SENSORISTICA

BALLBOT

IMU6050 e Sensore corrente INA219

SUNTO
Acquisizione ed elaborazione dati dai sensori tramite protocollo IIC.

STUDENTI:
Olivieri Davide
Igor Nociaro
Matteo Giacomobono
HarmanDeep Kaur

LABORATORIO DI AUTOMAZIONE

Relatore :

Andrea Bonci

INDICE

INTRODUZIONE

COS'E' UN BALLBOT?

Un Ballbot è un robot mobile dinamicamente stabile progettato per bilanciarsi su una singola ruota sferica attraverso il suo unico punto di contatto con il terreno. Esso è omnidirezionale e quindi eccezionalmente agile e manovrabile rispetto agli altri veicoli terrestri. La sua stabilità dinamica consente una migliore navigabilità in ambienti stretti e affollati. Il suo funzionamento è basato sul principio di un pendolo invertito.

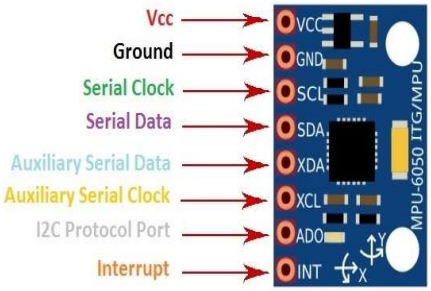
LAVORO DEL GRUPPO:

Uno dei maggiori problemi da risolvere in un sistema ad alta precisione come il Ballbot è quello di fornire al controllore dei dati con un'elevata precisione in modo da non portare il sistema ad un'instabilità.
Il lavoro del gruppo infatti è stato proprio incentrato sulla sensoristica del sistema in particolare l'acquisizione di dati da parte di due sensori: IMU6050 e INA219, e in seguito elaborarli in modo da trovare dei valori accurati da dare al controllore attraverso la comunicazione seriale IIC.

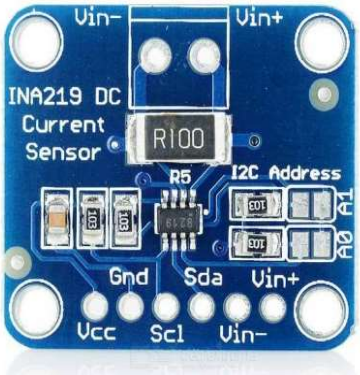
Tutte le sezioni sotto riportate presenteranno una spiegazione dettagliata sullo sviluppo del progetto dallo studio teorico dei componenti alla stesura del codice.

HARDWARE

Renesas Demonstration Kit YRDKRX63N a 100 pin, è la scheda di sviluppo su cui è montato il µ-controller RX63N ad alte prestazioni, per cui è possibile la programmazione, la codifica e il debugging mediante software Embedded nel nostro caso è stato utilizzato 'e2studio'.



MPU6050 è un Micro Electro-Mechanical Systems che consiste in un accelerometro a 3 assi e un giroscopio a 3 assi al suo interno. Necessari per misurare l'accelerazione, la velocità, l'orientamento, lo spostamento e molti altri parametri relativi al movimento di un sistema. Il sensore MPU6050 ha molte funzioni sul singolo chip. È costituito da un accelerometro MEMS, un giroscopio MEMS e un sensore di temperatura. Questo modulo è molto preciso durante la conversione di valori analogici in digitale perché ha un hardware convertitore analogico-digitale a 16 bit per ogni canale.



INA219 è un circuito integrato in grado di misurare l'assorbimento di corrente con 1% di precisione attraverso la resistenza di shunt trasmettendo i dati tramite protocollo IIC.

CENNI TEORICI IMU:

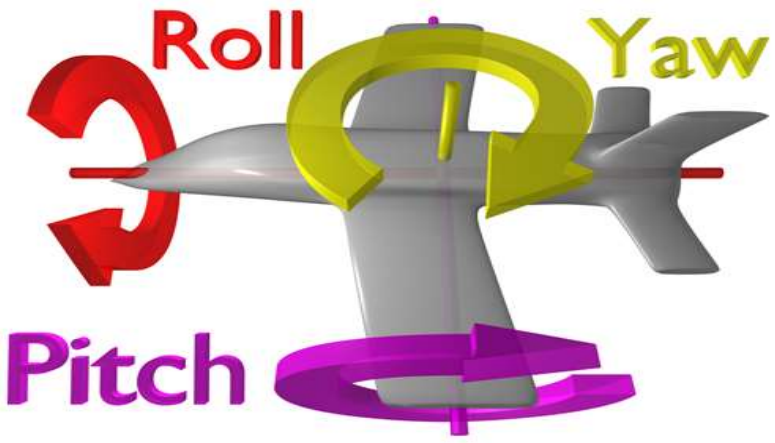
L'IMU comprende: ACCELLEROMETRO, GIROSCOPIO E magnetometro.

L'ACCELEROMETRO :

E' sostanzialmente un dispositivo in grado di rilevare o misurare un' accelerazione, ovvero una variazione di velocità nel tempo; può anche essere utilizzato come dispositivo per misurare l'inclinazione di un corpo rigido.

Un corpo può rotare nello spazio lungo 3 assi, facendo riferimento alla dinamica applicata ai velivoli si parla più propriamente di rollio, beccheggio e imbardata.

Per convenzione la rotazione del velivolo lungo l'**asse X** (in rosso) del velivolo si chiama rollio (**Roll**). La rotazione lungo **l'asse Y** del velivolo (in viola) si chiama beccheggio (**Pitch**) e la rotazione lungo **l'asse Z** del velivolo (in giallo) si chiama Imbardata (**Yaw**).



Immaginando ora che gli assi X e Y del velivolo si trovino sul piano dell'orizzonte e l'asse Z sia perpendicolare a questo piano. Definiamo con il termine di "orizzonte" il piano perpendicolare alla direzione del vettore rappresentato dalla forza di gravità.

Una rotazione intorno all'asse X : forma un angolo tra asse Y e orizzonte. Questo angolo che Y forma con l'orizzonte a causa della rotazione di X si chiama **angolo di rollio** (**Roll),** e viene generalmente indicato con la lettera greca **φ** (phi).

Una rotazione intorno all'asse Y : forma un angolo tra asse X e orizzonte. Questo angolo che X forma con l'orizzonte a causa della rotazione di Y si chiama **angolo di beccheggio (pitch)** e viene generalmente indicato con la lettera greca **θ** (theta).

Una rotazione intorno all'asse Z : non porta alla formazione di nessun nuovo angolo con l'orizzonte: la rotazione intorno a Z porta piuttosto gli assi X e Y a ruotare di un certo angolo sul piano in cui già si trovano. Questo angolo si chiama **angolo di imbardata (yaw)** e viene generalmente indicato con la lettera greca **ψ** (psi).

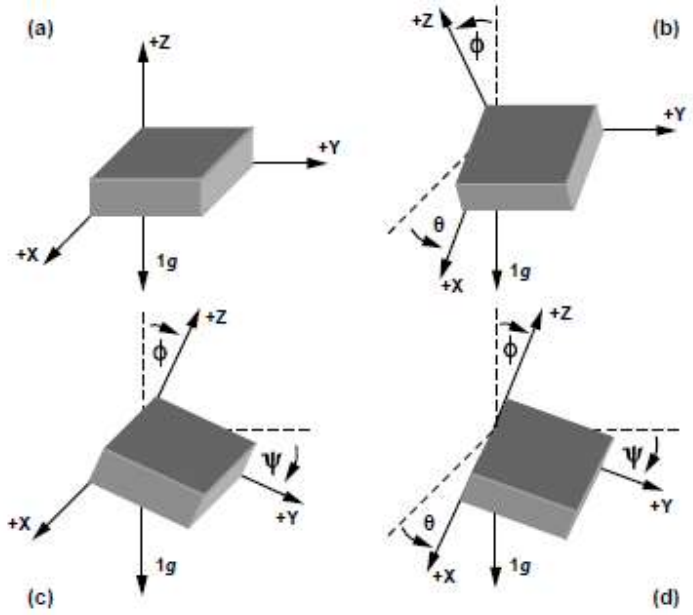
NOTA: Un accelerometro a 3 assi è in grado di rilevare l'angolo di rollio e l'angolo di beccheggio ma non l'angolo di imbardata per quanto appena detto dato che la rotazione intorno a Z non provoca una variazione degli angoli che gli assi formano rispetto all'orizzonte e di conseguenza i valori di accelerazione gravitazionale cui i singoli assi sono sottoposti.

PER FAR FRONTE A QUESTO PROBLEMA SI INTRODUCE IL GIROSCOPIO.

GIROSCOPIO:

Un giroscopio elettronico, in realtà, rileva la velocità angolare, poi per successiva integrazione si risale all'angolo.
I giroscopi sono indispensabili sugli elicotteri per evitare la rotazione indesiderata intorno all'asse di rotazione dell'elica principale. Un altro sistema, meno preciso, per rilevare l'angolo di imbardata potrebbe essere quello di utilizzare un magnetometro (una bussola elettronica).

COME MISURARE I VALORI DI UN ACCELLEROMETRO A 3 ASSI?



Il nostro oggetto non si muoverà più all'interno di una circonferenza come nel caso dell'accelerometro a 2 assi in cui Il valore di accelerazione sull'asse X è proporzionale al seno dell'angolo di inclinazione, mentre il valore di accelerazione sull'asse Y è proporzionale al coseno dell'angolo di inclinazione: le due sensibilità si compensano e il loro rapporto è una costante, per cui possiamo scrivere:

$$\theta = \arctan\left(\frac{Ax}{Ay}\right)$$ *angolo di beccheggio (Y)*

Nel nostro caso a **3 assi** si muoverà all'interno di una sfera: la forza di gravità agisce ora su 3 assi di misura, per cui nei calcoli bisogna tenere conto di tutti e 3 gli assi. Teniamo conto di una posizione iniziale del corpo adagiata sul piano per cui gli assi X e Y del nostro corpo si trovano coricati sull'orizzonte (e quindi formano un angolo di 0° con esso, come nella figura dell'aereo) e l'asse Z è perpendicolare al piano (forma un angolo di 90° con l'orizzonte, o anche di 0° con il vettore di accelerazione gravitazionale).

ANGOLO DI ROLLIO (ROLL)

$$\theta = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

ANGOLO DI BECCHEGGIO (PICHT)

$$\Phi = \arctan\left(\frac{A_z}{\sqrt{A_x^2 + A_y^2}}\right)$$

ANGOLO DI IMBARDATA (YAW) : misurato quest'ultimo con il giroscopio.

$$\varphi = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

Dopo aver acquisito il valore della velocità angolare di un asse, basterà dividere il valore per 1000 (1000ms = 1 secondo) e sommare il risultato in una variabile ogni volta che si verifica un piccolo spostamento del giroscopio. La formula finale sarà così:

```
if (GyroAsseZ > 1 || GyroAsseZ < -1) {
    GyroAsseZ   /= 1000;
    yaw += GyroAsseZ;
  }
```

Noi utilizziamo **sensori DIGITALI** che comunicano tramite seriale IIC e In questi casi il valore di accelerazione di ogni asse è contenuto in uno o più registri e si interroga quindi l'accelerometro per avere il valore misurato sugli assi.

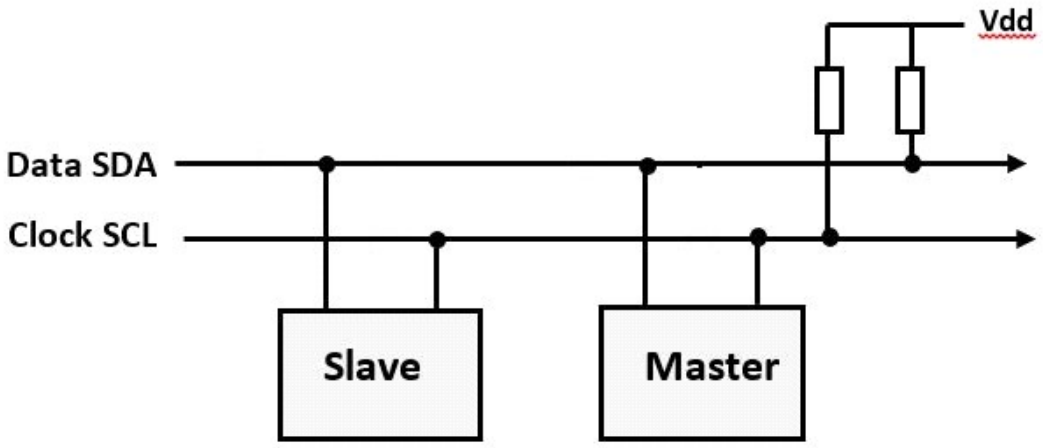
Mentre Gli accelerometri **analogici** vanno interfacciati con MCU dotate di convertitore A/D e quelli buoni hanno una larghezza di banda selezionabile, in cui la questione del rumore può essere risolta con il filtro di Kalman.

Segue in figura una tabella che riporta il ***collegamento*** della scheda con la IMU6050 attraverso l'Extension Header JN1 permettendo così la comunicazione.

Pin IMU	Numero Pin IMU	Pin Renesas	Numero Pin Renesas
VCC	1	3V3	3
GND	2	Ground	4
SDA	3	SDA	25
SCL	4	SCL	26

CENNI TEORICI PROTOCOLLO IIC

Il protocollo I2C è stato creato negli anni 80'; la sigla, comunemente indicata anche con I2C, sta per Inter-Integrated-Circuit. Il protocollo permette la comunicazione di dati tra due o più dispositivi utilizzando un bus a due fili (Figura 1).
In tale protocollo le informazioni sono inviate serialmente usando una linea per i dati (**SDA:** Serial Data line) ed una per il Clock (**SCL:** Serial Clock line).



(Figura1)

Il dispositivo **Master** è semplicemente il dispositivo che controlla il bus in un certo istante; tale dispositivo controlla il segnale di Clock e genera i segnali di START e di STOP, per Master ci riferiamo al microcontroller Renesas RX63N.

I dispositivi **Slave** semplicemente "ascoltano" il bus ricevendo dati dal Master o inviandone qualora questo ne faccia loro richiesta, chiameremo quindi Slave il sensore IMU6050 e il sensore INA219.

A differenza di altri bus, i due conduttori del bus IIC devono essere controllati come linee di uscita di tipo open-collector/open-drain e devono essere mantenute a livello alto usando un resistore di pull-up per ciascuna.

I valori della resistenza di pull-up dipendono da vari fattori, quali

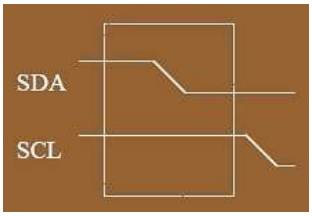
la tensione di **alimentazione 5V**, la capacità del bus ed il numero di dispositivi connessi.

In prima approssimazione possiamo dire che valori intorno ai ***2 kΩ*** vanno bene per la maggior parte delle applicazioni; valori inferiori potrebbero dare origine a correnti troppo elevate per i dispositivi.

La ***regola fondamentale*** da tenere sempre ben presente è che nella trasmissione dei dati, il segnale SDA può cambiare stato soltanto mentre il segnale SCL è basso, ed il suo valore viene poi letto durante il fronte di salita o di discesa del segnale SCL.

Una sequenza elementare di lettura o scrittura di dati tra Master e Slave segue il seguente ordine:

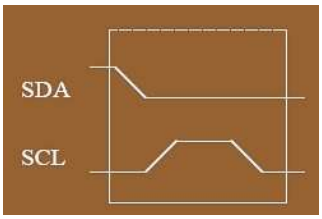
1. Invio del bit di START (S) da parte del Master: La condizione è realizzata mandando a livello basso la linea SDA mentre la linea SCL è a livello alto, poi abbassando anche la linea SCL.



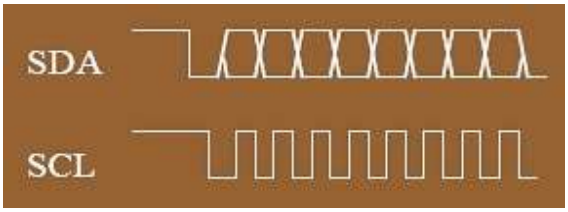
2. Invio dell'indirizzo dello slave (ADDR) ad opera del Master: Il primo byte che viene immesso sul bus dopo uno Start è quello di indirizzo per la periferica da attivare. IIC consente indirizzi a 7 bit più significativi, mentre il bit 0 (LSB) conterrà l'indicazione dell'operazione, se lettura o scrittura.

3. Invio del bit di Read (R) o di Write (W): che valgono rispettivamente 1 e 0 (sempre ad opera del master)

4. Attesa/invio del bit di Acknowledge (ACK) (bit di riconoscimento): Il protocollo IIC richiede che ogni byte trasmesso deve essere concluso con la condizione ACK, 8 impulsi del clock sono utilizzati per il sincronismo con i dati trasmessi dal dispositivo master, mentre il nono serve a sincronizzare la condizione di Acknowledge dell'unità ricevente.

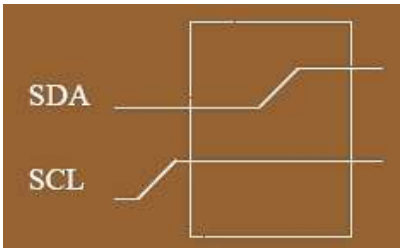


5. Invio/ricezione del byte dei dati (DATA): I dati trasmessi sono a 8 bit, dato che lo shift register ha questa dimensione. Se il dato ha dimensioni maggiori, occorrerà un adeguato numero di trasferimenti a 8 bit. Per ogni bit immesso sulla linea SDA viene generato un impulso di clock sulla linea SCL. I dati sono validi con SCL a livello alto.



6. Attesa/invio del bit di Acknowledge (ACK)

7. Invio del bit di STOP da parte del Master:
E' generato dal Master manipolando lo stato delle due linee, SCL viene rilasciata e successivamente viene rilasciata anche SDA. Alla fine della condizione di Stop, entrambe le linee sono a livello alto grazie alle resistenze di pull-up.



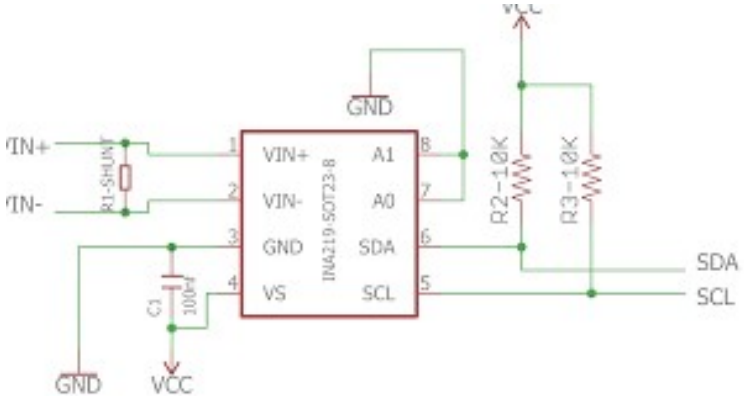
CENNI TEORICI SENSORE DI CORRENTE INA219:

INA219 è un amplificatore di rilevamento corrente digitale con un'interfaccia compatibile IIC e SMBus. Fornisce letture digitali di corrente, tensione e potenza necessarie per un processo decisionale accurato in sistemi controllati con precisione. I registri programmabili consentono una configurazione flessibile per la risoluzione delle misurazioni e un funzionamento continuo.

CARATTERISTICHE PRINCIPALI:

- Resistore di corrente 0,1 ohm 1% 2W.
- Tensione di alimentazione del carico sino a +26 V.
- Misura di corrente sino a ± 3.2 A, con una risoluzione di ± 0.8mA.
- Misure del PCB 23x20 mm.
- Questa scheda utilizza indirizzi IIC 7 bit:
  0x40 , 0x41 , 0x44 , 0x45 ..
- INA219 funziona da –40°C a 125°C.
- Pacchetti SOT23-8 e SOIC-8 INA219 è disponibile in due gradi: A e B.
- Il dispositivo utilizza un singolo alimentatore da 3V a 5V.

SCHEMA SENSORE INA219



I ***due ingressi analogici*** su INA219, ***IN+ e IN–***, si collegano a una resistenza shunt nel bus di interesse. INA219 è in genere alimentato da un'alimentazione separata da 3 a 5,5 V. Il bus rilevato può variare da 0 a 26 V.

INA219 rileva la piccola caduta attraverso la ***Rshunt*** per la tensione di shunt e rileva la tensione rispetto alla terra da IN– per la tensione del bus.

Quando il sensore è in modalità operativa normale (ovvero, i bit MODE del registro di configurazione sono impostati su 111), converte continuamente la tensione di shunt fino al numero impostato nella funzione di media della tensione di shunt (registro di configurazione, ***bit SADC***).
Il dispositivo converte quindi la tensione del bus fino al numero impostato nella media della tensione del bus (registro di configurazione, ***bit BADC***).

Tutti i calcoli di corrente e potenza vengono eseguiti in background e non contribuiscono al tempo di conversione.

La ***modalità ADC Off*** (impostata dal registro di configurazione, bit MODE) interrompe tutte le conversioni.

Sebbene INA219 possa essere letto in qualsiasi momento e i dati dell'ultima conversione rimangano disponibili, viene fornito il bit pronto per la conversione (registro di stato, ***bit CNVR***) per facilitare il coordinamento delle conversioni one-shot o attivate.

Per la ***misurazione della potenza***, la corrente e la tensione del bus vengono convertite in diversi punti nel tempo, a seconda della risoluzione e delle impostazioni della modalità media.
Ad esempio, se configurato per la media di campionamento a 12 bit e 128, è possibile fino a 68ms nel tempo tra il campionamento di questi due valori. Ancora una volta, questi calcoli vengono eseguiti in background e non si aggiungono al tempo di conversione complessivo.

Dopo aver programmato il registro di calibrazione, il registro corrente (04h) e il registro di potenza (03h) si aggiornano di conseguenza in base alle misurazioni della tensione di shunt e del bus corrispondenti. Fino a quando non viene programmato il registro di calibrazione, il registro corrente e il registro di potenza rimangono a zero.

Il registro di calibrazione viene calcolato in base all'equazione 1.

$$Cal = trunc(\frac{0{,}04096}{Current_{LSB} * R\_shunt})$$

Questa equazione include : 0,04096 è un valore fisso interno utilizzato per garantire il corretto ridimensionamento e il termine Current_LSB, che è il valore programmato per LSB per il registro corrente (04h).

$$Current_{LSB} = \frac{Max_{Current}}{2^{15}}$$

L'utente utilizza questo valore per convertire il valore nel registro corrente (04h) nella corrente effettiva in ampere.

COMUNICAZIONE IIC:

Due linee bidirezionali, SCL e SDA, collegano INA219 al bus. Sia SCL che SDA sono connessioni open-drain.

Come già visto nella sezione del protocollo IIC il dispositivo che avvia il trasferimento è chiamato Master (controllore)e i dispositivi controllati dal master sono Slave(INA219),il procedimento di trasferimento dati è lo stesso descritto nella sezione precedente.

INA219 ha due pin di indirizzo, A0 e A1.
La tabella 1 descrive i livelli logici dei pin per ciascuno dei 16 indirizzi possibili.
Lo stato dei pin A0 e A1 viene campionato su ogni comunicazione bus e deve essere impostato prima che si verifichi qualsiasi attività sull'interfaccia.
I pin dell'indirizzo vengono letti all'inizio di ogni evento di comunicazione.

A1	A0	SLAVE ADDRESS
GND	GND	1000000
GND	V_{S+}	1000001
GND	SDA	1000010
GND	SCL	1000011
V_{S+}	GND	1000100
V_{S+}	V_{S+}	1000101
V_{S+}	SDA	1000110
V_{S+}	SCL	1000111
SDA	GND	1001000
SDA	V_{S+}	1001001
SDA	SDA	1001010
SDA	SCL	1001011
SCL	GND	1001100
SCL	V_{S+}	1001101
SCL	SDA	1001110
SCL	SCL	1001111

(Tabella 1)

L'accesso a un registro particolare su INA219 si ottiene scrivendo il valore appropriato sul puntatore del registro. La scrittura in un registro inizia con il primo byte trasmesso dal Master. Questo byte è l'indirizzo Slave. INA219 quindi conferma la ricezione di un indirizzo valido. Il byte successivo trasmesso dal master è l'indirizzo del registro in cui verranno scritti i dati. Questo valore dell'indirizzo di registro aggiorna il puntatore al registro desiderato.

I successivi due byte vengono scritti nel registro indirizzato dal puntatore. INA219 conferma la ricezione di ciascun byte di dati. Il master può terminare il trasferimento di dati generando una condizione START o STOP.

Durante la lettura da INA219, l'ultimo valore memorizzato nel puntatore del registro mediante un'operazione di scrittura determina quale registro viene letto durante un'operazione di lettura. Per modificare il puntatore del registro per un'operazione di lettura, è necessario scrivere un nuovo valore nel puntatore del registro.

Se si desiderano letture ripetute dallo stesso registro, non è necessario inviare continuamente i byte del puntatore del registro; INA219 mantiene il valore del puntatore del registro fino a quando non viene modificato dalla successiva operazione di scrittura.

CODICE

In seguito riportiamo delle considerazioni sulle librerie implementate, in particolare si è deciso di strutturare il codice come se fosse un modulo indipendente importabile così su qualsiasi progetto. Abbiamo cercato di semplificare il più possibile i precedenti codici incapsulando tutte le sue funzionalità.

All'interno della cartella relativa alla stesura dei codici sono visibili le seguenti librerie:

- *Setup*
- *CMT*
- *I2C*
- *Mag*
- *Imu*
- *AHRS*
- *Ina219*
- *Main*

Ogni libreria creata possiede il suo ***file.h*** di intestazione in cui abbiamo definito le strutture utilizzate, il tipo di variabili e gli input e output delle funzioni con i relativi commenti per ognuna di essa. Procederemo in seguito con la spiegazione di ogni singola libreria creata, e delle funzioni al suo interno.

AHRS

Un ***AHRS*** (Attitudine and Headind Reference System) fornisce l'orientamento 3D integrando giroscopi e fondendo questi dati con i dati dell'accelerometro e del magnetometro. Con la fusione del sensore, la deriva dell'integrazione dei giroscopi è compensata da vettori di riferimento, vale a dire la gravità e il campo magnetico terrestre. Al fine di migliorare le prestazioni di questi sistemi AHRS che utilizzano i dati di velocità angolare e accelerazione da un'unità di misura inerziale (IMU), può essere utilizzato per calcolare una posizione relativa nel tempo.

La libreria AHRS ha il compito di rielaborare i dati del sensore IMU, è costituita da funzioni come ***getAHRS*** la quale ricava gli angoli di Tait-Bryan con il vantaggio che può essere chiamata con una frequenza più bassa di quella con cui viene aggiornato il filtro implementato nella funzione ***madgwickFilterUpdate.***

Mentre la funzione ***getYPR*** trasforma le misurazioni fatte in quaternioni (quaterna di numeri complessi) e ne restituisce i dati in gradi e in radianti nella struttura dati AHRS_data.

SETUP

```
//Routine di setup dell'imu
void Setup_MARG(AHRS_out* ahrs)
{
    char msg[12];

    lcd_initialize();
    lcd_clear();

    lcd_display(LCD_LINE1," IMU SETUP ");
    lcd_display(LCD_LINE2," IN CORSO ");

    CMT_init();
    imu_init(&ahrs->sens);
    mag_init(&ahrs->mag);

    lcd_display(LCD_LINE4,"Calibrazione");
    lcd_display(LCD_LINE5,"Magnetometro");

    calibrationYPR(msg, &ahrs->mag);
}
```

Per ridurre al minimo le chiamate del sistema di controllo AHRS si è implementata la libreria ***Setup_MARG*** la quale gestisce le inizializzazioni dei sensori, dello schermo e la calibrazione del magnetometro.

```
// Funzione di lettura dei dati dell' MARG filtrati
void Read_MARG(AHRS_out* ahrs)
{
    imu_read(&ahrs->raw, &ahrs->sens, &ahrs->temp);
    mag_read(&ahrs->mag);
    getYPR(&ahrs->mag, &ahrs->temp, &ahrs->ahrs_data);
}
```

Basterà una chiamata della funzione ***Read_MARG*** per leggere i valori dall'imu, dal magnetometro se necessario, attrarverso il puntatore alla struttura dati AHRS_out.

CMT

CMT è una libreria inclusa nel main, fornita dagli esempi Renesas, che permette di usare in modo semplificato le temporizzazioni. E'stata utilizzata per aggiornare i dati di lettura ad intervalli regolari, ciò avviene tramite funzioni che, in base al tempo di clock della periferica (48MHz), fanno entrare il programma in un ciclo vuoto che tiene in stallo il microprocessore per l'intervallo di tempo desiderato.

I2C

La libreria **I2C** è la responsabile della comunicazione tra Master e Slave, all'interno oltre alla funzione di inizializzazione sono visibili le due funzioni di letture e scrittura.
Rispettivamente la funzione ***i2c_read*** e ***i2c_write*** nelle quali verranno passati rispettivamente : l'indirizzo dello slave, il numero del registro, il numero di byte da leggero sullo slave, e il puntatore al buffer su cui verranno memorizzati i dati.

```
int i2c_read (uint8_t slave_addr, uint8_t register_number, uint8_t num_bytes,
                    uint8_t *dest_buff)
{
     /* Storage for the slave address and target register. */
    uint8_t     addr_and_register[2];
    riic_ret_t  ret = RIIC_OK;

    addr_and_register[0] = slave_addr<<1;
    addr_and_register[1] = register_number;

    ret |= R_RIIC_MasterTransmitHead(CHANNEL_0, addr_and_register, 2);

    /* Now read the data from the target register into the destination buffer. */
    ret |= R_RIIC_MasterReceive(CHANNEL_0, slave_addr<<1, dest_buff, num_bytes);

    return ret;
```

```
int i2c_write(uint8_t slave_addr, uint8_t register_number, uint8_t num_bytes,
                     uint8_t *source_buff)
{
    /* Storage for the slave address and target register. */
    uint8_t     addr_and_register[2];
    riic_ret_t  ret = RIIC_OK;

    addr_and_register[0] = slave_addr<<1;
    addr_and_register[1] = register_number;

    ret |= R_RIIC_MasterTransmitHead(CHANNEL_0, addr_and_register, 2);

    /* Now write the data from the source buffer into the target register. */
    ret |= R_RIIC_MasterTransmit(CHANNEL_0, source_buff, num_bytes);

    return ret;
```

INA219

Dopo avere creato una struttura nel file *ina219.h* per salvare i valori delle 3 correnti, una per ogni ruota, sfruttando così 3 sensori di corrente, nel file sorgente sono implementate la funzione di inizializzazione del sensore ***ina_init*** e la funzione di lettura per dei 3 sensori ***ina_read*** in seguito riportate.

File sorgente (ina219.c)

```c
#include <ina219.h>

#define INA219_IIC_ADDRESS_1    0x40
#define INA219_IIC_ADDRESS_2    0x41
#define INA219_IIC_ADDRESS_3    0x42
#define CURRENT_REGISTER    0x04

int ina_init(){
    if(i2c_init())
        return 0x1;
    //inizializza x3 ina219
    return 0;
}
int ina_read(INA_sens* ina){
    uint8_t tmp;
    //ina -> INA_current_1 = i2c_read(uint8_t slave_addr, uint8_t register_number, uint8_t num_bytes,uint8_t *dest_buff);
    ina -> INA_current_1 = i2c_read(INA219_IIC_ADDRESS_1, CURRENT_REGISTER, 1, &tmp);
    ina -> INA_current_2 = i2c_read(INA219_IIC_ADDRESS_2, CURRENT_REGISTER, 1, &tmp);
    ina -> INA_current_3 = i2c_read(INA219_IIC_ADDRESS_3, CURRENT_REGISTER, 1, &tmp);
    return 0;
}
```

File sorgente (ina219.h)

```c
#ifndef INA219_H_
#define INA219_H_

#include "i2c.h"

typedef struct{
    float INA_current_1;
    float INA_current_2;
    float INA_current_3;
} INA_sens;

int ina_init();

int ina_read(INA_sens*);

#endif /* INA219_H_ */
```

IMU

Come per il sensore di corrente la libreria IMU è compresa di un file di intestazione ***imu.h*** in cui sono definite 3 strutture :

IMU_raw che contiene dati grezzi del imu.

IMU_sens che contiene dati sensore imu.

IMU_temp che contiene dati rielaborati.

Nel file sorgente sono presenti le due classiche funzioni principali che sono ***imu_init*** responsabile dell'inizializzazione del sensore inerziale.

```
int imu_init(IMU_sens* imu_sens){
    // avvia i2c
    if (i2c_init())
        return 0x1;
    // avvia sensore con la libreria Inversense
    if (mpu_init(0))
        return 0x2;
    // abilita sensori Giroscopio e Accelerometro
    if (mpu_set_sensors(INV_XYZ_ACCEL | INV_XYZ_GYRO))
        return 0x3;
    // imposta frequenza la conversione dei dati inerziali
    if (mpu_set_sample_rate(200/*Hz*/)) //Hz
        return 0x4;

    // leggi sensibilità  del sensore
    mpu_get_accel_sens(&imu_sens->acc_sens);
    mpu_get_gyro_sens(&imu_sens->gyr_sens);
    return 0x0;
}
```

imu_read legge i dati dal giroscopio e dall'accelerometro, avendo come argomenti della funzione i puntatori delle 3 strutture.

```
int imu_read(IMU_raw* imu_raw, IMU_sens* imu_sens, IMU_temp* imu_temp){

    short data_acc[3] = {imu_raw->accRoll,imu_raw->accPitch,imu_raw->accYaw};
    mpu_get_accel_reg(data_acc, NULL);
    imu_raw->accRoll = data_acc[0];
    imu_raw->accPitch = data_acc[1];
    imu_raw->accYaw = data_acc[2];

    short data_gyr[3] = {imu_raw->gyrRoll, imu_raw->gyrPitch, imu_raw->gyrYaw};
    mpu_get_gyro_reg(data_gyr, NULL);
    imu_raw->gyrRoll = data_gyr[0];
    imu_raw->gyrPitch = data_gyr[1];
    imu_raw->gyrYaw = data_gyr[2];

    imu_temp->accRoll=imu_raw->accRoll / imu_sens->acc_sens;
    imu_temp->gyrRoll = imu_raw->gyrRoll / imu_sens->gyr_sens * 0.01745329252;
    imu_temp->accPitch = imu_raw->accPitch / imu_sens->acc_sens;
    imu_temp->gyrPitch = imu_raw->gyrPitch / imu_sens->gyr_sens * 0.01745329252;
    imu_temp->accYaw = imu_raw->accYaw / imu_sens->acc_sens;
    imu_temp->gyrYaw = imu_raw->gyrYaw / imu_sens->gyr_sens * 0.01745329252;
    return 0;
}
```

MAIN

Nel **main** ovvero il corpo principale del codice è stato ridotto al minimo indispensabile. E'stato necessario fare una chiamata *Exstern* della struct *timerClocks* in modo da non modificare la libreria CMT. Per cui se si vuole utilizzare la libreria CMT bisognerà importarla e dichiarare l'istanza di *TimerClocks* come *Exstern*.
E'bene fare attenzione alla frequenza con cui fare le chiamate di stampa su schermo, i limiti sono dettati dall'hardware utilizzato nel progetto, è sconsigliato di fare chiamate più di ogni decimo di secondo.

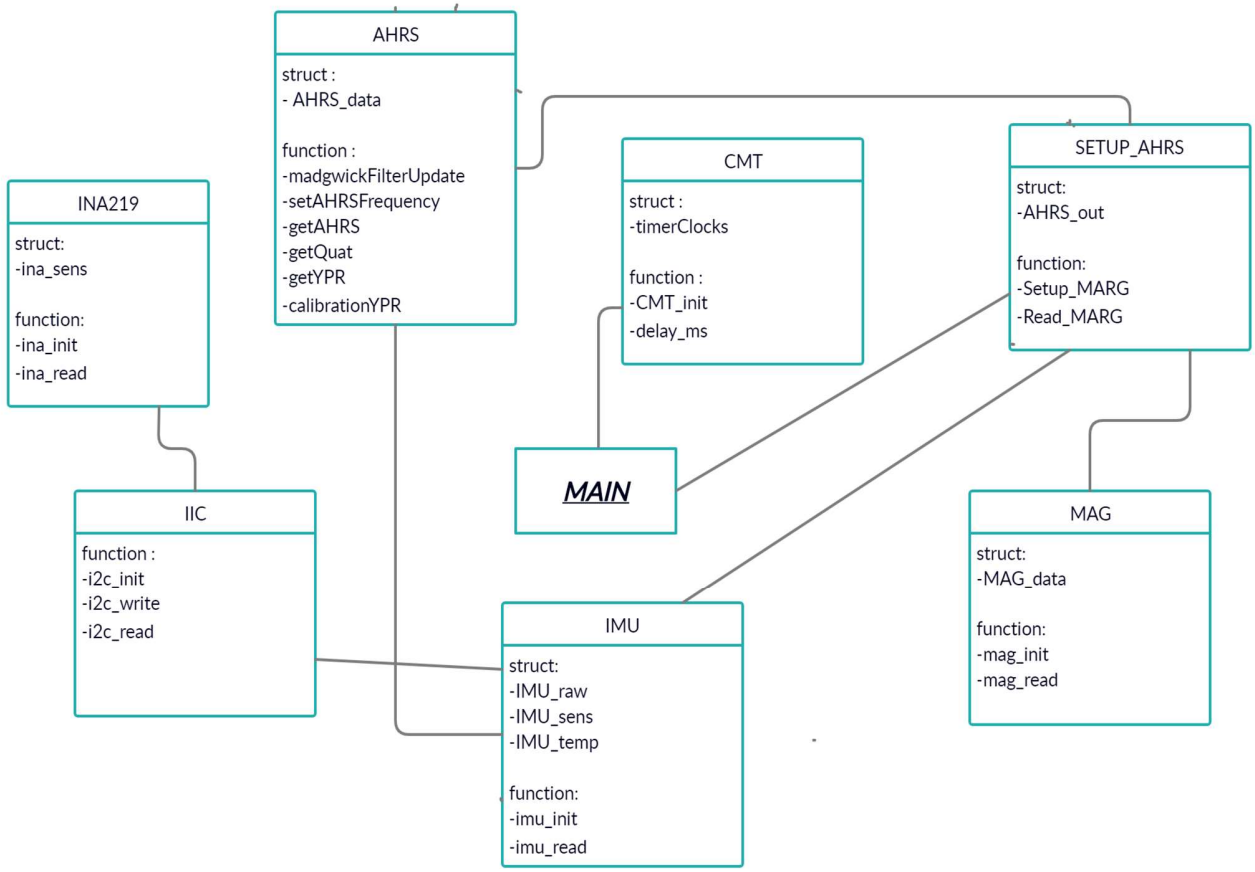
```
main(void)
{
    /*Dichiarazioni strutture dati utilizzate.*/
    AHRS_out ahrs;
    * extern struct timerClocks timers
    extern struct timerClocks timers;

    /* funzione di setup dello schermo, CMT, accelerometro, giroscopio, magnetometro e calibrazione magnetometro.*/
    Setup_MARG(&ahrs);

    while (1)
    {
        * La frequenza di lettura va impostata in base alla frequenza del mpu e magnetometro.
        if(timers.timer_5mS)
        {
            Read_MARG(&ahrs);
            RealTimeChart(&ahrs);

            timers.timer_5mS=0;

             * Frequenza di stampa su schermo dei dati.
            if(timers.timer_100mS)
            {
                Print_ABS(&ahrs);
                Print_Angoli(&ahrs);
                Print_VelAng(&ahrs);
                //Print_Temp(&ahrs);

                timers.timer_100mS = 0;
            }
        }
    }
```

PRINT_ON_SCREEN

E' una libreria non necessaria al fine del funzionamento del sistema di controllo AHRS, e non é richiesta da nessuna altra libreria; é un modulo a se stante, che integra la stampa.

Diagramma modello codice

E’ stato realizzato un semplice diagramma a linguaggio unificato che racchiude le relative funzioni utilizzate, al fine di semplificare la struttura generale del codice in modo da renderlo più comprensibile.



PROBLEMI TEST DI LABORATORIO

Durante la stesura del codice, la compilazione e la verifica delle funzionalità sui dispositivi sono stati spesso rinvenuti i seguenti problemi:

- ***Restituzione valori NaN***:
  Durante le prime compilazioni, alcuni valori degli angoli venivano stampati a schermo come NaN (Not a Number) e su LCD con degli asterischi. Il problema stava nella conversione tra alcuni tipi di dato (8/16/32 bit) che rendevano così la sequenza di bit non rilevabile come un numero. Il problema è stato risolto sistemando i vari casting che vengono effettuati all'interno della libreria *Imu* al fine di convertire i valori letti dall'IMU in float per avere una migliore accuratezza dei risultati ottenuti.

- ***Calibrazione dei valori***:
  Dopo aver ottenuto la stampa su LCD, alcuni di essi risultavano non corretti o imprecisi. Si è quindi passati ad una fase di calibrazione e modifica dei valori di scala per ottenere il miglior risultato possibile. Tale calibrazione è stata effettuata eseguendo una media aritmetica su 100 misure a sistema fermo ed in equilibrio. Eseguendo alcune ottimizzazioni in questi calcoli abbiamo raggiunto una precisione intorno al centesimo di grado.

- **Impossibilità di testare il codice ina219:**
  Purtroppo per cause di forza maggiore non è stato possibile accedere in laboratorio per testare la funzionalità della libreria ina219, benchè il codice compili perfettamente e senza errori non possiamo escludere la possibilità di eventuali errori di lettura da parte del sensore di corrente.

RIFERIMENTI BIBLIOGRAFICI

1. Hardware’s Manual Renesas RX63N
2. Datasheet MPU-6050
3. Datasheet INA219
4. https://x-io.co.uk/open-source-imu-and-ahrs-algorithms/
5. Esempi e2studio Renesas
6. IIC,https://i2c.info
3. “Informatica, Linguaggio C”