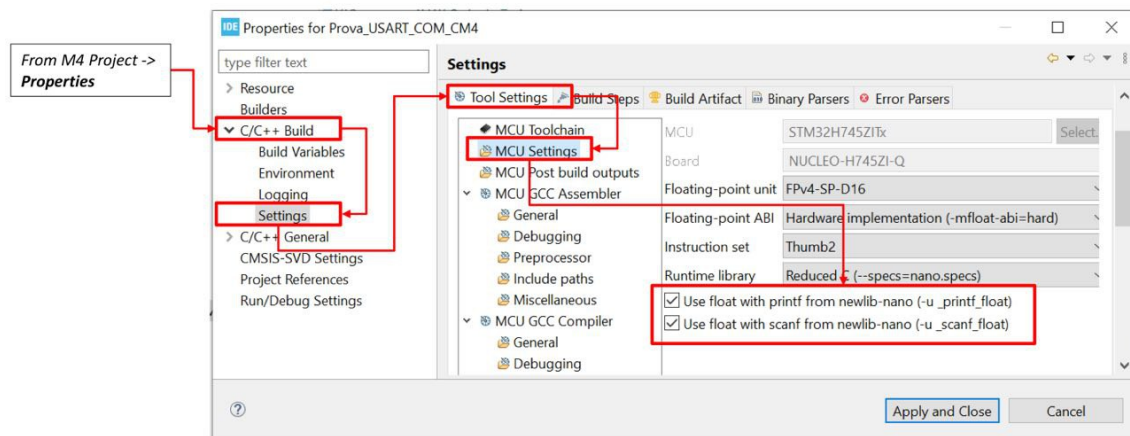
Queste funzioni sono utili per gestire l'input e l'output attraverso la comunicazione seriale UART/USART.

Seguire poi i passaggi riportati nell'immagine successiva:



**Figure 2.8:** Passi per reindirizzare correttamente l'input standard

Il supporto per i numeri in virgola mobile in printf/scanf non è abilitato per impostazione predefinita; per abilitarlo, seguire le seguenti istruzioni contenute nella seguente immagine:



**Figure 2.9:** Passi per impostare i float

## 2.4 Configurazione I2C

Ora passiamo alla configurazione dell'I2C. Si tratta di un protocollo di comunicazione a due fili utilizzato per collegare dispositivi integrati su una stessa scheda. Richiede solo due pin per la comunicazione dati (SDA e SCL) e un pin di alimentazione (VCC). Lo utilizzeremo per la comunicazione tra il BNO055 e il microcontrollore.

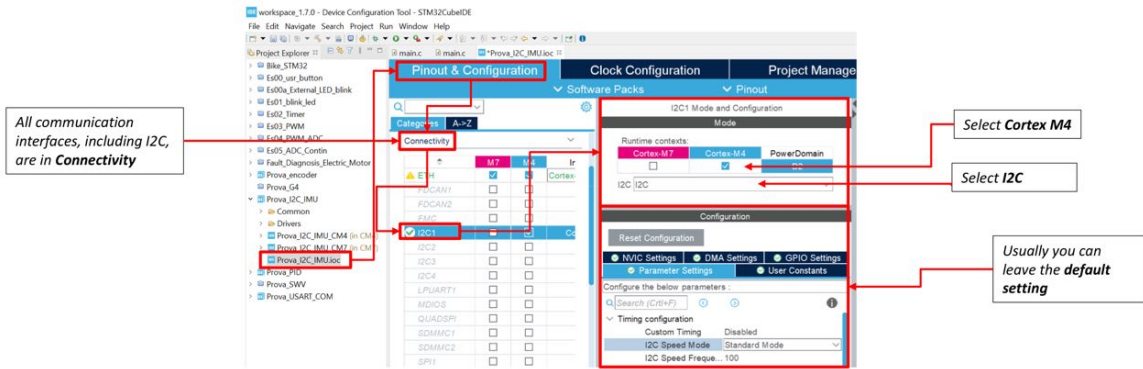


Figure 2.10: Passo 1 per la configurazione dell'I2C

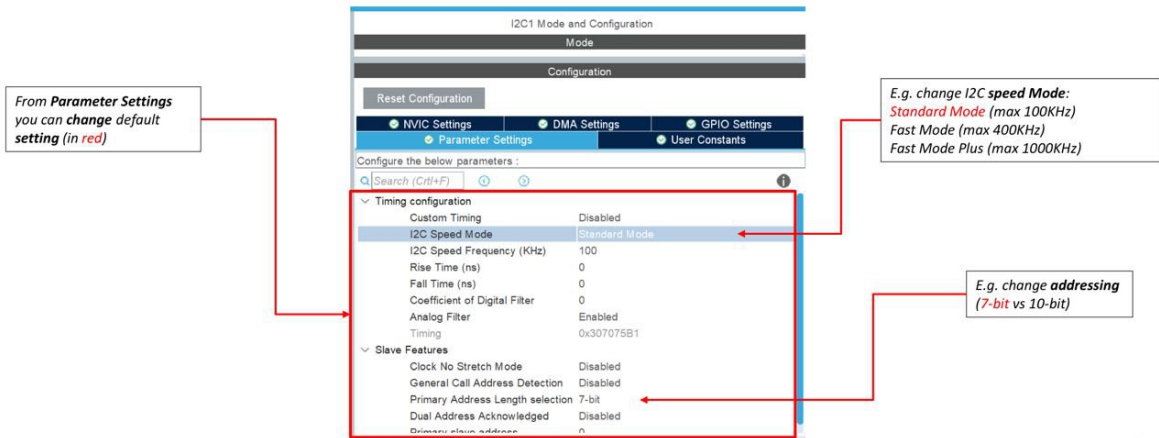


Figure 2.11: Passo 2 per la configurazione dell'I2C

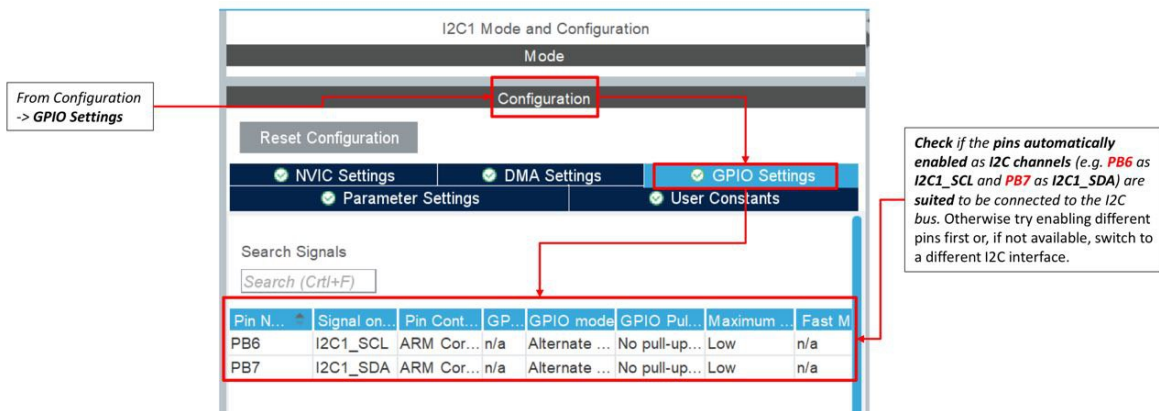


Figure 2.12: Passo 3 per la configurazione dell'I2C

## 2.5 Configurazione interrupt

Questo capitolo si concentra sulla configurazione del pin PA3 come interrupt su una board STM32. Gli interrupt consentono al microcontrollore di rispondere prontamente agli eventi esterni, garantendo una gestione efficiente e tempestiva delle attività.

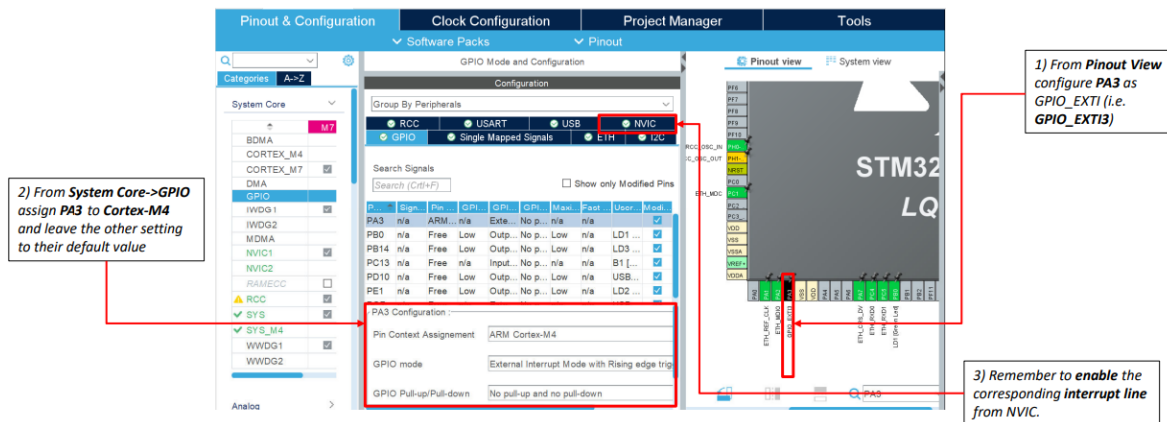


Figure 2.13: Passi per la configurazione dell'interrupt

In successione la parte di codice in cui viene mostrata la funzione Callback che viene eseguita in risposta a un'interruzione esterna sul pin PA3 precedentemente configurato.

Quando l'interruzione è generata dal pin PA3, la funzione imposta la variabile flag (flag\_BNO055\_Data\_Ready) a 1. Ciò indica che l'interruzione è stata causata da un evento che indica che i dati sono pronti ad essere letti dal sensore BNO055.

```

void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin){
    if(GPIO_Pin == GPIO_PIN_3){
        flag_BN0055_Data_Ready = 1;
    }
}

```

Listing 2.2: **Codice da inserire nel main**

Questo è il codice inserito nel ciclo while che reimposta la flag a 0 in attesa della successiva impostazione a 1 con i nuovi dati.

```

if(flag_BN0055_Data_Ready==1){
    flag_BN0055_Data_Ready = 1;  // impostata a 1 per simulare un
    ciclo continuo

    ...

    HAL_Delay(10);
}

```

Listing 2.3: **Codice da inserire nel while**

Durante i nostri test, abbiamo costantemente mantenuto flag\_BN0055\_Data\_Ready nel while impostata su 1, usando così il polling, con l'eccezione dei casi in cui abbiamo dovuto testare gli interrupt, vedi esempio del while 3.9.

## 2.6 Configurazione main

Passiamo alla parte del codice, per far funzionare tutto correttamente è necessario scrivere il codice mostrato sotto nel Main, le librerie è importante che siano in questo specifico ordine e le inizializzazioni invece potrebbero essere state messe in automatico dall'ide, controllare per sicurezza e aggiungere le mancanti.

```
//Librerie
#include "main.h"
#include "string.h"
#include "bno055.h"
#include "stdio.h"
#include "stdint.h"

//...codice...

int main(void)
{
    //Inizializzazioni, alcune potrebbe essere state inserite
    //dall'IDE, nel caso aggiungere le rimanenti
    HAL_Init();

    MX_GPIO_Init();
    MX_I2C1_Init();
    MX_USART3_UART_Init();

    bno055_assignI2C(&hi2c1);
    bno055_setup(); //il BNO055 viene inizializzato con questa
    //funzione

    HAL_Delay(1000);

    //...codice...
}

//...codice...
```

Listing 2.4: Codice da inserire nel main

## 3 Funzionalità

Questa libreria permette di impostare le varie modalità energetiche, scegliendo tra la modalità normale, a risparmio o sospensione. Offre la possibilità di utilizzare tutte le modalità operative, sia non-Fusion che Fusion. Inoltre è possibile rimappare gli assi, cambiare le configurazioni dei vari sensori, cambiare il formato dei dati in uscita, fare dei test di autoverifica e calibrare i sensori in modalità automatica.

### 3.1 Modalità di alimentazione

Il BNO055 offre tre diverse modalità di alimentazione:

1. **Modalità Normale:** In questa modalità, il sensore funziona a piena potenza e fornisce dati completi. È la modalità predefinita all'avvio.
2. **Modalità Risparmio Energetico:** Questa modalità riduce il consumo energetico del sensore ed è utile quando è richiesta una minore potenza di elaborazione. Può essere selezionata scrivendo nel registro `PWR_MODE`.
3. **Modalità Sospensione:** In questa modalità, il sensore è in uno stato di bassissimo consumo energetico e può essere utilizzato quando il dispositivo è inattivo o deve risparmiare energia. Anche questa modalità può essere selezionata scrivendo nel registro `PWR_MODE`.

#### 3.1.1 Selezione modalità di alimentazione

La modalità di alimentazione desiderata può essere selezionata utilizzando la funzione `bno055_setpowerMode(PowerMode mode)` scegliendo tra le tre modalità possibili, la funzione si occupa di scrivere nel registro `PWR_MODE` la modalità scelta

**Table 3.1:** *Selezione delle modalità di alimentazione*

Parametro	Valore (Reg Addr)	Modalità di Alimentazione
PWR_MODE	xxxxxx00b	Modalità Normale
PWR_MODE	xxxxxx01b	Modalità Risparmio Energetico
PWR_MODE	xxxxxx10b	Modalità Sospensione
PWR_MODE	xxxxxx11b	Invalid (Evitare la configurazione)

```

typedef enum {
    NORMAL_MODE = 0x00,
    LOW_POWER_MODE = 0x01,
    SUSPEND_MODE = 0x02
} PowerMode;

// Funzione per impostare la modalita energetica
void bno055_setPowerMode(PowerMode mode) {
    // Leggi il valore corrente di PWR_MODE
    uint8_t currentMode;
    bno055_readData(BNO055_PWR_MODE, &currentMode, 1);

    // Modifica solo i bit relativi alla modalita energetica
    currentMode &= 0xFC; // Azzera i primi due bit
    currentMode |= mode; // Imposta la nuova modalita energetica

    // Scrivi il nuovo valore in PWR_MODE
    bno055_writeData(BNO055_PWR_MODE, currentMode);
}

// Funzione per ottenere la modalita energetica attuale
PowerMode bno055_getPowerMode() {
    // Leggi il valore corrente di PWR_MODE
    uint8_t currentMode;
    bno055_readData(BNO055_PWR_MODE, &currentMode, 1);

    // Estrai la modalita energetica dai primi due bit
    currentMode &= 0x03;

    // Restituisci la modalita energetica come enumerazione PowerMode
    return (PowerMode)currentMode;
}

```

Listing 3.1: Codice della libreria

```

bno055_setPowerMode(NORMAL_MODE); //Inserire la modalita da impostare
PowerMode currentMode = bno055_getPowerMode(); //Metodo per ottenere
    la modalita impostata
printf("Modalita energetica attuale: %d\r\n", currentMode);

```

Listing 3.2: Codice da inserire nel main

## Modalità normale

In modalità normale, tutti i sensori richiesti per la modalità operativa selezionata sono sempre attivi. La mappa dei registri e i dispositivi interni della MCU sono sempre operativi in questa modalità.



## Modalità risparmio energetico

Se viene usato un interrupt e se non viene rilevata alcuna attività (cioè nessun movimento) per una durata configurabile (di default 5 secondi), il BNO055 entra in modalità risparmio energetico. In questa modalità, è attivo solo l'accelerometro. Una volta rilevato un movimento (cioè l'accelerometro segnala un'interruzione per qualsiasi movimento), il sistema si risveglia e entra in modalità normale. Sono possibili le seguenti impostazioni scrivendo nei registri appropriati:

Description	Parameter	Value	Reg Value	Restriction
Entering to sleep: NO Motion Interrupt	Detection Type	No Motion	[ACC_NM_SET] : xxxxxx1b	n/a
		Detection Axis	[ACC_INT_Settings] : bit4-bit2	Shares common bit with Any Motion interrupt axis selection
	Params	Duration	[ACC_NM_SET] : bit6-bit1	n/a
		Threshold	[ACC_NM_THRE] : bit7-bit0	n/a

Description	Parameter	Value	Reg Value
Waking up: Any Motion Interrupt	Detection Type	Detection Axis	[ACC_INT_Settings] : bit4-bit2
	Params	Duration	[ACC_INT_Settings] : bit1-bit0
		Threshold	[ACC_AM_THRES] : bit7-bit0

**Figure 3.1:** Modalità risparmio energetico - interrupt

Le tabelle dei registri è possibile trovarle nel datasheet del BNO055. Inoltre, i pin di interruzione possono essere configurati per fornire un'interruzione hardware all'host. Questa parte è strettamente legata agli interrupt, e come verrà scritto anche in seguito, è un argomento che non abbiamo approfondito nelle configurazioni in quanto l'utilizzo primario per l'imu che abbiamo considerato e testato è attraverso il polling. Per maggiori dettagli sugli interrupt, andare in sezione 3.9.

Il BNO055 è configurato per impostazioni ottimali di default per entrare in modalità sleep e svegliarsi. Per ripristinare questi valori, attivare il reset di sistema impostando il bit RST\_SYS nel registro SYS\_TRIGGER. Ci sono alcune limitazioni per raggiungere le prestazioni della modalità a basso consumo energetico:

- Solo le interruzioni per "No motion" e "Any motion" sono applicabili, mentre le interruzioni per "High-G" e "Slow motion" non sono applicabili in modalità a basso consumo energetico.
- La modalità a basso consumo energetico non è applicabile quando l'accelerometro non è utilizzato.

### Significati:

- No motion - l'interrupt viene attivato quando il sensore non rileva movimenti.
- Any motion - l'interrupt viene attivato quando il sensore rileva qualsiasi movimento.

- High-G - l'interrupt viene attivato quando il sensore rileva l'accelerazione elevata, generalmente utilizzato per rilevare urti.
- Slow motion - l'interrupt viene attivato quando il sensore rileva un movimento lento e fluido.

### **Modalità sospensione**

In modalità sospensione, il sistema è messo in pausa e tutti i sensori e il microcontrollore vengono messi in modalità sleep. Nessun valore nella mappa dei registri verrà aggiornato in questa modalità. Per uscire dalla modalità di sospensione, è necessario cambiare la modalità scrivendo nel registro `PWR_MODE`.

## 3.2 Modalità operative

le modalità disponibili in questo IMU sono 13, dove, ad eccezione della modalità di configurazione, le altre si dividono in modalità fusion e non-fusion. I sensori che operano in ciascuna modalità si possono vedere dalla seguente tabella.

Operating Mode		Available sensor signals			Fusion Data	
		Accel	Mag	Gyro	Relative orientation	Absolute orientation
Non-fusionmodes	CONFIGMODE	-	-	-	-	-
	ACCONLY	X	-	-	-	-
	MAGONLY	-	X	-	-	-
	GYROONLY	-	-	X	-	-
	ACCMAG	X	X	-	-	-
	ACCGYRO	X	-	X	-	-
	MAGGYRO	-	X	X	-	-
Fusion modes	AMG	X	X	X	-	-
	IMU	X	-	X	X	-
	COMPASS	X	X	-	-	X
	M4G	X	X	-	X	-
	NDOF_FMC_OFF	X	X	X	-	X
	NDOF	X	X	X	-	X

Figure 3.2: Modalità operative

```
typedef enum { // BNO055 operation modes
    BNO055_OPERATION_MODE_CONFIG = 0x00,
    // Sensor Mode
    BNO055_OPERATION_MODE_ACCONLY,
    BNO055_OPERATION_MODE_MAGONLY,
    BNO055_OPERATION_MODE_GYRONLY,
    BNO055_OPERATION_MODE_ACCMAG,
    BNO055_OPERATION_MODE_ACCGYRO,
    BNO055_OPERATION_MODE_MAGGYRO,
    BNO055_OPERATION_MODE_AMG, // 0x07
    // Fusion Mode
    BNO055_OPERATION_MODE_IMU,
    BNO055_OPERATION_MODE_COMPASS,
    BNO055_OPERATION_MODE_M4G,
    BNO055_OPERATION_MODE_NDOF_FMC_OFF,
    BNO055_OPERATION_MODE_NDOF // 0x0C
} bno055_opmode_t;
```

Figure 3.3: Modalità operative nella libreria

```

// Ottiene la modalita operativa del sensore BNO055
bno055_opmode_t bno055_getOperationMode() {
    bno055_opmode_t mode;
    bno055_readData(BNO055_OPR_MODE, &mode, 1);
    return mode;
}

// Imposta la modalita operativa del sensore BNO055 e attende un
// ritardo in base alla modalita
void bno055_setOperationMode(bno055_opmode_t mode) {
    bno055_writeData(BNO055_OPR_MODE, mode);
    if (mode == BNO055_OPERATION_MODE_CONFIG) {
        bno055_delay(100);
    } else {
        bno055_delay(80);
    }
}

```

Listing 3.3: Codice nella libreria

### 3.2.1 Config mode

La Config\_Mode del BNO055 è un metodo esclusivo per modificare le impostazioni interne del sensore e personalizzarne il funzionamento per specifiche applicazioni. È importante sottolineare alcuni aspetti chiave di questa modalità:

- **Reset dell'uscita dati e della fusione del sensore:** Quando si entra in Config\_Mode, tutti i dati di output del sensore vengono azzerati e il processo di fusione dei dati sensoriali viene interrotto. Ciò consente di modificare liberamente le impostazioni senza influenzare le letture in corso.
- **Accesso completo ai registri scrivibili:** La Config\_Mode è l'unica modalità che permette la modifica di tutti i registri scrivibili del BNO055. Questo livello di accesso è necessario per configurare completamente il sensore in base alle proprie esigenze.
- **Limitazioni:** Esistono alcune eccezioni a questa regola di accesso completo. I registri di interrupt (INT e INT\_MSK) e il registro della modalità operativa (OPR\_MODE) non possono essere modificati in Config\_Mode. Fortunatamente, questi registri possono essere modificati in qualsiasi altra modalità operativa.
- **Modalità operativa predefinita:** La Config\_Mode è impostata di default all'accensione del sensore o dopo un RESET. Ciò significa che per iniziare a leggere i dati del sensore, è necessario selezionare una modalità operativa diversa dalla configurazione.

```

void bno055_setOperationModeConfig() {
    bno055_setOperationMode(BNO055_OPERATION_MODE_CONFIG);
}

```

Listing 3.4: Codice nella libreria

### 3.2.2 Non-fusion modes

Le modalità Non-Fusion del BNO055 consentono di utilizzare il sensore in modo semplificato, leggendo i dati da singoli sensori o da combinazioni di essi, disabilitando invece la fusione dei dati che avviene normalmente. Questo permette di ottimizzare il consumo energetico in base alle necessità dell'applicazione.

#### **ACCONLY**

In questa modalità viene attivato solo l'accelerometro, mentre magnetometro e giroscopio vengono sospesi per ridurre il consumo energetico. In pratica, il BNO055 si comporta come un sensore di accelerazione stand-alone.

#### **MAGONLY**

In modalità MAGONLY, il BNO055 si comporta come un magnetometro stand-alone, con accelerometro e giroscopio disabilitati.

#### **GYROONLY**

In modalità GYROONLY, viene attivato solo il giroscopio, mentre accelerometro e magnetometro vengono sospesi per risparmiare energia.

#### **ACCMAG**

Vengono attivati sia l'accelerometro che il magnetometro, consentendo la lettura dei dati da entrambi i sensori.

#### **ACCGYRO**

In questa modalità, sono attivi sia l'accelerometro che il giroscopio, permettendo di leggere i dati da entrambi.

#### **MAGGYRO**

Vengono attivati sia il magnetometro che il giroscopio, consentendo la lettura dei dati da entrambi i sensori.

#### **AMG**

In questa modalità vengono attivati tutti e tre i sensori: accelerometro, magnetometro e giroscopio.

La scelta della modalità Non-Fusion dipende dall'applicazione e dai dati che si vogliono acquisire. Se, ad esempio, si è interessati solo all'accelerazione, la modalità ACCONLY è quella che consuma meno energia. Al contrario, la modalità AMG attiva tutti i sensori ed è quella che consuma di più, ma fornisce anche il set completo di dati.

```

// NON FUSION MODE

void bno055_setOperationModeACCONLY(){
    bno055_setOperationMode(BNO055_OPERATION_MODE_ACCONLY);
}

void bno055_setOperationModeMAGONLY(){
    bno055_setOperationMode(BNO055_OPERATION_MODE_MAGONLY);
}

void bno055_setOperationModeGYRONLY() {
    bno055_setOperationMode(BNO055_OPERATION_MODE_GYRONLY);
}

void bno055_setOperationModeACCMAG(){
    bno055_setOperationMode(BNO055_OPERATION_MODE_ACCMAG);
}

void bno055_setOperationModeACCGYRO(){
    bno055_setOperationMode(BNO055_OPERATION_MODE_ACCGYRO);
}

void bno055_setOperationModeMAGGYRO(){
    bno055_setOperationMode(BNO055_OPERATION_MODE_MAGGYRO);
}

void bno055_setOperationModeAMG(){
    bno055_setOperationMode(BNO055_OPERATION_MODE_AMG);
}

```

Listing 3.5: Codice nella libreria

```

bno055_vector_t gyr;
bno055_vector_t acc;
bno055_vector_t mag;
bno055_setOperationModeAMG();

```

Listing 3.6: Esempio di codice da inserire nel main

```

gyr = bno055_getVectorGyroscope();
acc = bno055_getVectorAccelerometer();
mag = bno055_getVectorMagnetometer();

printf("GYR - x: %+2.2f | y: %+2.2f | z: %+2.2f\r\n", gyr.x,
    gyr.y, gyr.z);
printf("ACC - x: %+2.2f | y: %+2.2f | z: %+2.2f\r\n", acc.x,
    acc.y, acc.z);
printf("MAG - x: %+2.2f | y: %+2.2f | z: %+2.2f\r\n", mag.x,
    mag.y, mag.z);

```

Listing 3.7: Esempio di codice da inserire nel while

```
GYR-x: +0.00 | y: -0.38 | z: +0.06  
ACC-x: +0.09 | y: -0.45 | z: -9.72  
MAG-x: -7.81 | y: +1.44 | z: +47.62  
GYR-x: +0.12 | y: +0.12 | z: +0.00  
ACC-x: +0.07 | y: -0.48 | z: -9.70  
MAG-x: -8.50 | y: +0.31 | z: +48.44  
GYR-x: +0.06 | y: -0.06 | z: +0.00  
ACC-x: +0.15 | y: -0.55 | z: -9.73  
MAG-x: -8.56 | y: +0.31 | z: +47.25
```

**Figure 3.4:** *Output dell'esempio mostrato*

### 3.2.3 Fusion modes

Le modalità di fusione del BNO055 combinano i dati provenienti da più sensori (accelerometro, magnetometro e giroscopio) per calcolare l'orientamento del dispositivo nello spazio. Esistono due tipi di orientamento: relativo (non assoluto) e assoluto.

#### Orientamento assoluto

Indica l'orientamento del sensore rispetto alla Terra e al suo campo magnetico, fornendo quindi la direzione del polo nord magnetico.

#### Orientamento relativo (non assoluto)

L'orientamento del sensore può variare a seconda della posizione iniziale in cui viene posizionato. Tutte le modalità di fusione forniscono l'orientamento tramite dati quaternionici o angoli di Eulero (roll, pitch e yaw).

#### Accelerazione lineare

L'accelerometro misura sia l'accelerazione dovuta alla gravità terrestre sia le accelerazioni dovute ai movimenti del dispositivo. Le modalità di fusione permettono di separare queste due fonti di accelerazione, fornendo quindi valori distinti per l'accelerazione lineare e il vettore gravitazionale.

#### Note importanti

- Tutte le modalità di fusione forniscono l'orientamento del sensore sotto forma di dati quaternionici o angoli di Eulero, approfondiremo questa parte nella sezione apposita 3.3.
- La calibrazione automatica in background di tutti i sensori è una caratteristica intrinseca dell'algoritmo di fusione e non può essere disabilitata.
- L'algoritmo di fusione utilizza i dati dell'accelerometro per compensare la deriva del giroscopio nel tempo. Quando il dispositivo è in movimento, i dati dell'accelerometro

vengono temporaneamente ignorati e la fusione si basa sul giroscopio per pitch e roll. Se i dati dell'accelerometro non possono essere utilizzati per un lungo periodo (ad esempio a causa di vibrazioni o movimenti costanti), i valori di pitch e roll potrebbero subire una deriva nel tempo. Allo stesso modo, se vengono rilevati dati distorti dal magnetometro, l'algoritmo li ignorerà automaticamente e ciò potrebbe causare una deriva del valore di heading (direzione) nel tempo (come in modalità IMU).

- L'algoritmo di fusione è stato principalmente progettato per tracciare il movimento umano. Se il dispositivo è sottoposto a grandi accelerazioni per un periodo prolungato (ad esempio in un veicolo che curva ad alta velocità o frena su una lunga distanza), potrebbe interpretare erroneamente questa forte accelerazione come il vettore gravitazionale. Per applicazioni in cui il sensore potrebbe essere esposto a velocità significative, si consiglia di testare il dispositivo per lo specifico caso d'uso.

## IMU

Calcola l'orientamento relativo del BNO055 nello spazio utilizzando i dati dell'accelerometro e del giroscopio. Questa modalità è veloce e fornisce un alto tasso di aggiornamento dei dati. IMU è l'unica modalità fusion che non si calibra automaticamente.

## COMPASS

Il modo COMPASS è progettato per misurare il campo magnetico terrestre e calcolare la direzione geografica. Il campo magnetico terrestre è un vettore con tre componenti: le componenti orizzontali x e y e la componente verticale z. Questo campo magnetico terrestre varia in base alla posizione sulla Terra e alla presenza naturale di minerali ferrosi nell'ambiente circostante.

Per calcolare l'orientamento o la direzione della bussola, il BNO055 utilizza solo le componenti orizzontali x e y del campo magnetico terrestre. Tuttavia, affinché queste misurazioni siano accurate, è necessario considerare anche la direzione del vettore di gravità. In altre parole, l'orientamento può essere calcolato correttamente solo quando vengono presi in considerazione sia il campo magnetico che l'accelerometro per determinare la direzione della gravità.

È importante notare che la precisione delle misurazioni dell'orientamento dipende dalla stabilità del campo magnetico circostante. Poiché il campo magnetico terrestre è generalmente molto più debole rispetto ai campi magnetici generati da dispositivi elettronici e altre sorgenti, è necessaria una calibrazione accurata per compensare eventuali disturbi magnetici esterni e interni.

La modalità COMPASS è utile per calcolare la direzione geografica, ma è importante garantire una corretta calibrazione per ottenere misurazioni accurate e affidabili. La stabilità del campo magnetico e la calibrazione sono fondamentali per garantire il corretto funzionamento di questa modalità.



```
float calculateHeading(float magX, float magY)
{
    float heading = atan2(magY, magX) * 180 / M_PI;

    if (heading < 0)
    {
        heading += 360;
    }

    return heading;
}
```

Listing 3.8: Codice nella libreria

```
bno055_vector_t acc;
bno055_vector_t mag;

bno055_setOperationModeCOMPASS();
```

Listing 3.9: Codice da inserire nel main

```
bno055_calibration_state_t cal = bno055_getCalibrationState();

acc = bno055_getVectorAccelerometer();
mag = bno055_getVectorMagnetometer();

float heading = calculateHeading(mag.x, mag.y);

printf("Orientamento magnetico: %f, cal - acc: %d | mag: %d\r\n",
    heading, cal.accel, cal.mag);
```

Listing 3.10: Codice nel while

Dato che la modalità COMPASS si autocalibra abbiamo deciso di provarla direttamente.

NOTA: Il sensore ha bisogno di stare in piano per poter misurare correttamente.

NOTA: Visto che anche con la NDOF si potrebbero ottenere gli stessi risultati dato che anch'essa utilizza magnetometro e giroscopio (più accelerometro) crediamo che l'utilità di questa modalità nei confronti della nine-degrees-of-freedom possa essere un minor consumo dovuto dall'assenza dell'accelerometro e forse un algoritmo che funziona meglio essendo specifico, ma di quest'ultima affermazione non abbiamo avuto riscontro dalle prove.

## M4G

Simile alla modalità IMU, ma utilizza il magnetometro per rilevare la rotazione invece del giroscopio. Consuma meno energia rispetto alla modalità IMU ma la precisione di misurazione dipende dalla stabilità del campo magnetico circostante. In questa modalità la calibrazione del magnetometro non è necessaria e non è disponibile.

## **NDOF\_FMC\_OFF**

Identica alla modalità NDOF ma con la calibrazione magnetometrica veloce disabilitata.

## **NDOF**

La Modalità NDOF (Nine Degrees of Freedom) è una modalità di fusione dei dati di orientamento a 9 gradi di libertà in cui i dati di accelerometro, giroscopio e magnetometro sono usati in un algoritmo per ricavare l'orientamento assoluto. Questo calcolo viene eseguito utilizzando sia quaternioni che angoli di Eulero, fornendo due rappresentazioni alternative dell'orientamento.

Unendo questi tre sensori, si ottiene un calcolo veloce che si traduce in un'elevata frequenza di output dei dati 3.16 e una notevole robustezza contro le distorsioni del campo magnetico. La calibrazione rapida del magnetometro, denominata "Fast Magnetometer Calibration," è attivata in questa modalità, consentendo una calibrazione rapida del magnetometro e una maggiore precisione nell'output dei dati.

L'utilizzo della Modalità NDOF è particolarmente vantaggioso quando si richiede un'alta precisione nei dati di orientamento e si desidera una risposta rapida e robusta ai cambiamenti nell'ambiente magnetico circostante. È importante notare che, a causa della fusione di dati da sensori multipli e della calibrazione del magnetometro, il consumo di energia attuale è leggermente superiore rispetto ad altre modalità di fusione dei dati.

La modalità NDOF fornisce due tipi di output principali: quaternioni, che rappresentano l'orientamento in modo compatto e preciso, e angoli di Eulero, che offrono una rappresentazione più intuitiva dell'orientamento in termini di roll, pitch e yaw. Questa versatilità nella rappresentazione dell'orientamento consente agli utenti di scegliere la forma più adatta alle loro esigenze applicative.

## 3.3 Gestione degli angoli

Questa sezione si focalizzerà sui quaternioni, sugli angoli di Eulero e sulla conversione tra i due sistemi di rappresentazione dell'orientamento. Esploreremo le caratteristiche di entrambi i metodi e forniremo algoritmi dettagliati per la conversione tra di essi, offrendo una panoramica completa per comprendere e gestire l'orientamento nello spazio tridimensionale.

### 3.3.1 Quaternioni

Un quaternione è una rappresentazione matematica dell'orientamento tridimensionale che utilizza quattro numeri reali:  $(w, x, y, z)$ . Il componente "w" è una parte reale, mentre "x", "y" e "z" sono parti immaginarie. L'orientamento è definito dalla relazione tra questi quattro numeri.

La rotazione di un oggetto può essere rappresentata da un quaternione unitario, ossia un quaternione in cui la somma dei quadrati dei componenti è uguale a 1:  $w^2 + x^2 + y^2 + z^2 = 1$ . La rotazione può essere applicata moltiplicando due quaternioni unitari, che rappresentano le rotazioni dell'oggetto in modo sequenziale. I quaternioni sono spesso preferiti in applicazioni in cui è necessaria una rappresentazione compatta e continua dell'orientamento, come nei giochi o nella robotica.

```
bno055_vector_t quat;  
bno055_setOperationModeNDOF();
```

Listing 3.11: Codice da inserire nel main

```
bno055_calibration_state_t cal = bno055_getCalibrationState();  
quat = bno055_getVectorQuaternion();  
printf("QUAT - x: %+2.2f | y: %+2.2f | z: %+2.2f | w: %+2.2f |  
      %+2.2d\r\n", quat.x, quat.y, quat.z, quat.w, cal.sys);  
HAL_Delay(10);
```

Listing 3.12: Codice nel while

Dato che la modalità NDOF si autocalibra abbiamo deciso di inserire la funzione per mostrare lo stato di calibrazione durante il test (ultimo dato di stampa, +03).

Per riuscire a leggere i dati e capire se fossero affidabili, e quanto, abbiamo dovuto creare un codice Matlab per vedere l'orientamento di un oggetto in uno spazio 3d. L'algoritmo fusion del Bno055 fornisce i dati dei quaternioni già normalizzati, quindi nel codice Matlab la parte relativa alla normalizzazione è stata commentata.

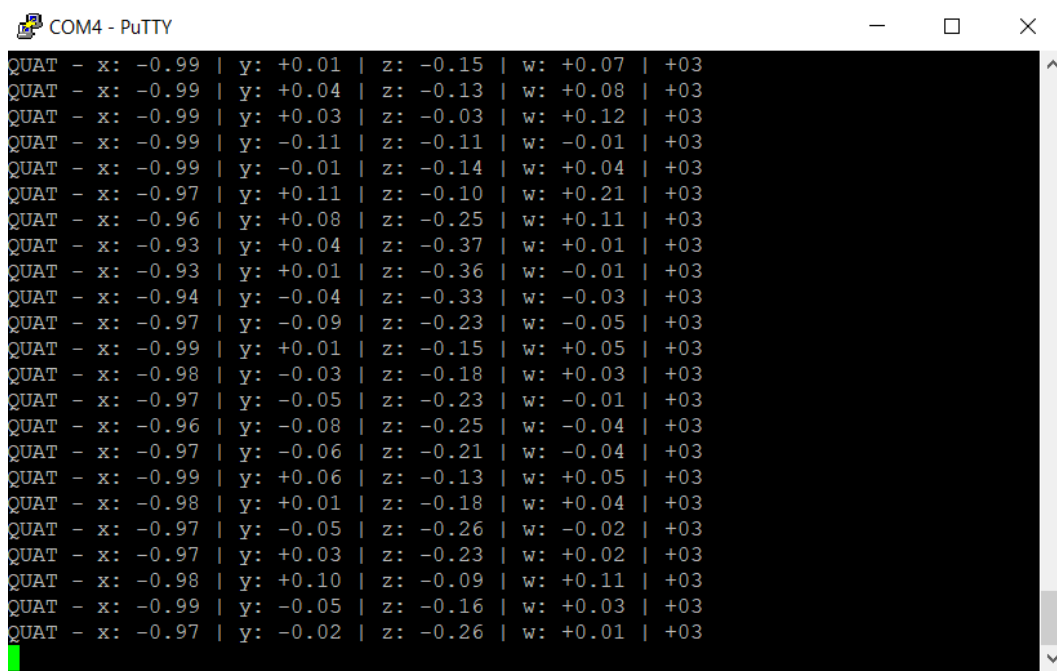


Figure 3.5: Output del codice mostrato

MatLab

```
% Configura la porta seriale
s = serialport('COM4', 115200); % Assicurati che 'COM4' corrisponda
    alla tua porta seriale e che il baud rate sia corretto
try
    % Inizializza la visualizzazione 3D con il parallelepipedo
    figure;
    %Definisci le coordinate del parallelepipedo
    cubeCoords = [
        0 0.5 0.4 0.4 0.5 0.4 0.4 0.5;
        0 0 -0.1 0.1 0 0 0 0;
        0 0 0 0 0 0.1 -0.1 0;
    ];

    cubePlot = plot3(cubeCoords(1, :), cubeCoords(2, :),
        cubeCoords(3, :), 'c-', 'LineWidth', 3);

    xlabel('X');
    ylabel('Y');
    zlabel('Z');
    title('Orientamento con parallelepipedo 3D');
    grid on;
    % Imposta i limiti dell'asse
    axis([-1 1 -1 1 -1 1]); % Modifica i valori in base alle
        dimensioni desiderate
    % Loop infinito per leggere dati
    while true
        % Legge una riga di dati dalla porta seriale
        data = readline(s);
```

```

% Aggiungi una stampa di debug per visualizzare la stringa
ricevuta
disp(['Dati ricevuti: ' data]);
% Estrai i componenti del quaternione e il valore aggiuntivo
cal.sys dalla stringa
matches = sscanf(data, 'QUAT - x: %f | y: %f | z: %f | w: %f
| %d\r\n');
% Verifica se il formato della stringa e corretto
if numel(matches) == 5
    % Assegna i valori estratti ai componenti del
    quaternione e al valore cal.sys
    qw = matches(4);
    qx = matches(1);
    qy = matches(2);
    qz = matches(3);
    calibrationValue = matches(5);
    % Normalizza il quaternione (l'algoritmo di BND055 li
    calcola gia normalizzati)
    % qNorm = sqrt(qw^2 + qx^2 + qy^2 + qz^2);
    % qw = qw / qNorm;
    % qx = qx / qNorm;
    % qy = qy / qNorm;
    % qz = qz / qNorm;
    % Calcola la matrice di rotazione dal quaternione
    rotationMatrix = [
        1 - 2*(qy^2 + qz^2), 2*(qx*qy - qw*qz), 2*(qx*qz +
        qw*qy);
        2*(qx*qy + qw*qz), 1 - 2*(qx^2 + qz^2), 2*(qy*qz -
        qw*qx);
        2*(qx*qz - qw*qy), 2*(qy*qz + qw*qx), 1 - 2*(qx^2 +
        qy^2)
    ];
    % Ruota le coordinate del parallelepipedo utilizzando la
    matrice di rotazione
    rotatedCubeCoords = rotationMatrix * cubeCoords;
    % Aggiorna la posizione del parallelepipedo nella
    visualizzazione 3D
    set(cubePlot, 'XData', rotatedCubeCoords(1, :), 'YData',
    rotatedCubeCoords(2, :), 'ZData',
    rotatedCubeCoords(3, :));

    drawnow; % Aggiorna la visualizzazione

else
    disp('Formato della stringa non valido.');
```

```

end
% Aggiungi una pausa (tempo di attesa) tra le letture
pause(0.01); % 10 millisecondi (puoi regolare il valore in
base alle tue esigenze)
end

catch exception
    % Gestione degli errori
    disp(['Errore: ' exception.message]);
end
% Chiudi la porta seriale alla fine
clear s;

```

Listing 3.13: Codice in Matlab

```

cubeCoords = [
    -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 0.5 0.5 0.5
      0.5 0.5 0.5 0.5 -0.5 -0.5;
    -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5
      0.5 -0.5 0.5 0.5 0.5;
    -0.5 -0.5 -0.5 -0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 -0.5 0.5
      -0.5 0.5 0.5 0.5 -0.5;
];

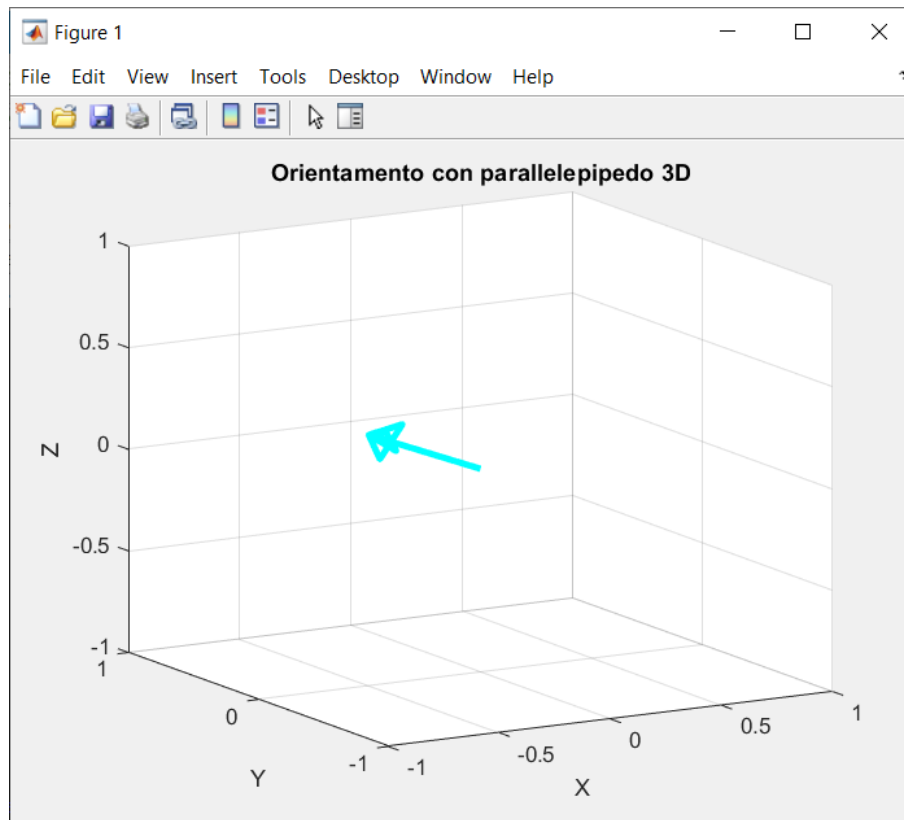
```

Listing 3.14: **Codice in Matlab**

Il codice creato in Matlab è fatto in modo tale da prendere l'esatta stampa dall'IDE di STM, infatti la stringa del printf in STM32Cubeide è uguale alla stringa dello sscanf di Matlab, (il +2.2 è solo uno specificatore di formato, non conta), nel caso si cambiasse il printf dell'IDE di STM va cambiata anche la sscanf nel codice Matlab. La rotazione della matrice è stata presa da Wikipedia [5].

$$R = \begin{bmatrix} 1 - 2s(q^2 + q^2) & 2s(q^2 q^2 - q^2 q^2) & 2s(q^2 q^2 + q^2 q^2) \\ 2s(q^2 q^2 + q^2 q^2) & 1 - 2s(q^2 + q^2) & 2s(q^2 q^2 - q^2 q^2) \\ 2s(q^2 q^2 - q^2 q^2) & 2s(q^2 q^2 + q^2 q^2) & 1 - 2s(q^2 + q^2) \end{bmatrix} \quad (3.1)$$

Inoltre nel codice ci sono 2 figure tra cui poter scegliere, una freccia e un parallelepipedo, sostituire la funzione cubeCoords a seconda di quella che si vuole utilizzare.



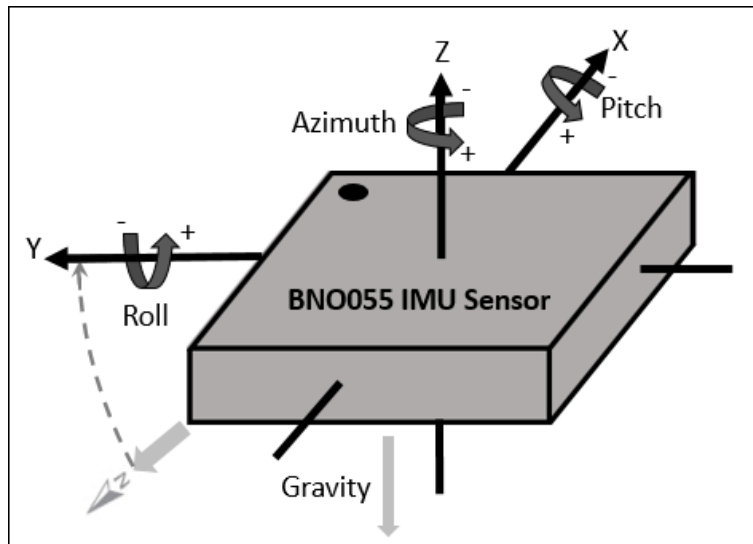
**Figure 3.6:** *Figure ottenuta con il codice mostrato sopra*

Abbiamo fatto dei test e abbiamo riscontrato essere molto preciso una volta che si sia calibrato del tutto ( $CAL = 3$ ).

Attenzione: Una volta che la calibrazione raggiunge il livello 3 non significa che tutti i sensori siano davvero calibrati, almeno in questa modalità, è possibile infatti siano calibrati solo il giroscopio e il magnetometro (i più importanti per i quaternioni), il risultato è comunque molto preciso ma se si vuole migliorare si consiglia di calibrare anche l'accelerometro.

### 3.3.2 Angoli di Eulero

Gli angoli di Eulero sono un'altra rappresentazione comune dell'orientamento. Questa rappresentazione utilizza tre angoli separati per descrivere la rotazione lungo gli assi principali. Gli angoli di Eulero sono generalmente indicati come roll ( $\phi$ ), pitch ( $\theta$ ) e yaw ( $\psi$ ).



**Figure 3.7:** *Orientamento assi BNO055*

Roll ( $\phi$ ): La rotazione intorno all'asse Y. Pitch ( $\theta$ ): La rotazione intorno all'asse X. Yaw ( $\psi$ ): La rotazione intorno all'asse Z. Gli angoli di Eulero forniscono una rappresentazione intuitiva dell'orientamento, ma possono essere soggetti a problemi di singolarità e ambiguità in alcune situazioni, noti come problemi di "blocco giroscopico". Tuttavia, sono spesso preferiti quando è importante comprendere visivamente come un oggetto è orientato.

Il calcolo degli angoli di Eulero a partire dai dati dei sensori è più diretto rispetto ai quaternioni, ma richiede una gestione attenta delle ambiguità e delle conversioni tra i diversi sistemi di coordinate.

```
bno055_vector_t eul;  
bno055_setOperationModeNDOF();
```

**Listing 3.15:** Codice da inserire nel main

```
bno055_calibration_state_t cal = bno055_getCalibrationState();  
eul = bno055_getVectorEuler();  
printf("EUL - Yaw: %+2.2f | Roll: %+2.2f | Pitch: %+2.2f |  
      %+2.2d\r\n", eul.x, eul.y, eul.z, cal.sys);  
HAL_Delay(10);
```

**Listing 3.16:** Codice da inserire nel while



```

% Configura la porta seriale
s = serialport('COM4', 115200); % Assicurati che 'COM4' corrisponda
    alla tua porta seriale e che il baud rate sia corretto
try
    % Inizializza la visualizzazione 3D con il parallelepipedo
    figure;
    % Definisci le coordinate del parallelepipedo
    cubeCoords = [
        -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 0.5 0.5 0.5
         0.5 0.5 0.5 -0.5 -0.5;
        -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5
         0.5 -0.5 0.5 0.5 0.5;
        -0.5 -0.5 -0.5 -0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 -0.5 0.5
        -0.5 0.5 0.5 0.5 -0.5;
    ];
    cubePlot = plot3(cubeCoords(1, :), cubeCoords(2, :),
        cubeCoords(3, :), 'c-', 'LineWidth', 3);

    xlabel('X');
    ylabel('Y');
    zlabel('Z');
    title('Orientamento con parallelepipedo 3D');
    grid on;
    % Imposta i limiti dell'asse
    axis([-1 1 -1 1 -1 1]); % Modifica i valori in base alle
        dimensioni desiderate
    % Loop infinito per leggere dati
    while true
        % Legge una riga di dati dalla porta seriale
        data = readline(s);
        % Aggiungi una stampa di debug per visualizzare la stringa
            ricevuta
        disp(['Dati ricevuti: ' data]);
        % Estrai gli angoli di Eulero dalla stringa
        matches = sscanf(data, 'EUL - Yaw: %f | Roll: %f | Pitch: %f
            | %d\r\n');
        % Verifica se il formato della stringa è corretto
        if numel(matches) == 4
            % Assegna i valori estratti agli angoli di Eulero
            % se si vuole cambiare il segno di un asse è possibile
                mettendo un meno es. yaw = -deg2rad(matches(1));
            yaw = deg2rad(matches(1));
            roll = deg2rad(matches(2));
            pitch = deg2rad(matches(3));
            calibrationValue = matches(4);
            % Calcola la matrice di rotazione dagli angoli di Eulero

            matrix_X_roll = [
                1, 0, 0;
                0, cos(roll), -sin(roll);
                0, sin(roll), cos(roll)
            ];

            matrix_Y_pitch = [
                cos(pitch), 0, sin(pitch);
                0, 1, 0;
                -sin(pitch), 0, cos(pitch)
            ];

            matrix_Z_yaw = [

```

```

        cos(yaw), -sin(yaw), 0;
        sin(yaw), cos(yaw), 0;
        0, 0, 1
    ];
    %calcolo della matrice ZYX
    rotationMatrix = matrix_Z_yaw * matrix_Y_pitch *
        matrix_X_roll;
    % Ruota le coordinate del parallelepipedo utilizzando la
    % matrice di rotazione
    rotatedCubeCoords = rotationMatrix * cubeCoords;
    % Aggiorna la posizione del parallelepipedo nella
    % visualizzazione 3D
    set(cubePlot, 'XData', rotatedCubeCoords(1, :), 'YData',
        rotatedCubeCoords(2, :), 'ZData',
        rotatedCubeCoords(3, :));

        drawnow; % Aggiorna la visualizzazione
    else
        disp('Formato della stringa non valido.');
```

end

% Aggiungi una pausa (tempo di attesa) tra le letture

pause(0.01); % 10 millisecondi (puoi regolare il valore in base alle tue esigenze)

end

catch exception

% Gestione degli errori

disp(['Errore: ' exception.message]);

end

% Chiudi la porta seriale alla fine

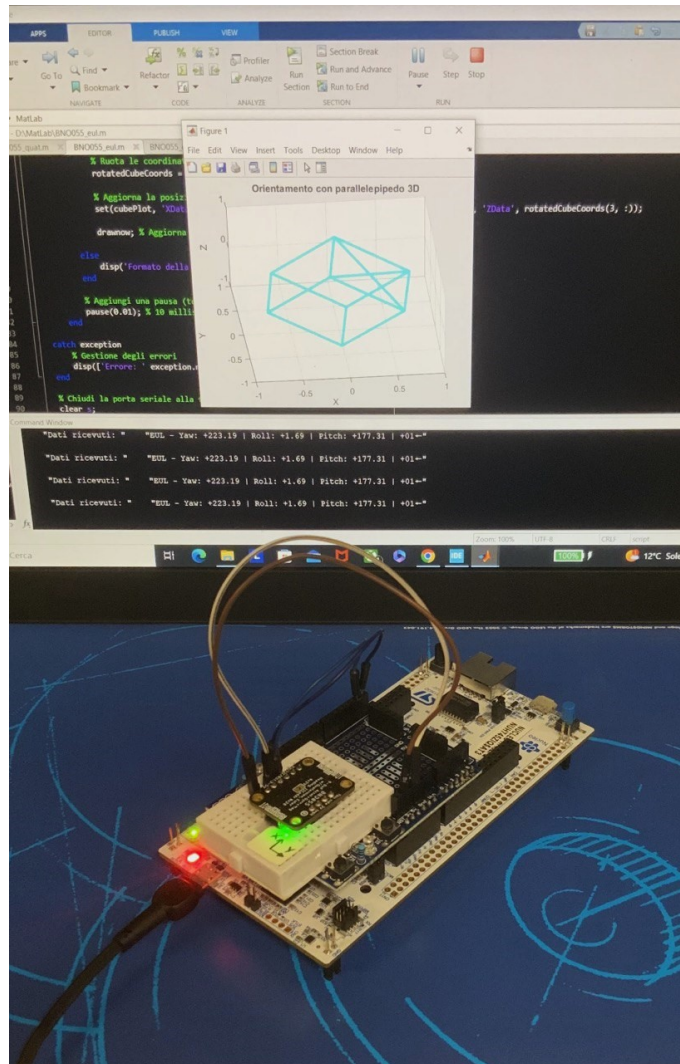
clear s;

Listing 3.17: Codice in Matlab

Anche in questo caso il codice matlab deve ottenere nello sscanf la stessa stringa dell'ide. Per creare questo codice ci siamo serviti delle matrici di rotazione, in particolare abbiamo utilizzato la matrice di Tait–Bryan angles ZYX ma può anche essere utilizzata la ZXY, le altre non sono state provate. I riferimenti possono essere trovati qui:

[link per matrici di rotazione roll, pitch e yaw](#) [6]

[link per calcolo della matrice prodotto](#) [7]



**Figure 3.8:** Risultato del codice mostrato

I dati ottenuti rispettano i range ammissibili mostrati in tabella 3.18, inoltre il loro impiego per ricostruire un parallelepipedo graficamente risulta corretto. Può risultare necessario cambiare il verso degli assi, come nel codice commentato nel codice Matlab corrente (es .  $\text{yaw} = -\text{deg2rad}(\text{matches}(1))$  ;), per rendere la rappresentazione grafica coerente con quanto misurato.

La rappresentazione ottenuta però non è precisa come quella ottenuta dai quaternioni, un motivo potrebbe essere dovuto al "gimbal lock". Quello che si è riscontrato è che in certi orientamenti un lieve spostamento causava un balzo nei dati ricevuti e di conseguenza uno scatto della rappresentazione.

Infatti, è stato riscontrato un salto dei valori misurati a seguito di lievi spostamenti in particolari orientamenti. In dettaglio, questa anomalia è stata notata ad un certo orientamento sull'asse x, circa 45 gradi. Conseguentemente, i salti rilevati nelle misurazioni comportano degli scatti nella rappresentazione grafica.

### 3.3.3 Conversione QUAT to EUL

Per poter eseguire calcoli veloci con i quaternioni e allo stesso tempo analizzare le rotazioni utilizzando gli angoli, potrebbe essere necessario disporre di un metodo efficiente per convertire un insieme di parametri nell'altro. Dal punto di vista di STM32CubeIde il codice è uguale a quello nella sezione quaternioni 3.3.1.

```
% Configura la porta seriale
s = serialport('COM4', 115200); % Assicurati che 'COM4' corrisponda
    alla tua porta seriale e che il baud rate sia corretto
try
    % Inizializza la visualizzazione 3D con il parallelepipedo
    figure;
    % Definisci le coordinate del parallelepipedo
    cubeCoords = [
        -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 0.5 0.5 0.5
         0.5 0.5 0.5 -0.5 -0.5;
        -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5 0.5 -0.5 -0.5 -0.5 0.5
         0.5 -0.5 0.5 0.5 0.5;
        -0.5 -0.5 -0.5 -0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 -0.5 0.5
        -0.5 0.5 0.5 0.5 -0.5;
    ];
    cubePlot = plot3(cubeCoords(1, :), cubeCoords(2, :),
        cubeCoords(3, :), 'c-', 'LineWidth', 3);
    xlabel('X');
    ylabel('Y');
    zlabel('Z');
    title('Orientamento con parallelepipedo 3D');
    grid on;
    % Imposta i limiti dell'asse
    axis([-1 1 -1 1 -1 1]); % Modifica i valori in base alle
        dimensioni desiderate
    % Loop infinito per leggere dati
    while true
        % Legge una riga di dati dalla porta seriale
        data = readline(s);
        % Aggiungi una stampa di debug per visualizzare la stringa
            ricevuta
        disp(['Dati ricevuti: ' data]);
        % Estrai i componenti del quaternion e il valore aggiuntivo
            cal.sys dalla stringa
        matches = sscanf(data, 'QUAT - x: %f | y: %f | z: %f | w: %f
            | %d\r\n');
        % Verifica se il formato della stringa è corretto
        if numel(matches) == 5
            % Assegna i valori estratti ai componenti del
                quaternion e al valore cal.sys
            qw = matches(4);
            qx = matches(1);
            qy = matches(2);
            qz = matches(3);
            calibrationValue = matches(5);
            % Calcola gli angoli di Eulero dal quaternion
            roll = atan2(2*(qw*qx + qy*qz), 1 - 2*(qx^2 + qy^2));
            % Calcola l'angolo di pitch
            % La variabile temp impedisce di ottenere numeri
                complessi
            % dovuti ad una certa inclinazione
            temp=1+2*(qw*qy - qx*qz);
            if temp<0
```

```

        temp=0;
    end
    pitch = -pi/2 + 2*atan2(sqrt(temp), sqrt(1-2*(qw*qy -
        qx*qz)));

    yaw = atan2(2*(qw*qz + qx*qy), 1 - 2*(qy^2 + qz^2));
    disp(['Angoli di Eulero: Roll = ' num2str(rad2deg(roll))
        ' Pitch = ' num2str(rad2deg(pitch)) ' Yaw = '
        num2str(rad2deg(yaw))]);
    % Ruota le coordinate del parallelepipedo utilizzando
    % gli angoli di Eulero
    rotatedCubeCoords = rotatecube(cubeCoords, roll, pitch,
        yaw);
    % Aggiorna la posizione del parallelepipedo nella
    % visualizzazione 3D
    set(cubePlot, 'XData', rotatedCubeCoords(1, :), 'YData',
        rotatedCubeCoords(2, :), 'ZData',
        rotatedCubeCoords(3, :));
    drawnow; % Aggiorna la visualizzazione
else
    disp('Formato della stringa non valido.');
```

```

end
    % Aggiungi una pausa (tempo di attesa) tra le letture
    pause(0.01); % 10 millisecondi (puoi regolare il valore in
        base alle tue esigenze)
end
catch exception
    % Gestione degli errori
    disp(['Errore: ' exception.message]);
end
% Chiudi la porta seriale alla fine
clear s;
% Funzione per ruotare le coordinate del parallelepipedo utilizzando
% gli angoli di Eulero
function rotatedCoords = rotatecube(coords, roll, pitch, yaw)

    matrix_X_roll = [
        1, 0, 0;
        0, cos(roll), -sin(roll);
        0, sin(roll), cos(roll)
    ];

    matrix_Y_pitch = [
        cos(pitch), 0, sin(pitch);
        0, 1, 0;
        -sin(pitch), 0, cos(pitch)
    ];

    matrix_Z_yaw = [
        cos(yaw), -sin(yaw), 0;
        sin(yaw), cos(yaw), 0;
        0, 0, 1
    ];

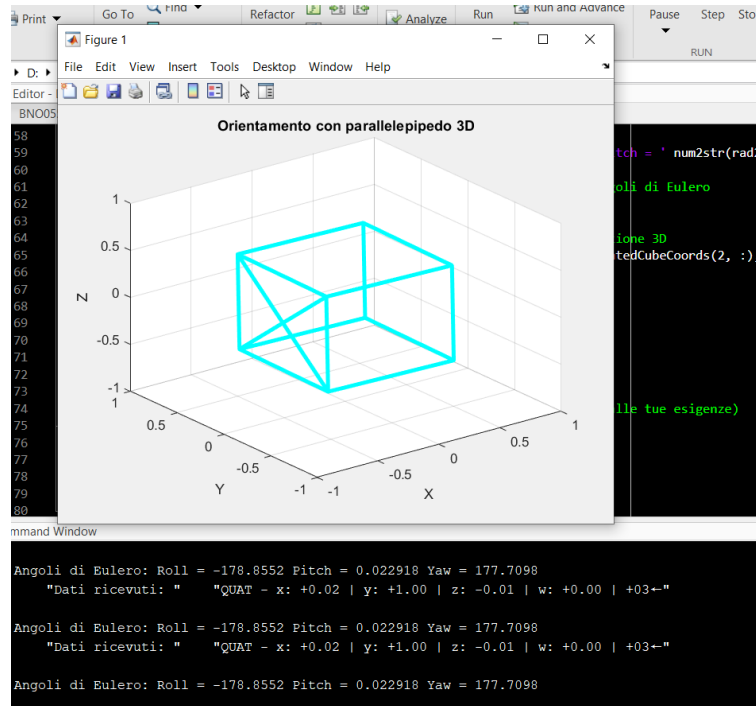
    rotationMatrix = matrix_Z_yaw * matrix_Y_pitch *
        matrix_X_roll;
    rotatedCoords = rotationMatrix * coords;
end

```

Listing 3.18: Codice in Matlab

Per ottenere questo codice abbiamo utilizzato il codice relativo ai quaternioni e ci abbiamo aggiunto una funzione per la rotazione delle matrici degli angoli di Eulero, è stata poi usata la formula di conversione "body 3-2-1" presa da Wikipedia [8]. L'algoritmo per il calcolo dell'angolo di Pitch è stato modificato per evitare errori numerici di calcolo.

NOTA: se viene utilizzata una sequenza di rotazione diversa, ad esempio XYZ, le formule cambiano. Se si desidera implementare una conversione da quaternioni ad angoli di Eulero diversa da quella indicata bisogna considerare la rispettiva formula di conversione [9].



**Figure 3.9:** Esempio del test effettuato

I valori di Roll, Pitch e Yaw non corrispondono ai valori che avrebbe preso il BNO055, infatti il bno055 utilizza questa convenzione di range 3.18 mentre gli angoli calcolati utilizzato la convenzione: Roll in asse Y (-180/+180), Pitch in asse X (-90/+90) e Yaw in asse Z (-180/+180) [10].

I risultati ottenuti risultano molto più fluidi e precisi rispetto ad ottenere fin da subito i dati con gli angoli di Eulero.

## 3.4 Rimappatura degli assi

La posizione in cui il dispositivo è montato non dovrebbe limitare l'output di dati del dispositivo BNO055. Gli assi del dispositivo possono essere riconfigurati per fare riferimento a nuovi assi di riferimento. La rimappatura degli assi coinvolge due byte di configurazione: `AXIS_MAP_CONFIG` e `AXIS_MAP_SIGN`, i quali permettono di personalizzare la configurazione degli assi in base alle necessità dell'applicazione.

### 3.4.1 `AXIS_MAP_CONFIG`

Questo byte di configurazione, situato all'indirizzo del registro `AXIS_MAP_CONFIG`, permette di specificare la rimappatura degli assi. I suoi bit sono definiti come segue:

- **Bit 7 (Reserved):** Riservato per scopi futuri.
- **Bit 6 (Remapped Z axis value):** Rappresenta l'asse Z rimappato.
- **Bit 5 (Remapped Y axis value):** Rappresenta l'asse Y rimappato.
- **Bit 4 (Remapped X axis value):** Rappresenta l'asse X rimappato.
- **Bit 3-0 (Reserved):** Riservati per scopi futuri.

I due bit meno significativi (**Bit 1** e **Bit 0**) definiscono l'asse rimappato nel seguente modo:

- **Valore 00:** Rappresenta l'asse X.
- **Valore 01:** Rappresenta l'asse Y.
- **Valore 10:** Rappresenta l'asse Z.
- **Valore 11:** Condizione non valida.

Se si cerca di configurare lo stesso asse su due o più assi differenti, il BNO055 considererà questa condizione come non valida e ripristinerà la configurazione precedente nel registro.

Il valore predefinito è: Asse X = X, Asse Y = Y e Asse Z = Z (`AXIS_MAP_CONFIG = 0x24`).

### 3.4.2 `AXIS_MAP_SIGN`

Questo byte di configurazione, situato all'indirizzo del registro `AXIS_MAP_SIGN`, consente di specificare il segno degli assi rimappati. I suoi bit sono definiti come segue:

- **Bit 7 (Reserved):** Riservato per scopi futuri.
- **Bit 6 (Remapped X axis sign):** Rappresenta il segno dell'asse X rimappato.
- **Bit 5 (Remapped Y axis sign):** Rappresenta il segno dell'asse Y rimappato.
- **Bit 4 (Remapped Z axis sign):** Rappresenta il segno dell'asse Z rimappato.

- **Bit 3-0 (Reserved):** Riservati per scopi futuri.

Ogni bit di segno corrisponde all'asse rimappato e può assumere uno dei seguenti valori:

- **Valore 0:** Segno positivo.
- **Valore 1:** Segno negativo.

Il valore predefinito è 0x00.

La rimappatura degli assi è una funzionalità utile quando la posizione di montaggio del dispositivo richiede una personalizzazione dell'orientamento degli assi per adattarsi alle esigenze specifiche dell'applicazione.

```
// Imposta la mappatura degli assi per il sensore BNO055
void bno055_setAxisMap(bno055_axis_map_t axis) {
    uint8_t axisRemap = (axis.z << 4) | (axis.y << 2) | (axis.x);
    uint8_t axisMapSign = (axis.x_sign << 2) | (axis.y_sign << 1) |
        (axis.z_sign);
    bno055_writeData(BNO055_AXIS_MAP_CONFIG, axisRemap);
    bno055_writeData(BNO055_AXIS_MAP_SIGN, axisMapSign);
}
```

Listing 3.19: Codice nella libreria

```
//Queste sono le impostazioni di default

bno055_axis_map_t myAxisMap;
myAxisMap.x = 0;           // Configura la mappatura dell'asse X
myAxisMap.y = 1;           // Configura la mappatura dell'asse Y
myAxisMap.z = 2;           // Configura la mappatura dell'asse Z
myAxisMap.x_sign = 0;      // Configura il segno dell'asse X
myAxisMap.y_sign = 0;      // Configura il segno dell'asse Y
myAxisMap.z_sign = 0;      // Configura il segno dell'asse Z

bno055_setAxisMap(myAxisMap);
```

Listing 3.20: Codice da inserire nel main



### 3.5 Configurazione sensori

La configurazione dei sensori è un aspetto cruciale per il corretto funzionamento del BNO055 e il calcolo delle uscite di fusione. La configurazione dei sensori è strettamente collegata alle modalità operative della fusione del BNO055. Pertanto, è importante comprendere come le impostazioni dei sensori influenzino il comportamento complessivo del dispositivo.

Quando il BNO055 è configurato per funzionare in una delle modalità operative di fusione (come ad esempio NDOF o COMPASS), le opzioni di configurazione dei sensori sono limitate, poiché il sensore utilizza internamente una serie di impostazioni predeterminate per garantire un funzionamento ottimale della fusione dei dati.

Tuttavia, quando il BNO055 è impostato in una delle modalità operative non di fusione (come ad esempio ACCONLY o GYROONLY), è possibile aggiornare le impostazioni dei sensori. Ciò significa che è possibile personalizzare le modalità operative dei singoli sensori, come l'accelerometro e il giroscopio, per soddisfare le esigenze specifiche dell'applicazione.

Le impostazioni dei sensori includono parametri come la frequenza di campionamento, la gamma di misurazione e altre opzioni di configurazione che possono influire sulla precisione e sulla velocità dei dati forniti dal BNO055.

È importante notare che una configurazione errata dei sensori può influire negativamente sulle prestazioni del dispositivo. Pertanto, è consigliabile consultare attentamente il datasheet del BNO055 e comprendere le varie opzioni di configurazione dei sensori disponibili prima di apportare modifiche alle impostazioni predefinite.

In breve, la configurazione dei sensori è un aspetto chiave per ottenere risultati accurati e affidabili dal BNO055, e una comprensione approfondita di queste impostazioni è essenziale per massimizzare il valore del sensore nelle tue applicazioni.

### 3.5.1 Configurazione predefinita

Nelle modalità di fusione, le impostazioni dei sensori sono controllate dal BNO055 e configurate come definito nella seguente tabella 3.2. Nelle modalità non di fusione, le impostazioni dei sensori possono essere configurate dall'utente mentre sono in modalità CONFIG\_MODE e vengono mantenute anche dopo lo spegnimento del sensore.

Sensore	Parametri	Valore
Accelerometer	Power Mode	NORMAL
	Range	+/- 4g
	Bandwidth	62.5Hz
	Resolution	14 bit
Gyroscope	Power Mode	NORMAL
	Range	2000 °/s
	Bandwidth	32Hz
	Resolution	16 bit
Magnetometer	Power Mode	FORCED
	ODR	20Hz
	Repetition XY	15
	Repetition Z	16
	Resolution x/y/z	13/13/15 bit

**Table 3.2:** *Configurazione predefinita dei sensori all'accensione*

- Power Mode: determina lo stato di alimentazione del sensore.
- Range: specifica l'intervallo di misurazione.
- Bandwidth: determina la frequenza massima misurabile.
- Resolution: indica il numero di bit utilizzato per la rappresentazione.
- Data Rate (ODR): determina la frequenza di campionamento dei dati.
- Repetition: specifica il numero di campionamenti per ogni misurazione.

### 3.5.2 Configurazione accelerometro

La configurazione dell'accelerometro è strettamente legata alle impostazioni del sensore di fusione della BNO055. Pertanto, le possibilità di configurazione sono limitate quando si esegue in una modalità fusion, come indicato nella colonna "Vincoli" della tabella. La configurazione dell'accelerometro può essere modificata scrivendo nel registro `ACC_Config`.

La Tabella sottostante mostra diverse configurazioni dell'accelerometro.

Parametro	Valori	Reg Addr	Vincoli
G Range	2G	[ACC_Config]: xxxxxx00b	Auto controllato in modalità fusione
	4G	[ACC_Config]: xxxxxx01b	
	8G	[ACC_Config]: xxxxxx10b	
	16G	[ACC_Config]: xxxxxx11b	
Bandwidth	7.81Hz	[ACC_Config]: xxx000xxb	Auto controllato in modalità fusione
	15.63Hz	[ACC_Config]: xxx001xxb	
	31.25Hz	[ACC_Config]: xxx010xxb	
	62.5Hz	[ACC_Config]: xxx011xxb	
	125Hz	[ACC_Config]: xxx100xxb	
	250Hz	[ACC_Config]: xxx101xxb	
	500Hz	[ACC_Config]: xxx110xxb	
	1000Hz	[ACC_Config]: xxx111xxb	
Operation Mode	Normal	[ACC_Config]: 000xxxxxb	Auto controllato in modalità fusione
	Suspend	[ACC_Config]: 001xxxxxb	
	Low Power 1	[ACC_Config]: 010xxxxxb	
	Standby	[ACC_Config]: 011xxxxxb	
	Low Power 2	[ACC_Config]: 100xxxxxb	
	Deep Suspend	[ACC_Config]: 101xxxxxb	

**Table 3.3:** Configurazioni dell'accelerometro

La modalità operativa dell'accelerometro non è configurabile dall'utente quando la modalità di alimentazione del BNO055 è impostata su basso consumo. Viene sovrascritto il valore configurato dall'utente su "Normale" quando si passa dalla modalità di configurazione a qualsiasi altra modalità operativa.

```

void bno055_configureAccelerometer(uint8_t gRange, float bandwidth,
uint8_t operationMode) {
    bno055_setPage(1); //il registro si trova in page 1
    uint8_t accConfigReg = 0;

    if (gRange == 2) {
        accConfigReg |= 0; // [ACC_Config]: xxxxxx00b
    } else if (gRange == 4) {
        accConfigReg |= 1; // [ACC_Config]: xxxxxx01b
    } else if (gRange == 8) {
        accConfigReg |= 2; // [ACC_Config]: xxxxxx10b
    } else if (gRange == 16) {
        accConfigReg |= 3; // [ACC_Config]: xxxxxx11b
    } else {
        printf("Errore: G Range non valido\r\n");
        return;
    }
    if (bandwidth == 7.81) {
        accConfigReg |= (0 << 2); // [ACC_Config]: xxx000xxb
    } else if (bandwidth == 15.63) {
        accConfigReg |= (1 << 2); // [ACC_Config]: xxx001xxb
    } else if (bandwidth == 31.25) {
        accConfigReg |= (2 << 2); // [ACC_Config]: xxx010xxb
    } else if (bandwidth == 62.5) {
        accConfigReg |= (3 << 2); // [ACC_Config]: xxx011xxb
    } else if (bandwidth == 125) {
        accConfigReg |= (4 << 2); // [ACC_Config]: xxx100xxb
    } else if (bandwidth == 250) {
        accConfigReg |= (5 << 2); // [ACC_Config]: xxx101xxb
    } else if (bandwidth == 500) {
        accConfigReg |= (6 << 2); // [ACC_Config]: xxx110xxb
    } else if (bandwidth == 1000) {
        accConfigReg |= (7 << 2); // [ACC_Config]: xxx111xxb
    } else {
        printf("Errore: Bandwidth non valido\r\n");
        return;
    }
    if (operationMode == NORMAL) {
        accConfigReg |= 0; // [ACC_Config]: 000xxxxxb
    } else if (operationMode == SUSPEND) {
        accConfigReg |= (1 << 5); // [ACC_Config]: 001xxxxxb
    } else if (operationMode == LOW_POWER_1) {
        accConfigReg |= (2 << 5); // [ACC_Config]: 010xxxxxb
    } else if (operationMode == STANDBY) {
        accConfigReg |= (3 << 5); // [ACC_Config]: 011xxxxxb
    } else if (operationMode == LOW_POWER_2) {
        accConfigReg |= (4 << 5); // [ACC_Config]: 100xxxxxb
    } else if (operationMode == DEEP_SUSPEND) {
        accConfigReg |= (5 << 5); // [ACC_Config]: 101xxxxxb
    } else {
        printf("Errore: Modalita di funzionamento non valida\r\n");
        return;
    }
    bno055_setOperationModeConfig();
    bno055_writeData(BNO055_ACC_CONFIG, accConfigReg);
    bno055_setPage(0); //tornare in page 0
}

```

Listing 3.21: Codice della libreria

```
uint8_t gRange = ?; //scrivere il valore desiderato  
float bandwidth = ?; //scrivere il valore desiderato  
uint8_t operationMode = ?; //scrivere la modalita desiderata  
  
bno055_configureAccelerometer(gRange, bandwidth, operationMode);  
    //richiama la funzione di configurazione
```

Listing 3.22: **Codice da inserire nel main**

Nel caso fossero inseriti valori non validi, la funzione stamperà il messaggio di errore e continuerà lasciando i parametri invariati.

NOTA: Per le configurazioni del giroscopio e del magnetometro la modalità di utilizzo è la stessa.

### 3.5.3 Configurazione giroscopio

Le uscite di fusione del BNO055 sono strettamente collegate alle impostazioni del sensore di velocità angolare. Di conseguenza, le possibilità di configurazione sono limitate quando si esegue in una modalità fusion, come indicato nella colonna "Vincoli" della tabella. La configurazione del giroscopio può essere modificata scrivendo nel registro **GYR\_Config**.

La Tabella sottostante mostra diverse configurazioni del giroscopio.

Parametro	Valori	Reg Addr	Vincoli
G Range	2000dps 1000dps 500dps 250dps 125dps	[GYR_Config_0]: xxxxx000b [GYR_Config_0]: xxxxx001b [GYR_Config_0]: xxxxx010b [GYR_Config_0]: xxxxx011b [GYR_Config_0]: xxxxx100b	Auto controllato in modalità fusione
Bandwidth	523Hz 230Hz 116Hz 47Hz 23Hz 12Hz 64Hz 32Hz	[GYR_Config_0]: xx000xxx b [GYR_Config_0]: xx001xxx b [GYR_Config_0]: xx010xxx b [GYR_Config_0]: xx011xxx b [GYR_Config_0]: xx100xxx b [GYR_Config_0]: xx101xxx b [GYR_Config_0]: xx110xxx b [GYR_Confi_Og]: xx111xxx b	Auto controllato in modalità fusione
Operation Mode	Normal Fast Power Up Deep Suspend Suspend Advanced Powersave	[GYR_Config_1]: xxxxx000b [GYR_Config_1]: xxxxx001b [GYR_Config_1]: xxxxx010b [GYR_Config_1]: xxxxx011b [GYR_Config_1]: xxxxx100b	Auto controllato in modalità fusione

**Table 3.4:** Configurazioni del giroscopio

```

void bno055_configureGyroscope(uint8_t gRange, uint8_t bandwidth,
uint8_t operationMode) {
    bno055_setPage(1); //il registro si trova in page 1
    uint8_t gyrConfigReg_0 = 0;
    uint8_t gyrConfigReg_1 = 0;
    if (gRange == 2000) {
        gyrConfigReg_0 |= 0; // [GYRO_Config_0]: xxxxx000b
    } else if (gRange == 1000) {
        gyrConfigReg_0 |= 1; // [GYRO_Config_0]: xxxxx001b
    } else if (gRange == 500) {
        gyrConfigReg_0 |= 2; // [GYRO_Config_0]: xxxxx010b
    } else if (gRange == 250) {
        gyrConfigReg_0 |= 3; // [GYRO_Config_0]: xxxxx011b
    } else if (gRange == 125) {
        gyrConfigReg_0 |= 4; // [GYRO_Config_0]: xxxxx100b
    } else {
        printf("Errore: G Range non valido\r\n");
        return;
    }
    if (bandwidth == 523) {
        gyrConfigReg_0 |= (0 << 3); // [GYRO_Config_0]: xx000xxxb
    } else if (bandwidth == 230) {
        gyrConfigReg_0 |= (1 << 3); // [GYRO_Config_0]: xx001xxxb
    } else if (bandwidth == 116) {
        gyrConfigReg_0 |= (2 << 3); // [GYRO_Config_0]: xx010xxxb
    } else if (bandwidth == 47) {
        gyrConfigReg_0 |= (3 << 3); // [GYRO_Config_0]: xx011xxxb
    } else if (bandwidth == 23) {
        gyrConfigReg_0 |= (4 << 3); // [GYRO_Config_0]: xx100xxxb
    } else if (bandwidth == 12) {
        gyrConfigReg_0 |= (5 << 3); // [GYRO_Config_0]: xx101xxxb
    } else if (bandwidth == 64) {
        gyrConfigReg_0 |= (6 << 3); // [GYRO_Config_0]: xx110xxxb
    } else if (bandwidth == 32) {
        gyrConfigReg_0 |= (7 << 3); // [GYRO_Config_0]: xx111xxxb
    } else {
        printf("Errore: Bandwidth non valido\r\n");
        return;
    }
    if (operationMode == NORMAL) {
        gyrConfigReg_1 |= 0; // [GYRO_Config_1]: xxxxx000b
    } else if (operationMode == FAST_POWER_UP) {
        gyrConfigReg_1 |= 1; // [GYRO_Config_1]: xxxxx001b
    } else if (operationMode == DEEP_SUSPEND) {
        gyrConfigReg_1 |= 2; // [GYRO_Config_1]: xxxxx010b
    } else if (operationMode == SUSPEND) {
        gyrConfigReg_1 |= 3; // [GYRO_Config_1]: xxxxx011b
    } else if (operationMode == ADVANCED_POWERSAVE) {
        gyrConfigReg_1 |= 4; // [GYRO_Config_1]: xxxxx100b
    } else {
        printf("Errore: Modalita di funzionamento non valida\r\n");
        return;
    }
    bno055_setOperationModeConfig();
    bno055_writeData(BNO055_GYRO_CONFIG_0, gyrConfigReg_0);
    bno055_writeData(BNO055_GYRO_CONFIG_1, gyrConfigReg_1);
    bno055_setPage(0); //tornare in page 0
}

```

Listing 3.23: Codice della libreria

### 3.5.4 Configurazione magnetometro

Le uscite di fusione del BNO055 sono strettamente collegate alle impostazioni del sensore magnetometro. Pertanto, le possibilità di configurazione sono limitate quando si esegue in una modalità fusion, come indicato nella colonna "Vincoli" della tabella. È possibile modificare la configurazione del magnetometro scrivendo nel registro **MAG\_Config**.

La tabella sottostante mostra le diverse configurazioni del magnetometro disponibili.

Parametro	Valori	Reg Addr	Vincoli
Data output rate	2Hz	[MAG_Config]: xxxxx000b	Auto controllato in modalità fusione
	6Hz	[MAG_Config]: xxxxx001b	
	8Hz	[MAG_Config]: xxxxx010b	
	10Hz	[MAG_Config]: xxxxx011b	
	15Hz	[MAG_Config]: xxxxx100b	
	20Hz	[MAG_Config]: xxxxx101b	
	25Hz	[MAG_Config]: xxxxx110b	
	30Hz	[MAG_Config]: xxxxx111b	
Operation Mode	Low Power	[MAG_Config]: xxx00xxxb	Auto controllato in modalità fusione
	Regular	[MAG_Config]: xxx01xxxb	
	Enhanced Regular	[MAG_Config]: xxx10xxxb	
	High Accuracy	[MAG_Config]: xxx11xxxb	
Power Mode	Normal	[MAG_Config]: x00xxxxxb	Auto controllato in modalità fusione
	Sleep	[MAG_Config]: x01xxxxxb	
	Suspend	[MAG_Config]: x10xxxxxb	
	Force Mode	[MAG_Config]: x11xxxxxb	

**Table 3.5:** *Configurazioni del magnetometro*



```

void bno055_configureMagnetometer(uint8_t dataOutputRate, uint8_t
operationMode, uint8_t powerMode) {
    bno055_setPage(1); //il registro si trova in page 1
    uint8_t magConfigReg = 0;

    if (dataOutputRate == 2) {
        magConfigReg |= 0 ; // [MAG_Config]: xxxxx000b
    } else if (dataOutputRate == 6) {
        magConfigReg |= 1 ; // [MAG_Config]: xxxxx001b
    } else if (dataOutputRate == 8) {
        magConfigReg |= 2 ; // [MAG_Config]: xxxxx010b
    } else if (dataOutputRate == 10) {
        magConfigReg |= 3 ; // [MAG_Config]: xxxxx011b
    } else if (dataOutputRate == 15) {
        magConfigReg |= 4 ; // [MAG_Config]: xxxxx100b
    } else if (dataOutputRate == 20) {
        magConfigReg |= 5 ; // [MAG_Config]: xxxxx101b
    } else if (dataOutputRate == 25) {
        magConfigReg |= 6 ; // [MAG_Config]: xxxxx110b
    } else if (dataOutputRate == 30) {
        magConfigReg |= 7 ; // [MAG_Config]: xxxxx111b
    } else {
        printf("Errore: Data Output Rate non valido\r\n");
        return;
    }

    if (operationMode == LOW_POWER) {
        magConfigReg |= (0 << 3); // [MAG_Config]: xxx00xxxxb
    } else if (operationMode == REGULAR) {
        magConfigReg |= (1 << 3); // [MAG_Config]: xxx01xxxxb
    } else if (operationMode == ENHANCED_REGULAR) {
        magConfigReg |= (2 << 3); // [MAG_Config]: xxx10xxxxb
    } else if (operationMode == HIGH_ACCURACY) {
        magConfigReg |= (3 << 3); // [MAG_Config]: xxx11xxxxb
    } else {
        printf("Errore: Operation Mode non valida\r\n");
        return;
    }

    if (powerMode == NORMAL) {
        magConfigReg |= (0 << 5); // [MAG_Config]: x00xxxxxb
    } else if (powerMode == SLEEP ) {
        magConfigReg |= (1 << 5); // [MAG_Config]: x01xxxxxb
    } else if (powerMode == SUSPEND) {
        magConfigReg |= (2 << 5); // [MAG_Config]: x10xxxxxb
    } else if (powerMode == FORCE_MODE) {
        magConfigReg |= (3 << 5); // [MAG_Config]: x11xxxxxb
    } else {
        printf("Errore: Power Mode non valida\r\n");
        return;
    }

    bno055_setOperationModeConfig();
    bno055_writeData(BNO055_MAG_CONFIG, magConfigReg);
    bno055_setPage(0); //tornare in page 0
}

```

Listing 3.24: Codice della libreria

## 3.6 Offsets

### 3.6.1 Configurazione dell'Offset dell'Accelerometro

L'offset dell'accelerometro può essere configurato utilizzando i registri elencati nella seguente tabella 3.6. Per configurare l'offset dell'accelerometro sono necessari sei byte, con 2 byte dedicati a ciascuno dei tre assi: X, Y e Z. La configurazione avrà effetto solo quando l'utente scriverà l'ultimo byte, cioè ACC\_OFFSET\_Z\_MSB.

Nome del Registro	Valore di Registro Predefinito (Bit 0 – Bit 7)
ACC_OFFSET_X_LSB	0x00
ACC_OFFSET_X_MSB	0x00
ACC_OFFSET_Y_LSB	0x00
ACC_OFFSET_Y_MSB	0x00
ACC_OFFSET_Z_LSB	0x00
ACC_OFFSET_Z_MSB	0x00

**Table 3.6:** *Impostazioni Predefinite dei Registri dell'Offset dell'Accelerometro*

Il range degli offset per l'accelerometro è di +/- 500 in LSB.

Offset per il Sensore	Range Massimo dell'Offset in LSB
Accelerometro	+/- 500

**Table 3.7:** *Impostazioni del Range dell'Offset dell'Accelerometro*

Rappresentazione dell'Unità	Valore
m/s <sup>2</sup>	1 m/s <sup>2</sup> = 100 LSB
mg	1 mg = 1 LSB

**Table 3.8:** *Impostazioni dell'Unità dell'Accelerometro*

### 3.6.2 Configurazione dell'Offset del Magnetometro

L'offset del magnetometro può essere configurato utilizzando i registri elencati nella seguente tabella 3.9. Per configurare l'offset del magnetometro sono necessari sei byte, con 2 byte dedicati a ciascuno dei tre assi: X, Y e Z. La configurazione avverrà solo quando l'utente scriverà l'ultimo byte, cioè MAG\_OFFSET\_Z\_MSB. Pertanto, l'utente deve scrivere l'ultimo byte ogni volta che desidera cambiare la configurazione.

Nome del Registro	Valore di Registro Predefinito (Bit 0 – Bit 7)
MAG_OFFSET_X_LSB	0x00
MAG_OFFSET_X_MSB	0x00
MAG_OFFSET_Y_LSB	0x00
MAG_OFFSET_Y_MSB	0x00
MAG_OFFSET_Z_LSB	0x00
MAG_OFFSET_Z_MSB	0x00

**Table 3.9:** *Impostazioni Predefinite dei Registri dell'Offset del Magnetometro*

Il range dell'offset del magnetometro è di +/-6400 in LSB.

Rappresentazione dell'Unità	Valore
$\mu\text{T}$ (microtesla)	1 $\mu\text{T}$ = 16 LSB

**Table 3.10:** *Impostazioni dell'Unità del Magnetometro*

### 3.6.3 Configurazione dell'Offset del Giroscopio

L'offset del giroscopio può essere configurato utilizzando i registri elencati nella seguente tabella 3.11. Per configurare l'offset del giroscopio sono necessari sei byte, con 2 byte dedicati a ciascuno dei tre assi: X, Y e Z. La configurazione avverrà solo quando l'utente scriverà l'ultimo byte, cioè GYR\_OFFSET\_Z\_MSB. Pertanto, l'utente deve scrivere l'ultimo byte ogni volta che desidera cambiare la configurazione.

Nome del Registro	Valore di Registro Predefinito (Bit 0 – Bit 7)
GYR_OFFSET_X_LSB	0x00
GYR_OFFSET_X_MSB	0x00
GYR_OFFSET_Y_LSB	0x00
GYR_OFFSET_Y_MSB	0x00
GYR_OFFSET_Z_LSB	0x00
GYR_OFFSET_Z_MSB	0x00

**Table 3.11:** *Impostazioni Predefinite dei Registri dell'Offset del Giroscopio*

Il range dell'offset per il giroscopio è di +/-2000 in LSB.

Offset per il Sensore	Range Massimo dell'Offset in LSB
Giroscopio	+/- 2000

**Table 3.12:** *Impostazioni del Range dell'Offset del Giroscopio*

Rappresentazione dell'Unità	Valore
Dps (gradi al secondo)	1 Dps = 16 LSB
Rps (radianti al secondo)	1 Rps = 900 LSB

**Table 3.13:** *Impostazioni dell'Unità del Giroscopio*

### 3.6.4 Configurazione del Raggio

Il raggio è la distanza tra l'asse di rotazione e il punto attivo del sensore, e può essere configurato utilizzando i registri elencati nella seguente tabella 3.14. Per configurare il raggio, sono necessari complessivamente quattro byte (due byte per ciascuno dell'accelerometro e del magnetometro). La configurazione avverrà solo quando l'utente scriverà l'ultimo byte, ossia ACC\_RADIUS\_MSB e MAG\_RADIUS\_MSB. Pertanto, l'utente deve scrivere l'ultimo byte ogni volta che desidera cambiare la configurazione.

Nome del Registro	Valore di Registro Predefinito (Bit 0 – Bit 7)
ACC_RADIUS_LSB	0x00
ACC_RADIUS_MSB	0x00
MAG_RADIUS_LSB	0x00
MAG_RADIUS_MSB	0x00

**Table 3.14:** *Impostazioni Predefinite dei Registri del Raggio*

Il range del raggio per l'accelerometro è di +/-2048 in LSB, mentre per il magnetometro è compreso tra 144 e 1280 in LSB. Per quanto riguarda il giroscopio, il parametro del raggio non è applicabile (NA).

Raggio per il Sensore	Range Massimo in LSB
Accelerometro	+/- 2048
Magnetometro	144 a 1280
Giroscopio	NA

**Table 3.15:** *Impostazioni del Range del Raggio*

Questi offset abbiamo provato ad utilizzarli per una calibrazione manuale ma senza risultato, i passaggi sono però riportati nella sezione "Test calibrazione manuale" 3.7.8.

## 3.7 Calibrazione

Nonostante il software di fusione dei sensori esegua l'algoritmo di calibrazione per tutti e tre i sensori (accelerometro, giroscopio e magnetometro) in background per rimuovere gli offset, alcune fasi preliminari devono essere garantite affinché questa calibrazione automatica possa avvenire. L'accelerometro e il giroscopio sono relativamente meno suscettibili alle interferenze esterne, pertanto l'offset è trascurabile. Tuttavia, il magnetometro è sensibile ai campi magnetici esterni e, quindi, per garantire la corretta precisione dell'orientamento (heading accuracy), devono essere seguite le seguenti fasi di calibrazione. A seconda dei sensori selezionati in modalità di fusione, i seguenti semplici passaggi devono essere eseguiti dopo ogni 'Power on Reset' per la corretta calibrazione del dispositivo.

### 3.7.1 Calibrazione accelerometro

Per calibrare correttamente l'accelerometro, seguire i seguenti passaggi:

1. Posizionare il dispositivo in 6 diverse posizioni stabili per alcuni secondi, consentendo all'accelerometro di effettuare la calibrazione.
2. Assicurarsi che ci sia un movimento lento tra le 2 posizioni stabili.
3. Le 6 posizioni stabili possono essere in qualsiasi direzione, ma assicurarsi che il dispositivo si trovi almeno una volta perpendicolare agli assi x, y e z.
4. È possibile leggere il registro CALIB\_STAT per verificare lo stato di calibrazione dell'accelerometro.

### 3.7.2 Calibrazione giroscopio

Per calibrare correttamente il giroscopio, seguire i seguenti passaggi:

1. Posizionare il dispositivo in una singola posizione stabile per alcuni secondi, consentendo al giroscopio di effettuare la calibrazione.
2. È possibile leggere il registro CALIB\_STAT per verificare lo stato di calibrazione del giroscopio.

### 3.7.3 Calibrazione magnetometro

I magnetometri sono generalmente sensibili sia alle distorsioni di tipo hard-iron che a quelle di tipo soft-iron, ma nella maggior parte dei casi si tratta di distorsioni di tipo hard-iron. Di seguito sono riportati i passaggi per calibrare il magnetometro per le distorsioni di tipo hard-iron.

1. Le distorsioni di tipo hard-iron possono essere corrette posizionando il magnetometro in sei diverse posizioni stabili e notando i valori dei campi magnetici rilevati.
2. Una volta acquisiti i dati relativi ai campi magnetici in queste posizioni, è possibile calcolare le correzioni necessarie per compensare le distorsioni di tipo hard-iron.

3. Assicurarsi di seguire le istruzioni dettagliate nella nota applicativa HSMI durante la gestione, la saldatura e il montaggio del sensore sulla scheda PCB per evitare influenze magnetiche indesiderate.

#### **3.7.4 Calibrazione del Compasso, Modalità M4G e Modalità NDOF\_FMC\_OFF**

Per calibrare correttamente il compasso in modalità M4G e modalità NDOF\_FMC\_OFF, seguire i seguenti passaggi:

1. Eseguire movimenti casuali (ad esempio, disegnando il numero '8' nell'aria) fino a quando il registro CALIB\_STAT indica una calibrazione completa.
2. È importante notare che sono necessari più movimenti di calibrazione per ottenere una calibrazione completa del magnetometro in modalità NDOF\_FMC\_OFF rispetto alla modalità NDOF.

#### **3.7.5 Calibrazione in Modalità NDOF**

Per calibrare correttamente il sensore in modalità NDOF, seguire i seguenti passaggi:

1. Eseguire gli stessi movimenti casuali necessari per calibrare il sensore nella modalità FMC\_OFF. Tuttavia, in questa modalità, è necessario eseguire relativamente meno movimenti di calibrazione (aumentando leggermente il consumo di corrente) per ottenere una calibrazione completa del magnetometro.
2. È possibile leggere il registro CALIB\_STAT per verificare lo stato di calibrazione del magnetometro.

## Modalità di utilizzo

Consigliamo di calibrare il BNO055 nelle modalità fusion, sfruttando la calibrazione automatica, come mostrato nella sezione Fusion Modes 3.2.3 utilizzando la funzione per ottenere lo stato della calibrazione di ciascun sensore.

Stampando i dati relativi allo stato del sensore, si può notare che inizialmente questi hanno valore nullo. Dopo pochi secondi, mantenendo il sensore fermo e in piano, il giroscopio raggiunge lo stato 3. Per permettere ai sensori rimanenti di raggiungere lo stato 3 è necessario eseguire le azioni riportate in sezione "Calibrazione" 3.7.

Nel seguente codice viene mostrato come vedere lo stato di tutti i sensori durante la calibrazione.

```
// Ottiene lo stato di calibrazione del sensore BNO055
bno055_calibration_state_t bno055_getCalibrationState() {
    bno055_setPage(0);
    bno055_calibration_state_t cal = {.sys = 0, .gyro = 0, .mag = 0,
    .accel = 0};
    uint8_t calState = 0;
    bno055_readData(BNO055_CALIB_STAT, &calState, 1);
    cal.sys = (calState >> 6) & 0x03;
    cal.gyro = (calState >> 4) & 0x03;
    cal.accel = (calState >> 2) & 0x03;
    cal.mag = calState & 0x03;
    return cal;
}
```

Listing 3.25: Codice nella libreria

```
bno055_vector_t quat;
bno055_setOperationModeNDOF();
```

Listing 3.26: Codice inserito nel main

```
bno055_calibration_state_t cal = bno055_getCalibrationState();
quat = bno055_getVectorQuaternion();
printf("GYR : %+2.2d | ACC : %+2.2d | MAG : %+2.2d | %+2.2d\r\n",
    cal.gyro, cal.accel, cal.mag, cal.sys);
```

Listing 3.27: Codice inserito nel while

Si è scelto di utilizzare la modalità fusion NDOF, ma è uguale ad ogni altra modalità fusion.

Inizialmente appena avviato il programma si troverà questo output, con tutti i sensori a zero.

```
GYR : +00 | ACC : +00 | MAG : +00 | +00
GYR : +00 | ACC : +00 | MAG : +00 | +00
GYR : +00 | ACC : +00 | MAG : +00 | +00
G
```

**Figure 3.10:** 1 - Risultato del codice

Se il dispositivo è fermo su un piano, dopo pochi secondi il giroscopio si sarà completamente calibrato.

```
GYR : +03 | ACC : +00 | MAG : +00 | +00
GYR : +03 | ACC : +00 | MAG : +00 | +00
GYR : +03 | ACC : +00 | MAG : +00 | +00
G
```

**Figure 3.11:** 2 - Risultato del codice

Muovendo il dispositivo ad otto si ottiene questo risultato, notare che oltre ad essere al 3 il giroscopio e il magnetometro, è a 3 anche l'ultimo parametro che è riferito al sistema. Normalmente dovrebbe raggiungere il 3 solo dopo che tutti i sensori si siano calibrati, ma nella modalità NDOF può farlo anche senza l'accelerometro perchè l'algoritmo di fusion funziona principalmente grazie agli altri due sensori.

```
GYR : +03 | ACC : +00 | MAG : +03 | +03
GYR : +03 | ACC : +00 | MAG : +03 | +03
GYR : +03 | ACC : +00 | MAG : +03 | +03
G
```

**Figure 3.12:** 3 - Risultato del codice

Una volta fatti i passaggi relativi alla calibrazione dell'accelerometro diventerà 3 anch'esso e il dispositivo perderà meno spesso la calibrazione.

```
GYR : +03 | ACC : +03 | MAG : +03 | +03
GYR : +03 | ACC : +03 | MAG : +03 | +03
GYR : +03 | ACC : +03 | MAG : +03 | +03
G
```

**Figure 3.13:** 3 - Risultato del codice



### 3.7.6 Calibrazione del Soft-Iron (SIC)

Il BNO055 supporta la calibrazione del Soft-Iron (SIC) tramite una matrice di calibrazione 3x3. All'avvio, la matrice di identità viene utilizzata come coefficienti per il segnale del magnetometro. Una matrice di compensazione SIC può efficientemente compensare le distorsioni dovute a materiali magnetici morbidi nelle vicinanze del sensore. I dati del magnetometro vengono prima moltiplicati per questa matrice SIC prima di essere utilizzati nell'algoritmo di fusione dei sensori.

$$\text{Matrice di Calibrazione Soft-Iron (SIC)} = \begin{bmatrix} M_{sb0Lsb0} & M_{sb1Lsb1} & M_{sb2Lsb2} \\ M_{sb3Lsb3} & M_{sb4Lsb4} & M_{sb5Lsb5} \\ M_{sb6Lsb6} & M_{sb7Lsb7} & M_{sb8Lsb8} \end{bmatrix}$$

Ciascun valore della matrice è rappresentato da un intero con segno su 16 bit e può rappresentare un valore nell'intervallo da -2.0000 a 1.9999 utilizzando il fattore di conversione  $1/2^{14}$ .

User Value	Conversion Factor	Actual Value
-32768	$1/2^{14}$	-2.0000
...		...
-16384	$1/2^{14}$	-1.0000
...		...
0	$1/2^{14}$	0.0000
...		...
16384	$1/2^{14}$	1.0000
...		...
32767	$1/2^{14}$	1.9999

**Figure 3.14:** *Fattore di Scala della Matrice di Calibrazione Soft-Iron (SIC)*

Il metodo migliore per generare la matrice di calibrazione Soft-Iron (SIC) è utilizzare le bobine di Helmholtz, anche se è possibile effettuare una calibrazione di campo meno precisa utilizzando il campo magnetico della Terra.

La generazione della matrice SIC non è stata trattata in questa relazione.

### 3.7.7 Profilo di Calibrazione

Una volta che il dispositivo è stato calibrato, il profilo di calibrazione può essere riutilizzato per ottenere immediatamente i dati di orientamento corretti subito dopo un 'Power of Reset'. Tuttavia, una volta che il sensore entra nella procedura interna di calibrazione, il profilo di calibrazione viene sovrascritto con i nuovi offset del sensore e il raggio del sensore appena ottenuti. A seconda dell'applicazione, è necessario garantire che siano stati presi i passi necessari per una corretta calibrazione del sensore.

Il profilo di calibrazione include gli offset del sensore e il raggio del sensore. Il sistema host può leggere gli offset e il raggio solo dopo che è stata raggiunta una calibrazione completa e la modalità operativa è stata cambiata in CONFIG\_MODE. Fare riferimento ai registri degli offset del sensore e del raggio del sensore.

È importante utilizzare gli offset corretti e il raggio del sensore corrispondente. Offset errati possono comportare dati di orientamento non affidabili, anche a un livello di accuratezza di calibrazione 3. Per impostare il profilo di calibrazione, è necessario seguire i seguenti passaggi:

1. Selezionare la modalità operativa CONFIG\_MODE.
2. Scrivere i dati corrispondenti degli offset del sensore e del raggio.
3. Cambiare la modalità operativa in modalità di fusione (fusion mode).

Abbiamo provato a scrivere degli offset per fare una calibrazione manuale ma non siamo riusciti ad ottenere i risultati sperati, nella successiva sezione vi è il test effettuato.

### 3.7.8 Test calibrazione manuale

Abbiamo anche provato a fare la calibrazione inserendo i valori degli OFFSET nei registri ma senza risultato, sono state create due funzioni specifiche, "calibrateAccel" e "setAccelCalibration" per ottenere degli offset e per inserirli nel registro corretto.

```
// Ottiene i dati di calibrazione del sensore BNO055
bno055_calibration_data_t bno055_getCalibrationData() {
    bno055_calibration_data_t calData;
    uint8_t buffer[22];
    bno055_opmode_t operationMode = bno055_getOperationMode();
    bno055_setOperationModeConfig();
    bno055_setPage(0);

    bno055_readData(BNO055_ACC_OFFSET_X_LSB, buffer, 22);

    // Assume un processore little-endian
    memcpy(&calData.offset.accel, buffer, 6);
    memcpy(&calData.offset.mag, buffer + 6, 6);
    memcpy(&calData.offset.gyro, buffer + 12, 6);
    memcpy(&calData.radius.accel, buffer + 18, 2);
    memcpy(&calData.radius.mag, buffer + 20, 2);

    bno055_setOperationMode(operationMode);

    return calData;
}

// Imposta i dati di calibrazione per il sensore BNO055
void bno055_setCalibrationData(bno055_calibration_data_t calData) {
    uint8_t buffer[22];
    bno055_opmode_t operationMode = bno055_getOperationMode();
    bno055_setOperationModeConfig();
    bno055_setPage(0);

    // Assume un processore little-endian
    memcpy(buffer, &calData.offset.accel, 6);
    memcpy(buffer + 6, &calData.offset.mag, 6);
    memcpy(buffer + 12, &calData.offset.gyro, 6);
    memcpy(buffer + 18, &calData.radius.accel, 2);
    memcpy(buffer + 20, &calData.radius.mag, 2);

    for (uint8_t i=0; i < 22; i++) {
        bno055_writeData(BNO055_ACC_OFFSET_X_LSB+i, buffer[i]);
    }

    bno055_setOperationMode(operationMode);
}
```

Listing 3.28: Codice nella libreria

```

// Raccoglie dati per offset
bno055_vector_xyz_int16_t calibrateAccel() {
    double xSum = 0, ySum = 0, zSum = 0;

    // Raccogli i dati di campionamento
    for (int i = 0; i < 100; i++) {
        bno055_vector_t accelData = bno055_getVectorAccelerometer();
        // Legge i dati dell'accelerometro
        xSum += accelData.x;
        ySum += accelData.y;
        zSum += accelData.z;
        HAL_Delay(100);
    }

    // Calcola la media
    bno055_vector_xyz_int16_t offset;
    offset.x = xSum / 100;
    offset.y = ySum / 100;
    offset.z = (zSum / 100) - 9.81;

    return offset;
}

// Imposta gli offset ottenuti dalla funzione precedente
nell'accelerometro
void setAccelCalibration(bno055_vector_xyz_int16_t offset) {
    bno055_calibration_data_t calData;

    // Imposta gli offset calcolati
    calData.offset.accel.x = offset.x;
    calData.offset.accel.y = offset.y;
    calData.offset.accel.z = offset.z;

    // Altri valori di offset e raggio possono essere impostati a 0
    o a valori preesistenti
    calData.offset.gyro = (bno055_vector_xyz_int16_t){0};
    calData.offset.mag = (bno055_vector_xyz_int16_t){0};
    calData.radius.accel = 0;
    calData.radius.mag = 0;

    // Utilizza i valori calcolati per impostare i dati di
    calibrazione
    bno055_setCalibrationData(calData);
}

```

Listing 3.29: Codice nella libreria

```

// Imposta la modalita operativa del BNO055 per utilizzare solo
// l'accelerometro.
// Questo e utile per la calibrazione dell'accelerometro in
// isolamento dagli altri sensori.
bno055_setOperationModeACCONLY();

// Calibra l'accelerometro. Questa funzione calcola gli offset
// dell'accelerometro
// raccogliendo campioni di dati, calcolando la media e restituendo
// il risultato.
// Questo aiuta a compensare eventuali errori sistematici o bias nei
// dati dell'accelerometro.
bno055_vector_xyz_int16_t accelOffset = calibrateAccel();

// Imposta i dati di calibrazione dell'accelerometro nel BNO055
// usando gli offset calcolati.
// Questo aggiorna i valori di calibrazione interni del BNO055 in
// modo che utilizzi
// i nuovi offset per tutte le future misurazioni dell'accelerometro.
setAccelCalibration(accelOffset);

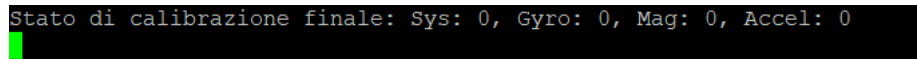
// Dopo la calibrazione, ottiene lo stato di calibrazione attuale
// del BNO055.
// Questo include lo stato di calibrazione per il sistema, il
// giroscopio, il magnetometro
// e l'accelerometro, fornendo un feedback su quanto bene ciascun
// sensore e calibrato.
bno055_calibration_state_t finalCalState =
    bno055_getCalibrationState();

// Stampa lo stato di calibrazione finale. Questo e utile per il
// debug e per verificare che
// la calibrazione sia stata eseguita correttamente. Se la
// calibrazione e riuscita, ci si aspetterebbe
// che i valori di calibrazione per l'accelerometro (e forse altri
// sensori, a seconda della loro
// configurazione e utilizzo) siano migliorati.
printf("Stato di calibrazione finale: Sys: %d, Gyro: %d, Mag: %d,
    Accel: %d\r\n",
        finalCalState.sys, finalCalState.gyro, finalCalState.mag,
        finalCalState.accel);

```

Listing 3.30: Codice da inserire nel main

Questo è l'output che si ottiene dal codice appena mostrato, tutti gli stati rimangono a zero, bisognerebbe capire se siano stati settati i nuovi offset, per questo abbiamo utilizzato il debug.



```

Stato di calibrazione finale: Sys: 0, Gyro: 0, Mag: 0, Accel: 0

```

Figure 3.15: Risultato del codice

Utilizzando il debug siamo arrivati a questo punto del codice. Il programma deve impostare i dati presi da *calibrateAccel()* memorizzati in *accelOffset*. È da notare che i valori sono tagliati in quanto *bno055\_calibration\_state\_t* è una struttura di interi.

```

170 bno055_setOperationModeACCONLY();
171 bno055_vector_xyz_int16_t accelOffset = calibrateAccel();
172 setAccelCalibration(accelOffset);
173
174 bno055_calibration_state_t finalCalState = bno055_getCalibrationState();
175 printf("Stato di calibrazione finale: Sys: %d, Gyro: %d, Mag: %d, Accel: %d\r\n",
176        finalCalState.sys, finalCalState.gyro, finalCalState.mag, finalCalState.accel);
177

```

>	myAxisMap	bno055_axis_map_t	{...}
▼	accelOffset	bno055_vector_xyz_int16_t	{...}
	x	int16_t	0
	y	int16_t	0
	z	int16_t	-19
▼	finalCalState	bno055_calibration_state_t	{...}
	sys	uint8_t	0 '\0'
	gyro	uint8_t	0 '\0'
	mag	uint8_t	0 '\0'
	accel	uint8_t	0 '\0'

In figura si possono vedere i valori precedenti.

Name	Type	Value
▼ offset	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	-19
▼ calData	bno055_calibration_data_t	{...}
▼ offset	bno055_calibration_offset_t	{...}
> gyro	bno055_vector_xyz_int16_t	{...}
> mag	bno055_vector_xyz_int16_t	{...}
▼ accel	bno055_vector_xyz_int16_t	{...}
x	int16_t	-27526
y	int16_t	-16313
z	int16_t	2618
▼ radius	bno055_calibration_radius_t	{...}
mag	uint16_t	41943
accel	uint16_t	15728

Questi valori poi vengono sostituiti.

```
void setAccelCalibration(bno055_vector_xyz_int16_t offset) {
    bno055_calibration_data_t calData;

    // Imposta gli offset calcolati
    calData.offset.accel.x = offset.x;
    calData.offset.accel.y = offset.y;
    calData.offset.accel.z = offset.z;

    // Altri valori di offset e raggio possono essere impostati a 0 o a valori
    calData.offset.gyro = (bno055_vector_xyz_int16_t){0};
    calData.offset.mag = (bno055_vector_xyz_int16_t){0};
    calData.radius.accel = 0;
    calData.radius.mag = 0;

    // Utilizza i valori calcolati per impostare i dati di calibrazione
    bno055_setCalibrationData(calData); // Assumi che questa funzione scriva i
}
```

Name	Type	Value
offset	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	-19
calData	bno055_calibration_data_t	{...}
offset	bno055_calibration_offset_t	{...}
gyro	bno055_vector_xyz_int16_t	{...}
mag	bno055_vector_xyz_int16_t	{...}
accel	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	-19
radius	bno055_calibration_radius_t	{...}
mag	uint16_t	41943
accel	uint16_t	15728

Poi entrando nella funzione *bno055\_setCalibrationData* viene messa la config mode per poter modificare i registri.

```
// Imposta i dati di calibrazione per il sensore BNO055
void bno055_setCalibrationData(bno055_calibration_data_t calData) {
    uint8_t buffer[22];
    bno055_opmode_t operationMode = bno055_getOperationMode();
    bno055_setOperationModeConfig();
    bno055_setPage(0);

    // Assume un processore little-endian
    memcpy(buffer, &calData.offset.accel, 6);
    memcpy(buffer + 6, &calData.offset.mag, 6);
    memcpy(buffer + 12, &calData.offset.gyro, 6);
    memcpy(buffer + 18, &calData.radius.accel, 2);
    memcpy(buffer + 20, &calData.radius.mag, 2);

    for (uint8_t i=0; i < 22; i++) {
        bno055_writeData(BNO055_ACC_OFFSET_X_LSB+i, buffer[i]);
    }

    bno055_setOperationMode(operationMode);
}
```

▼ calData	bno055_calibration_data_t	{...}
▼ offset	bno055_calibration_offset_t	{...}
> gyro	bno055_vector_xyz_int16_t	{...}
▼ mag	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	0
> accel	bno055_vector_xyz_int16_t	{...}
▼ radius	bno055_calibration_radius_t	{...}
mag	uint16_t	0
accel	uint16_t	0
> buffer	uint8_t [22]	0x10047f68
operationMode	bno055_opmode_t	BNO055_OPERATION_MODE...

In calData ci sono i nuovi offset da impostare, li mette nel buffer e poi con il ciclo for sposta i valori del buffer nel registro corretto.

▼ offset	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	-19
▼ calData	bno055_calibration_data_t	{...}
▼ offset	bno055_calibration_offset_t	{...}
▼ gyro	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	0
▼ mag	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	0
▼ accel	bno055_vector_xyz_int16_t	{...}
x	int16_t	0
y	int16_t	0
z	int16_t	-19
▼ radius	bno055_calibration_radius_t	{...}
mag	uint16_t	0
accel	uint16_t	0

il registro che ottiene lo stato dei sensori non viene aggiornato con le modalità non fusion, facendo una prova identica ma mettendo la modalità operativa NDOF funziona con i nuovi offset (ma sono peggiori di quelli che prenderebbe in automatico) quindi per ora consigliamo di utilizzare l'imu in modalità Fusion in modo che si autocalibri automaticamente.



## 3.8 Dati in uscita

A seconda della modalità operativa selezionata, il dispositivo restituisce dati che possono risultare grezzi (ottenuti senza algoritmo di fusione) o elaborati (ottenuti tramite algoritmo di fusione). Questa sezione descrive i dati di output per ciascuna modalità.

### 3.8.1 Selezione dell'unità di misura

Le unità di misura per i vari output del BNO055 possono essere configurate scrivendo nel registro UNIT\_SEL. Di seguito sono riportate le unità di misura disponibili:

Dato	Unità	Registro Valore
Accelerazione	$\text{m s}^{-2}$	[UNIT_SEL] : xxxxxx0b
	mg	[UNIT_SEL] : xxxxxx1b
Accelerazione Lineare Vettore Gravità	$\text{m s}^{-2}$	[UNIT_SEL] : xxxxxx0b
Forza Magnetica	Micro Tesla	NA
Velocità Angolare	Dps	[UNIT_SEL] : xxxxxx0xb
	Rps	[UNIT_SEL] : xxxxxx1xb
Angoli di Eulero	Gradi	[UNIT_SEL] : xxxxx0xxb
	Radiani	[UNIT_SEL] : xxxxx1xxb
Quaternione	unità Quaternion	NA
Temperatura	$^{\circ}\text{C}$	[UNIT_SEL] : xxx0xxxxb
	$^{\circ}\text{F}$	[UNIT_SEL] : xxx1xxxxb

**Table 3.16:** *Selezione dell'unità*

### 3.8.2 Formato dei dati di uscita

Il formato dei dati di uscita può essere selezionato scrivendo nel registro UNIT\_SEL. Questo permette di cambiare tra la definizione di orientamento dei sistemi operativi Windows e Android.

Parametro	Valori	Registro Valore
Formato dati di uscita	Windows	[UNIT_SEL] : 0xxxxxxx
	Android	[UNIT_SEL] : 1xxxxxxx

**Table 3.17:** *Formato dati di fusione*

I formati di dati di uscita si basano sulla seguente convenzione di angoli di rotazione per roll, pitch e yaw:

Angolo di rotazione	Android	Windows
Pitch	$180^{\circ}$ a $-180^{\circ}$ (oraio)	$-180^{\circ}$ a $180^{\circ}$ (orario)
Roll	$-90^{\circ}$ a $90^{\circ}$	$-90^{\circ}$ a $90^{\circ}$
Yaw	$0^{\circ}$ a $360^{\circ}$ (orario)	$0^{\circ}$ a $360^{\circ}$ (orario)

**Table 3.18:** *Convenzioni angoli di rotazione*

### 3.8.3 UNIT\_SEL

Il registro UNIT\_SEL è un registro di configurazione utilizzato per selezionare le unità di misura per i dati.

Inoltre è possibile selezionare il formato di output dei dati scrivendo nel registro. Ciò consente all'utente di passare tra le definizioni di orientamento descritte dai sistemi operativi Windows e Android.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
<b>Access</b>	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
<b>Reset</b>	1	0	0	0	0	0	0	0
<b>Content</b>	ORI_Android_Windows	reserved		TEMP_Unit	reserved	EUL_Unit	GYR_Unit	ACC_Unit

DATA	bits	Description
ORI_Android_Windows	7	Read: Current selected orientation mode Write: Select orientation mode 0: Windows orientation 1: Android orientation See section 3.6.2 for more details
TEMP_Unit	5	Read: Current selected temperature units Write: Select temperature units 0: Celsius 1: Fahrenheit See section 3.6.1 for more details
EUL_Unit	3	Read: Current selected Euler units Write: Select Euler units 0: Degrees 1: Radians See section 3.6.1 for more details
GYR_Unit	2	Read: Current selected angular rate units Write: Select angular rate units 0: dps 1: rps See section 3.6.1 for more details
ACC_Unit	1	Read: Current selected acceleration units Write: Select acceleration units 0: m/s <sup>2</sup> 1: mg See section 3.6.1 for more details

```
void bno055_configureUnits(0, 0, 0, 0, 1); //Valori default
//Per modificare i parametri cambiare tra 0 e 1 a seconda dell'unita
//di misura che si vuole, la corrispondenza dei valori e' quella
//della tabella sopra (2.5.3 UNIT_SEL)
```

Listing 3.31: Codice da inserire nel main

```

void bno055_configureUnits(uint8_t accelUnit, uint8_t gyroUnit,
uint8_t eulerUnit, uint8_t tempUnit, uint8_t orientMode) {
    uint8_t unitConfig = 0;

    // Configura l'unita di accelerazione
    if (accelUnit == 0) { //ms^2
        unitConfig |= 0; // xxxxxxx0b
    } else if (accelUnit == 1) { // mg
        unitConfig |= 1; // xxxxxxx1b
    } else {
        printf("Errore: Acceleration Unit non valida\r\n");
        return;
    }

    // Configura l'unita di velocita angolare
    if (gyroUnit == 0) { // Dps
        unitConfig |= (0 << 1); // xxxxxx0xb
    } else if (gyroUnit == 1) { // Rps
        unitConfig |= (1 << 1); // xxxxxx1xb
    } else {
        printf("Errore: Gyro Unit non valida\r\n");
        return;
    }

    // Configura l'unita degli angoli di Eulero
    if (eulerUnit == 0) { // Gradi
        unitConfig |= (0 << 2); // xxxxx0xb
    } else if (eulerUnit == 1) { // Radianti
        unitConfig |= (1 << 2); // xxxxx1xb
    } else {
        printf("Errore: Euler Unit non valida\r\n");
        return;
    }

    // Configura l'unita di temperatura
    if (tempUnit == 0) { // gradi celsius
        unitConfig |= (0 << 4); // xxx0xxxxb
    } else if (tempUnit == 1) { // gradi fahrenheit
        unitConfig |= (1 << 4); // xxx1xxxxb
    } else {
        printf("Errore: Temperature Unit non valida\r\n");
        return;
    }

    // Configura la modalita di orientamento
    if (orientMode == 0) { // Windows
        unitConfig |= (0 << 7); // 0xxxxxxx
    } else if (orientMode == 1) { // Android
        unitConfig |= (1 << 7); // 1xxxxxxx
    } else {
        printf("Errore: Orientation Mode non valida\r\n");
        return;
    }

    bno055_setOperationModeConfig();
    bno055_writeData(BNO055_UNIT_SEL, unitConfig);
}

```

Listing 3.32: Codice della libreria

### 3.8.4 Tassi di Dati in Uscita per la Fusione

Nella seguente tabella sono elencate le frequenze output dei dati di ogni sensore in base alla modalità.

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 3.16:** *Tassi di Dati in Uscita per la Fusione*

## 3.9 Interrupt

### 3.9.1 Interrupt PIN

INT è configurato come pin di interruzione per segnalare un'interruzione all'host. Il trigger di interrupt è configurato come raising edge ed è agganciato al pin INT. Una volta che si verifica un'interruzione, il pin INT viene impostato su alto e rimarrà alto fino a quando non verrà ripristinato dall'host [11]. Questo può essere fatto impostando RST\_INT nel registro SYS\_TRIGGER grazie alla funzione mostrata di seguito.

```
bno055_writeData(BNO055_SYS_TRIGGER, 0x40); //reset int
// 0x40 perche bisogna impostare il sesto bit del registro partendo
da 0
```

Listing 3.33: Codice da inserire nel while

La funzione dovrebbe essere chiamata all'inizio del while, dopo aver settato la flag del ciclo a zero. Successivamente gli interrupt possono essere abilitati impostando il bit corrispondente nel registro di abilitazione degli interrupt (INT\_EN) e disabilitati quando viene azzerato. Di seguito viene mostrata la tabella del registro corrispondente.

DATA	bits	Description
ACC_NM	7	Enable and disable of Accelerometer no motion or slow motion interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
ACC_AM	6	Enable and disable of Accelerometer any motion interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
ACC_HIGH_G	5	Enable and disable of Accelerometer high-g interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
GYR_DRDY <sup>6</sup>	4	Enable and disable of Gyroscope data ready interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
GYR_HIGH_RATE	3	Enable and disable of gyroscope high rate interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
GYRO_AM	2	Enable and disable of gyroscope any motion interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
MAG_DRDY <sup>6</sup>	1	Enable and disable of magnetometer data ready interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt
ACC_BSX_DRDY <sup>6</sup>	0	Enable or disable of Accelerometer or BSX data ready interrupt Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable interrupt

Figure 3.17: Registro Int\_EN

```

void bno055_set_int_en(uint8_t setAccNM, uint8_t setAccAM, uint8_t
setAccHighG, uint8_t setGyrDRDY, uint8_t setGyrHighRate, uint8_t
setGyrAM, uint8_t setMagDRDY, uint8_t setAccBSXDRDY){
    bno055_setPage(1);
    uint8_t setInt_en = 0;
    // 0 disabilita l'interrupt, 1 lo abilita
    if(setAccNM == 0){
        setInt_en |= (0 << 7);
    }else{
        setInt_en |= (1 << 7);
    }
    if(setAccAM == 0){
        setInt_en |= (0 << 6);
    }else{
        setInt_en |= (1 << 6);
    }
    if(setAccHighG == 0){
        setInt_en |= (0 << 5);
    }else{
        setInt_en |= (1 << 5);
    }
    if(setGyrDRDY == 0){
        setInt_en |= (0 << 4);
    }else{
        setInt_en |= (1 << 4);
    }
    if(setGyrHighRate == 0){
        setInt_en |= (0 << 3);
    }else{
        setInt_en |= (1 << 3);
    }
    if(setGyrAM == 0){
        setInt_en |= (0 << 2);
    }else{
        setInt_en |= (1 << 2);
    }
    if(setMagDRDY == 0){
        setInt_en |= (0 << 1);
    }else{
        setInt_en |= (1 << 1);
    }
    if(setAccBSXDRDY == 0){
        setInt_en |= (0 << 0);
    }else{
        setInt_en |= (1 << 0);
    }
    bno055_setOperationModeConfig();
    bno055_writeData(BNO055_INT_EN, setInt_en);
    bno055_setPage(0);
}

```

Listing 3.34: Codice della libreria

Gli interrupt possono essere indirizzati al pin INT impostando il corrispondente bit di interrupt nel registro INT\_MSK. Anche per questo registro viene mostrata la tabella.

DATA	bits	Description
ACC_NM	7	Masking of Accelerometer no motion or slow motion interrupt, when enabled the interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
ACC_AM	6	Masking of Accelerometer any motion interrupt, when enabled the interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
ACC_HIGH_G	5	Masking of Accelerometer high-g interrupt, when enabled the interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
GYR_DRDY <sup>6</sup>	4	Masking of gyroscope data ready interrupt, when enabled together with interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
GYR_HIGH_RATE	3	Masking of gyroscope high rate interrupt, when enabled the interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
GYRO_AM	2	Masking of gyroscope any motion interrupt, when enabled the interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
MAG_DRDY <sup>6</sup>	1	Masking of magnetometer data ready interrupt, when enabled together with interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable
ACC_BSXD <sup>6</sup>	0	Masking of Accelerometer or BSX data ready interrupt, when enabled together with interrupt will update the INT_STA register and trigger a change on the INT pin, when disabled only the INT_STA register will be updated. Read: 1: Enabled / 0: Disabled Write: 1: Enable / 0: Disable

**Figure 3.18:** *Registro Int\_MSK*

Per motivi di spazio le successive funzioni vengono mostrate solo in forma di dichiarazione.

```
// dichiarazione delle funzioni
void bno055_set_int_msk(uint8_t setAccNM, uint8_t setAccAM, uint8_t
    setAccHighG, uint8_t setGyrDRDY, uint8_t setGyrHighRate, uint8_t
    setGyrAM, uint8_t setMagDRDY, uint8_t setAccBSXD6);

void bno055_get_int_msk();
```

**Listing 3.35:** Codice della libreria

Sono presenti anche i registri relativi agli interrupt dei sensori in relazione agli assi, ACC\_INT\_SETTINGS e GYR\_INT\_SETTINGS.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
<b>Access</b>	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
<b>Reset</b>	0	0	0	0	0	0	1	1
<b>Content</b>	HG_Z_AXIS S	HG_Y_AXIS S	HG_X_AXIS S	AM/NM_Z_ AXIS	AM/NM_Y_ _AXIS	AM/NM_X_ _AXIS	AM_DUR <1:0>	

DATA	bits	Description
HG_Z_AXIS	7	Select which axis of the accelerometer is used to trigger a high-G interrupt 1: Enabled; 0: Disabled
HG_Y_AXIS	6	Select which axis of the accelerometer is used to trigger a high-G interrupt 1: Enabled; 0: Disabled
HG_X_AXIS	5	Select which axis of the accelerometer is used to trigger a high-G interrupt 1: Enabled; 0: Disabled
AM/NM_Z_AXIS	4	Select which axis of the accelerometer is used to trigger a any motion or no motion interrupt 1: Enabled; 0: Disabled
AM/NM_Y_AXIS	3	Select which axis of the accelerometer is used to trigger a any motion or no motion interrupt 1: Enabled; 0: Disabled
AM/NM_X_AXIS	2	Select which axis of the accelerometer is used to trigger a any motion or no motion interrupt 1: Enabled; 0: Disabled
AM_DUR <1:0>	<1:0>	Any motion interrupt triggers if [AM_DUR<1:0>+1] consecutive data points are above the any motion interrupt threshold define in ACC_AM_THRES register

**Figure 3.19:** *Registro ACC\_INT\_Settings*

```
// dichiarazione delle funzioni
void bno055_set_acc_int_settings(uint8_t HG_Z_axis, uint8_t
    HG_Y_axis, uint8_t HG_X_axis, uint8_t AM_NM_Z_axis, uint8_t
    AM_NM_Y_axis, uint8_t AM_NM_X_axis, uint8_t AM_DUR1, uint8_t
    AM_DUR0);

void bno055_get_acc_int_settings();
```

**Listing 3.36:** Codice della libreria



	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
<b>Access</b>	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
<b>Reset</b>	0	0	0	0	0	0	0	0
<b>Content</b>	HR_FILT	AM_FILT	HR_Z_AXIS	HR_Y_AXIS	HR_X_AXIS	AM_Z_AXIS	AM_Y_AXIS	AM_X_AXIS

DATA	bits	Description
HR_FILT	7	'1' ('0') selects unfiltered (filtered) data for high rate interrupt
AM_FILT	6	'1' ('0') selects unfiltered (filtered) data for any motion interrupt
HR_Z_AXIS	5	1' ('0') enables (disables) high rate interrupt for z-axis
HR_Y_AXIS	4	1' ('0') enables (disables) ) high rate interrupt for y-axis
HR_X_AXIS	3	1' ('0') enables (disables) ) high rate interrupt for x-axis
AM_Z_AXIS	2	1' ('0') enables (disables) any motion interrupt for z-axis
AM_Y_AXIS	1	1' ('0') enables (disables) any motion interrupt for y-axis
AM_X_AXIS	0	1' ('0') enables (disables) any motion interrupt for x-axis

**Figure 3.20:** *Registro GYR\_INT\_Settings*

```
// dichiarazione delle funzioni
void bno055_set_gyr_int_settings(uint8_t HR_Filt, uint8_t AM_Filt,
    uint8_t HR_Z_Axis, uint8_t HR_Y_Axis, uint8_t HR_X_Axis, uint8_t
    AM_Z_Axis, uint8_t AM_Y_Axis, uint8_t AM_X_Axis);

void bno055_get_gyr_int_settings();
```

**Listing 3.37: Codice della libreria**

È stata creata nella libreria una funzione che abilita gli interrupt settando tutti i bit dei registri citati ad 1.

Mettendo i bit di ACC\_INT\_SETTINGS e GYR\_INT\_SETTINGS ad uno significa che sono stati abilitati tutti gli assi.

Ad esempio, è qui mostrato il codice per abilitare l'interrupt dei 3 assi dell'accelerometro con condizione "any motion" e senza tenere in considerazione la gravità. Nel caso in cui il dispositivo risulti fermo, indipendentemente dall'assetto con il quale si presenta, non verrà mai generato alcun interrupt. Qualora, invece, il dispositivo risulti soggetto ad accelerazioni oltre la gravità, viene generato un interrupt.

```
bno055_set_int_en(0, 1, 0, 0, 0, 0, 0, 0);
bno055_set_int_msk(0, 1, 0, 0, 0, 0, 0, 0);
bno055_set_acc_int_settings(0, 0, 0, 1, 1, 1, 1, 1);
bno055_set_gyr_int_settings(0, 0, 0, 0, 0, 0, 0, 0);

bno055_setOperationModeNDof(); //importante che la modalita
operativa sia inizializzata successivamente
```

Listing 3.38: Codice da inserire nel main

```
if(flag_BNO055_Data_Ready==1){
    flag_BNO055_Data_Ready = 0;
    bno055_writeData(BNO055_SYS_TRIGGER, 0x40); //reset int

    bno055_calibration_state_t cal = bno055_getCalibrationState();
    quat = bno055_getVectorQuaternion();

    printf("QUAT - x: %+2.2f | y: %+2.2f | z: %+2.2f | w: %+2.2f |
    %+2.2d\r\n", quat.x, quat.y, quat.z, quat.w, cal.sys);

    HAL_Delay(10);
}
```

Listing 3.39: Codice da inserire nel while

Non è stato approfondito invece il registro INT\_STA che riguarda lo stato degli interrupt dei vari sensori né le impostazioni riguardanti gli interrupt né il paragrafo relativo alle impostazioni degli interrupt "Interrupt Settings" del datasheet.

È stato invece testato che modificando le configurazioni dei sensori gli interrupt si aggiornavano di conseguenza, quindi ad esempio impostando un G range dell'accelerometro a 2 il sensore prende dati con più sensibilità rispetto ad un G range di 8.

## 3.10 Test di autoverifica

### 3.10.1 POST - auto test al momento dell'accensione

Durante l'avvio del dispositivo, viene eseguito un auto-test al momento dell'accensione. Questa caratteristica verifica che i sensori connessi e il microcontrollore stiano rispondendo e funzionando correttamente. Vengono eseguiti i seguenti test:

Componenti	Tipo di Test
Accelerometro	Verifica ID chip
Magnetometro	Verifica ID chip
Giroscopio	Verifica ID chip
Microcontrollore	Auto-Test di Costruzione in Memoria

**Table 3.19:** *Auto-Test al Momento dell'Accensione*

I risultati del POST sono memorizzati nel registro ST\_RESULT, dove un bit impostato indica che il test è stato superato e un bit cancellato indica che l'auto-test è fallito.

```
// Ottiene i risultati del test di autodiagnosi iniziale dal sensore BN0055
bno055_self_test_result_t bno055_getSelfTestResult_start() {
    bno055_setPage(0);
    uint8_t tmp;
    bno055_self_test_result_t res = {
        .mcuState = 0,
        .gyrState = 0,
        .magState = 0,
        .accState = 0};
    bno055_readData(BN0055_ST_RESULT, &tmp, 1);
    res.mcuState = (tmp >> 3) & 0x01;
    res.gyrState = (tmp >> 2) & 0x01;
    res.magState = (tmp >> 1) & 0x01;
    res.accState = (tmp >> 0) & 0x01;
    return res;
}
```

**Listing 3.40:** Codice nella libreria

Questa funzione ottiene dal registro ST\_RESULT i valori associati ai sensori, il cui significato è indicato dalla seguente tabella.

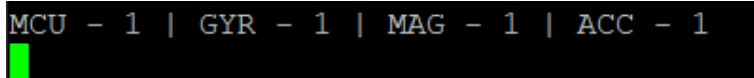
Bit	Componente	Risultato Auto-Test
3	Microcontrollore	1 indica test superato; 0 indica test fallito
2	Giroscopio	1 indica test superato; 0 indica test fallito
1	Magnetometro	1 indica test superato; 0 indica test fallito
0	Accelerometro	1 indica test superato; 0 indica test fallito

**Table 3.20:** *Risultati Auto-Test (ST\_RESULT)*

La modalità di utilizzo è la seguente:

```
bno055_setOperationModeConfig();  
  
bno055_self_test_result_t res = bno055_getSelfTestResult_start();  
  
printf("MCU - %u | GYR - %u | MAG - %u | ACC - %u\r\n",  
       res.mcuState, res.gyrState, res.magState, res.accState);
```

Listing 3.41: Codice da inserire nel main



```
MCU - 1 | GYR - 1 | MAG - 1 | ACC - 1
```

Figure 3.21: Risultato del codice

I valori sono tutti a 1, ciò significa che i sensori all'avvio del dispositivo sono funzionanti.

### 3.10.2 BIST - auto test integrato

L'host può innescare un auto-test dalla MODALITÀ CONFIG. Il test può essere attivato impostando il bit SELF\_TEST nel registro SYS\_TRIGGER; i risultati sono memorizzati nel registro ST\_RESULT. Durante l'esecuzione dell'auto-test, tutte le altre funzionalità sono messe in pausa. Si raccomanda di innescare l'auto-test solo dalla modalità di configurazione.

Durante l'auto-test:

- Quando si innesta il BIST, lo stato del sistema SYS\_STATUS cambia in "esecuzione dell'auto-test", ma lo stato del sistema SYS\_STATUS sarà aggiornato solo in caso di fallimento, non in caso di successo.
- Per sapere se il BIST è riuscito o fallito, dovresti:
  1. Innescare BIST
  2. Attendere 400ms
  3. Leggere il registro SYS\_ERROR (0x3A):
    - SYS\_ERROR rimarrà a 0 in caso di successo (0 = Nessun errore)
    - SYS\_ERROR mostrerà 3 in caso di fallimento dell'auto-test (3 = Risultato auto-test fallito)
- In caso di auto-test fallito (SYS\_ERROR = 3), puoi vedere quale sensore ha fallito leggendo il registro ST\_RESULT (0x36) relativo al bit corrispondente. Il bit corrispondente sarà impostato su '1' se l'auto-test del sensore è stato riuscito, ma mostrerà '0' se l'auto-test è fallito.

<b>Componenti</b>	<b>Tipo di Test</b>
Accelerometro	Auto-test integrato
Magnetometro	Auto-test integrato
Giroscopio	Auto-test integrato
Microcontrollore	Non eseguito

**Table 3.21:** *Auto-Test Integrato*

<b>Codice SYS_STATUS</b>	<b>Significato</b>
0	Sistema inattivo
1	Errore di sistema
2	Inizializzazione periferiche
3	Inizializzazione sistema
4	Esecuzione auto-test
5	Algoritmo di fusione dei sensori in esecuzione
6	Sistema in esecuzione senza algoritmo di fusione

**Table 3.22:** *Codici di Stato del Sistema (SYS\_STATUS)*

<b>Codice SYS_ERR</b>	<b>Significato</b>
0	Nessun errore
1	Errore di inizializzazione periferica
2	Errore di inizializzazione sistema
3	Risultato auto-test fallito
4	Valore mappa registro fuori portata
5	Indirizzo mappa registro fuori portata
6	Errore scrittura mappa registro
7	Modalità basso consumo non disponibile per la modalità operativa selezionata
8	Modalità di alimentazione accelerometro non disponibile
9	Errore di configurazione dell'algoritmo di fusione
10	Errore di configurazione sensore

**Table 3.23:** *Codici di Errore del Sistema (SYS\_ERR)*

```

// Ottiene i risultati del test di autodiagnosi dal sensore BN0055
bno055_self_test_result_t bno055_getSelfTestResult() {
    bno055_setPage(0);
    // Legge il valore attuale del registro SYS_TRIGGER
    uint8_t sys_trigger_value;
    bno055_readData(BN0055_SYS_TRIGGER, &sys_trigger_value, 1);
    // Stampa il valore del bit meno significativo prima della
    scrittura
    printf("Bit meno significativo prima della scrittura: %d\r\n",
        sys_trigger_value & 0x01);
    // Imposta il bit del Self-Test
    sys_trigger_value |= 0x01; // Bit 0: Self-Test
    // Stampa il valore del bit meno significativo dopo la scrittura
    printf("Bit meno significativo dopo la scrittura: %d\r\n",
        sys_trigger_value & 0x01);
    // Scrive il valore modificato nel registro SYS_TRIGGER
    bno055_writeData(BN0055_SYS_TRIGGER, sys_trigger_value);

    bno055_self_test_result_t res = bno055_getSelfTestResult_start();

    return res;
}

```

Listing 3.42: Codice nella libreria

```

// Ottiene lo stato del sistema dal sensore BN0055
uint8_t bno055_getSystemStatus() {
    bno055_setPage(0);
    uint8_t tmp;
    bno055_readData(BN0055_SYS_STATUS, &tmp, 1);
    return tmp;
}

// Ottiene il codice di errore del sistema dal sensore BN0055
uint8_t bno055_getSystemError() {
    bno055_setPage(0);
    uint8_t tmp;
    bno055_readData(BN0055_SYS_ERR, &tmp, 1);
    return tmp;
}

```

Listing 3.43: Codice nella libreria

```

bno055_setOperationModeConfig();

bno055_self_test_result_t res = bno055_getSelfTestResult();

printf("MCU - %u | GYR - %u | MAG - %u | ACC - %u\r\n",
    res.mcuState, res.gyrState, res.magState, res.accState);

HAL_Delay(600);

uint8_t tmp = bno055_getSystemStatus();
uint8_t err = bno055_getSystemError();
printf("STATE - %u | ERR - %u\r\n", tmp, err);

HAL_Delay(600);

```

Listing 3.44: Codice da inserire nel main

```

Bit meno significativo prima della scrittura: 0
Bit meno significativo dopo la scrittura: 1
MCU - 0 | GYR - 0 | MAG - 0 | ACC - 0
STATE - 4 | ERR - 0

```

**Figure 3.22:** Risultato del codice

Attualmente si riscontra un problema nel riconoscimento dei sensori funzionanti, evidenziato nell'immagine 3.22.

Il sistema è in modalità Auto-Test (visibile da STATE - 4) e non mostra alcun errore (ERR - 0), come indicato nelle tabelle 3.23 e 3.23. Tuttavia, nonostante ciò, i sensori MCU, GYR, MAG e ACC risultano tutti a 0.

Nelle prime due righe dell'output sono presenti due stampe per verificare se il registro fosse stato cambiato per attivare la modalità di test automatico, e infatti lo è, come dimostra il valore STATE - 4. Tuttavia, i sensori rimangono a 0.

Il registro ST\_RESULT serve a identificare quale sensore non è operativo nel caso in cui SYS\_ERR e SYS\_STATUS segnalino un errore. Tuttavia, se non ci sono errori, il registro ST\_RESULT non fornisce ulteriori informazioni e potrebbe essere trascurato. Ciò che è più importante è garantire il corretto funzionamento di SYS\_ERR e SYS\_STATUS.

Quest'ultima cosa è stata provata in molti modi durante la realizzazione della libreria, qui sotto un piccolo esempio:

```

bno055_setOperationModeAMG();
uint8_t tmp = bno055_getSystemStatus();
uint8_t err = bno055_getSystemError();
printf("STATE - %u | ERR - %u\r\n", tmp, err);

bno055_setOperationModeConfig();
uint8_t tmp2 = bno055_getSystemStatus();
uint8_t err2 = bno055_getSystemError();
printf("STATE - %u | ERR - %u\r\n", tmp2, err2);

```

**Listing 3.45:** Codice da inserire nel main

```

STATE - 6 | ERR - 0
STATE - 0 | ERR - 0

```

**Figure 3.23:** Risultato del codice

## 4 Interfacce digitali

Il BNO055 supporta due interfacce digitali per la comunicazione tra il dispositivo slave e host: I2C che supporta il protocollo HID-I2C e le modalità I2C Standard e Fast; e l'interfaccia UART.

L'interfaccia attiva è selezionata dallo stato dei pin di selezione del protocollo (PS1 e PS0), la Tabella 4.1 mostra la mappatura tra i pin di selezione del protocollo e la modalità di interfaccia selezionata.

PS1	PS0	Funzionalità
0	0	Interfaccia I2C Standard/Fast
0	1	HID su I2C
1	0	Interfaccia UART
1	1	Riservato

**Table 4.1:** *Mappatura dei pin di selezione del protocollo*

Non è consentito mantenere fluttuanti i pin di selezione del protocollo.

Entrambe le interfacce digitali condividono parzialmente gli stessi pin, la mappatura dei pin per ogni interfaccia è mostrata nella Tabella 4.2.

PIN	Interfacce I2C (PS1=0b0)	Interfaccia UART (PS1.PS0=0b10)
COM0	SDA	Tx
COM1	SCL	Rx
COM2	GNDIO	
COM3	Selezione dell'indirizzo I2C	

**Table 4.2:** *Mappatura dei pin di interfaccia digitale*

La seguente tabella mostra le specifiche elettriche dei pin di interfaccia:

Parameter	Symbol	Condition	Min	Typ	Max	Units
Pull-up Resistance, COM3 pin	$R_{up}$	Internal Pull-up Resistance to VDDIO	20	40	60	$k\Omega$
Input Capacitance	$C_{in}$			5	10	pF
I <sup>2</sup> C Bus Load Capacitance (max. drive capability)	$C_{I2C\_Load}$				400	pF

**Figure 4.1:** *Specifiche elettriche dei pin di interfaccia*



## 4.1 Protocollo I2C

Il bus I2C utilizza le linee di segnale SCL (= pin SCx, clock seriale) e SDA (= pin SDx, ingresso e uscita dati seriali). Entrambe le linee sono collegate esternamente a VDDIO tramite resistori di pull-up in modo che siano elevate quando il bus è libero. L'interfaccia IPC del BNO055 è compatibile con la specifica I2C UM10204 Rev. 03, 19 giugno 2007, (per maggiori dettagli consultare il manuale [12]).

Il BNO055 supporta la modalità standard e la modalità veloce I2C, è supportata solo la modalità indirizzo a 7 bit. L'interfaccia I2C del BNO055 utilizza lo stretching del clock. L'indirizzo I2C predefinito del dispositivo BNO055 è 0101001b (0x29). L'indirizzo alternativo 0101000b (0x28), in modalità I2C il pin di ingresso COM3 può essere utilizzato per selezionare tra l'indirizzo I2C primario e alternativo come mostrato nella successiva tabella.

Configurazione I2C	Stato_COM3	Indirizzo I2C
slave	High	0x29
slave	Low	0x28
HID-I2C	X	0x40

**Table 4.3:** Selezione indirizzo I2C

Nel datasheet questa parte viene approfondita con le specifiche temporali, il funzionamento generale, in scrittura e lettura.

### 4.1.1 HAL\_Error

Lavorando con il sensore, potrebbe capitare di ottenere un output di questo tipo durante l'esecuzione del codice, si tratta di un problema di collegamento.

```
HAL_I2C_STATE_RESET
HAL_I2C_Master_Transmit HAL_ERROR
HAL_I2C_Master_Transmit HAL_ERROR
HAL_I2C_ERROR_TIMEOUT
HAL_I2C_STATE_RESET
HAL_I2C_Master_Transmit HAL_ERROR
GYR - x: +69.00 | y: +256.00 | z: +1996.56 | +00
ACC - x: +11.04 | y: +40.96 | z: +319.45 | +00
MAG - x: +69.00 | y: +256.00 | z: +1996.56 | +00
QUAT - x: +0.25 | y: +1.95 | z: +0.13 | w: +0.07 | +00
```

**Figure 4.2:** HAL\_Error

Ci sono 2 cavi che riguardano l'I2C, il Serial CLock (SCL) e il Serial DAta (SDA), se l'errore è causato da un collegamento precario del cavo SDA, la soluzione è semplicemente riconnettere bene il cavo e il programma inizierà a stampare i dati corretti senza doverlo resettare, al contrario, se il problema è il cavo SCL sarà necessario un reset, nel nostro caso il pulsante RESET della board va benissimo.

## 4.2 Protocollo UART

Il BNO055 supporta l'interfaccia UART per la comunicazione. L'UART (Universal Asynchronous Receiver Transmitter) è un'interfaccia di comunicazione seriale asincrona che consente la trasmissione di dati tra due dispositivi utilizzando una linea di trasmissione e una linea di ricezione.

### Configurazione UART del BNO055

- Baud Rate: 115200 bps
- Formato dati: 8N1 (8 bit di dati, nessuna parità, 1 bit di stop)
- Lunghezza massima del pacchetto: 128 byte (sia per lettura che scrittura)

#### Register write

Command:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	.....	Byte (n+4)
Start Byte	Write	Reg addr	Length	Data 1	.....	Data n
0xAA	0x00	<..>	<..>	<..>	.....	<..>

Acknowledge Response:

Byte 1	Byte 2
Response Header	Status
0xEE	0x01: WRITE_SUCCESS 0x03: WRITE_FAIL 0x04: REGMAP_INVALID_ADDRESS 0x05: REGMAP_WRITE_DISABLED 0x06: WRONG_START_BYTE 0x07: BUS_OVER_RUN_ERROR 0x08: MAX_LENGTH_ERROR 0x09: MIN_LENGTH_ERROR 0x0A: RECEIVE_CHARACTER_TIMEOUT

#### Register read

Command:

Byte 1	Byte 2	Byte 2	Byte 3
Start Byte	Read	Reg addr	Length
0xAA	0x01	<..>	<..>

Read Success Response:

Byte 1	Byte 2	Byte 3	.....	Byte (n+2)
ResponseByte	length	Data 1	.....	Data n
0xBB	<..>			

Read Failure or Acknowledge Response:

Byte 1	Byte 2
Response Header	Status
0xEE	0x02: READ_FAIL 0x04: REGMAP_INVALID_ADDRESS 0x05: REGMAP_WRITE_DISABLED 0x06: WRONG_START_BYTE 0x07: BUS_OVER_RUN_ERROR 0x08: MAX_LENGTH_ERROR 0x09: MIN_LENGTH_ERROR 0x0A: RECEIVE_CHARACTER_TIMEOUT

Figure 4.3: Registro di scrittura e lettura UART

### 4.3 HID su I2C

HID su I2C è un protocollo standard per collegare dispositivi a host tramite I2C. Il principale vantaggio di HID è l'esistenza di driver generici per diversi dispositivi di input (come sensori) che possono essere utilizzati con sensori che implementano i corrispondenti profili HID ben definiti. HID su I2C descrive come vengono scambiati i messaggi (report ed eventi) tra il dispositivo e l'host. Una descrizione della struttura di questi report viene fornita dal dispositivo e letta dall'host durante l'inizializzazione del dispositivo all'avvio del sistema host. Per informazioni dettagliate su HID, consultare la documentazione HID su I2C di Microsoft.

## Conclusioni e sviluppi futuri

Il task assegnato era creare un manuale d'uso e una libreria operativa per l'imu BMO055. L'obiettivo è stato raggiunto, tuttavia bisogna specificare che alcune funzionalità meritano un ulteriore approfondimento.

Al momento la libreria è funzionale per:

- Cambio della modalità di alimentazione.
- Cambio modalità operativa.
- Rimappatura dei sensori.
- Configurazione dei sensori.
- Cambio dell'unità di misura dei dati in uscita.
- Abilitazione degli interrupt.
- Autotest di verifica dell'integrità del sensore.
- Calibrazione in modalità fusion.

Nella relazione vi sono i passi usati per la configurazione dell'imu, informazioni sulle funzionalità, esempi, test e un approfondimento riguardo i quaternioni e gli angoli di Eulero.

## Problemi aperti

- Nella parte dell'autotest BIST abbiamo riscontrato problematiche nell'impiego della funzione che deve fornire informazioni utili ad individuare i sensori danneggiati (spiegazione approfondita in sezione 3.10.2).
- La calibrazione manuale, con modalità operativa non-fusion, non ha restituito il risultato atteso (spiegazione approfondita in sezione 3.7.8).
- Gli interrupt potrebbero essere ulteriormente approfonditi, prendendo in considerazione anche la sezione "Interrupt Settings" presente nel datasheet (spiegazione approfondita in sezione 3.9).



## Bibliografia

- [1] A. Bonci, “Serial Communication Protocols, url =<https://lc.cx/I57lpp>.”
- [2] ivyknob, “BNO055 library to use with STM32 HAL, url =[https://github.com/ivyknob/bno055\\_stm32](https://github.com/ivyknob/bno055_stm32).”
- [3] B. Sensortec, “BNO055 datasheet, url =<https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf>.”
- [4] K. Townsend, “Adafruit BNO055 Absolute Orientation Sensor, url =<https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bno055-absolute-orientation-sensor.pdf>.”
- [5] Wikipedia, “Quaternions and spatial rotation, url =[https://en.wikipedia.org/wiki/Quaternions\\_and\\_spatial\\_rotation](https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation).”
- [6] G. G. Slabaugh, “Computing Euler angles from a rotation matrix, url =<https://eecs.qmul.ac.uk/gslabaugh/publications/euler.pdf>.”
- [7] Wikipedia, “Euler angles, url =[https://en.wikipedia.org/wiki/Euler\\_angles](https://en.wikipedia.org/wiki/Euler_angles).”
- [8] Wikipedia, “Conversion between quaternions and Euler angles, url =[https://en.wikipedia.org/wiki/Conversion\\_between\\_quaternions\\_and\\_Euler\\_angles](https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles).”
- [9] E. Bernardes and S. Viollet, “Quaternion to euler angles conversion: A direct, general and computationally efficient method,” *PLoS ONE*, vol. 17, 11 2022.
- [10] J. H, “Roll pitch yaw calculation, url =<https://lc.cx/mbXtc6>.”
- [11] R. Mischianti, “BNO055: modalità di alimentazione, accelerometro e interrupt di movimento – 4, url = <https://mischianti.org/it/bno055-modalita-di-alimentazione-accelerometro-e-interrupt-di-movimento-4/>.”
- [12] NXP, “UM10204 I2C-bus specification and user manual, url =<https://www.nxp.com/docs/en/user-guide/UM10204.pdf>.”