*Università degli Studi di Firenze*

Corso di Laurea in Ingegneria Elettrica e dell'automazione


Anno Accademico 2012/2013

Elaborato di:

Laboratorio di automatica e azionamenti C.I

Stima e identificazione A

# *Controllo di una barchetta autonoma*

PROFESSORI:

***Michele Basso***

***Luigi Chisci***

CANDIDATO:

***Marco Montagni***

***Ermete Cauduro Bianchi***

*Anno Accademico 2013/2014*

# Indice delle figure

# Introduzione

Il lavoro proposto si pone l'obiettivo di implementare il controllo di una barchetta autonoma per la pesca sportiva.

Verrà analizzata la soluzione proposta descrivendo il sistema, nella cura della realizzazione della parte meccanica e nella scelta dei riferimenti, nella scelta dei componenti elettrici e di quelli elettronici descrivendo le scelte fatte in ogni parte.

Sarà descritto lo studio del Firmware, il sistema operativo della barchetta, con le relative problematiche fisiche riscontrate e con le relative stime degli stati per ottenere il controllo desiderato. Infine saranno analizzati i log, ottenuti tramite l'interfaccia, che hanno fornito gli errori per una più attenta e futura taratura del controllo.

Per ulteriori chiarimenti riguardo al codice si consulti l'allegato e per quanto riguarda le librerie modificate e incluse nel codice, esse sono state ricondivise sul forum http://forum.arduino.cc .

# 1 - Descrizione del sistema

Il sistema in esame presenta varie peculiarità che verranno descritte in questo capitolo. Di fatto una combinazione ponderata di parte meccanica, elettrica ed elettronica ha permesso di ottenere stabilità e facilità di realizzazione/utilizzo.

## 1.1 - Descrizione della parte meccanica

La meccanica è abbastanza basilare. Un motore elettrico brushless outrunner da 700[rpm/V] è accoppiato mediante uno spinotto da 5mm a 4mm di diametro ad un albero che trasmette la rotazione ad una elica sommersa. Lo spinotto, fatto al tornio, presenta 2 grani per fermare i due alberini sulla parte piatta di essi. Il motore è fissato allo scafo tramite un alloggiamento con inclinazione regolabile posto sulla flangia del motore. Internamente una serpentina, comunicante con l'esterno tramite due micro innesti da irrigazione, raffredda il motore che è stato fissato ponendo della pasta termo conduttiva per abbassarne la resistenza termica. Il timone cambia la sua inclinazione tramite un servo analogico vincolato tramite un filo di acciaio armonico passante da un soffietto di gomma per evitare che entri acqua. Il timone, in ergal, scende di 90mm di profondità per permettere una migliore controllabilità e presenta un foro verso la direzione della barca, che permette il pescaggio dell'acqua per il raffreddamento del motore, comunicante con l'interno tramite un tubo di silicone fissato con una guarnizione modellabile bicomponente ed un innesto da aria compressa. La calotta presenta una cavità per l'alloggiamento della parte elettronica ed è fissata allo scafo con delle linguette in plastica derivanti da tubazioni riscaldate e tagliate. Si è scelto di evitare guarnizioni in quanto, avendo visto altri modelli di barche rc commerciali, è sempre necessario utilizzare del nastro adesivo per sigillare la calotta ed evitare infiltrazione di acqua.

### 1.1.1 - Descrizione della barchetta

La forma dell'imbarcazione deriva da una combinazione di estetica e valutazione di modelli commerciali. Infatti, per permettere la navigazione in mare è stata scelta una

forma a catamarano leggermente più larga dei modelli rc da velocità pura presenti in commercio. Il tunnel è stato accorciato a poppa per evitare che prenda il volo durante una navigazione ad alta velocità. È stata eliminata una porzione a poppa delle due derive/scafi che compongono il natante ispirandosi ai Drug Race e cercando di migliorare l'affronto delle piccole onde sulla riva del mare in quanto mantiene un certo beccheggio in navigazione.

### 1.1.2 - Descrizione dei sistemi di riferimento

Per quanto riguarda i sistemi di riferimento, essi sono stati scelti nel modo che sembrava più congeniale alla risoluzione del problema di navigazione.

La piattaforma inerziale è quindi stata progettata rispettando i versi degli integrati MEMS degli accelerometri e dei giroscopi che sono gli stessi su x, y e z (con questo si intende che gli integrati disposti sul PCB hanno il verso sui rispettivi assi in modo congruo tra loro) come è possibile vedere dalla Figura 1.

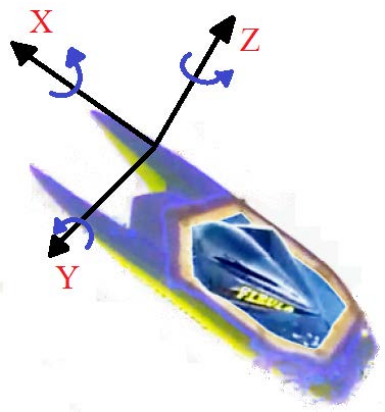


**Figura 1 Sistema di riferimento Barchetta**

Come è possibile vedere dalla Figura 1 il sistema di riferimento Barchetta è posizionato dove si trova realmente la IMU, ovvero cercando la distanza massima dal motore che disturberebbe il magnetometro triassiale.

Per il sistema di riferimento fisso "mondo" sono state adottate le classiche coordinate di Latitudine e Longitudine in gradi decimali.

L'origine degli assi è quindi stata presa su 0,0 ovvero meridiano di Greenwich ed equatore. Come è possibile vedere dalla Figura 2 L'angolo *θd*(*θ desiderato)* rappresenta l'angolo desiderato dal polo nord geografico per arrivare al punto A calcolato dalle coordinate di Latitudine e Longitudine del GPS, mentre l'angolo *θb*(*θ barchetta*) rappresenta l'angolo tra il nord magnetico e la proiezione sull'acqua dell'asse X della barchetta calcolato con la piattaforma inerziale, le matrici di rotazione ed il filtro di Kalman.

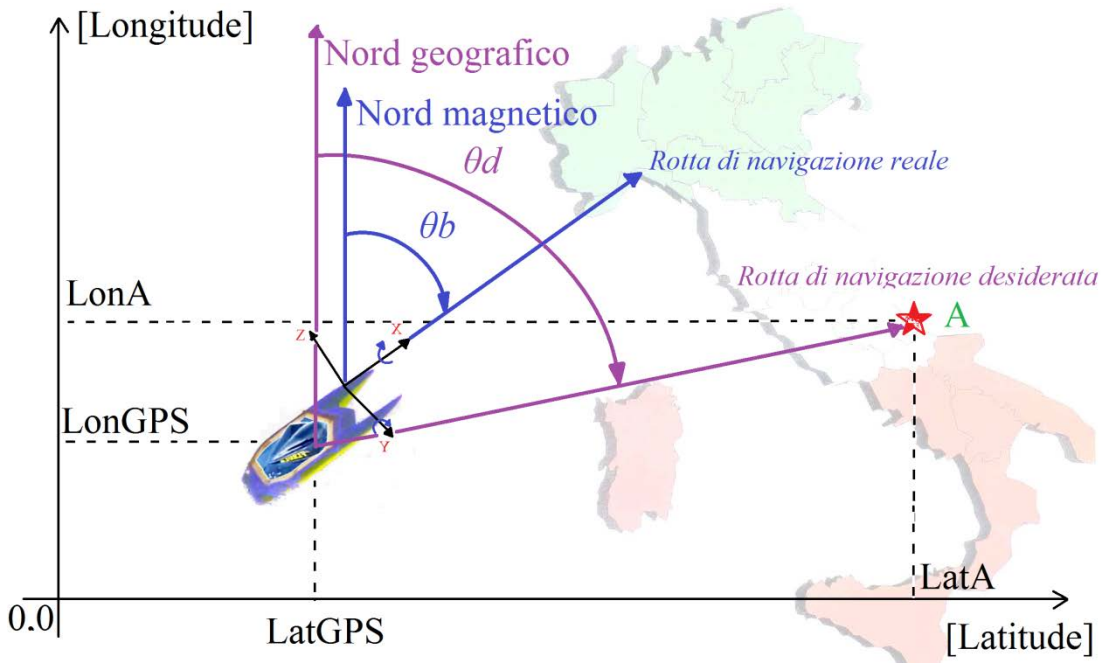


**Figura 2 Sistema di riferimento fisso**

Il range degli angoli è stato scelto da +180° a -180° ed è stato pensato un modo per effettuare le virate sempre dalla parte più conveniente in termini di angolo desiderato e angolo di orientazione della barca.

### 1.1.3 - La realizzazione

La barca è stata costruita modellando un blocco di polistirolo facilmente reperibile come isolante per tetti. Il costo ne ha fatta un'ottima base di partenza per definire le forme, anche se non sono state realizzate perfettamente simmetriche al millimetro, risultano dritte ad occhio. Successivamente è stata ricoperta di resina epossidica e stucco di vetroresina e successivamente poliestere per ricostruzione da carrozzeria. È

stata inoltre cartata con carta abrasiva del 400 in modo tale da risultare liscia al tatto e successivamente verniciata. È stato praticato il taglio della calotta con un disco dentato piccolo da Dremel e successivamente con diluente nitro è stato sciolto il polistirolo lasciato sulla parte a prora. Le immagini di Figura 3 chiariscono i dettagli di lavorazione effettuati.


**Figura 3 Realizzazione**

È stata inoltre realizzata una teca per il trasporto e mantenimento della barchetta in quanto avendo timone e parti facilmente esposte alla rottura.

La sigla F.L.B.V.L.A. è l'acronimo di "finché la barca va' lasciala andare".

### 1.1.4 - Costi

Una parte importante del progetto è stata quella della scelta delle componenti.

Infatti una attenta ricerca su Ebay e nella zona è stata la chiave per un costo complessivamente ridotto. C'è da aggiungere che il motore, le vernici ed i materiali da carrozzeria non sono stati acquistati ma donati in comodato d'uso e che anche grazie a questo è stato possibile ottenere un prezzo abbastanza basso. Rispetto a moduli commerciali del Lego mainstorm le caratteristiche tecniche del sistema proposto sono nettamente superiori, nel particolare della IMU, e a prezzi decisamente più bassi.

| Parte | Materiale | Modello | Costo[€] |
|---|---|---|---|
| Meccanica | Blocco coibentante | Polistirolo 1x0,5x0,2m | 2 |
| | Carta vetrata | 100 e 400 | 2 |
| | Resina epossidica | Raytec bic | 10 |
| | Poliestere | Poly | 5 |
| | Vernice | Da carroziera acqua | 10 |
| | Timone | 100x40mm | 8 |
| | Soffietti | da barca rc | 1 |
| | Supporto motore | Watercooled | 10 |
| | Elica/trasmissione | Plastica M4/200x4mm | 3 |
| Elettronica | Arduino | mega 2560 | 20 |
| | IMU 10Dof | GY-80 | 7 |
| | Scheda pcb | PCB presensibilizzata | 4 |
| | Cavetti M/F | Cavetti colorati 20cm | 2 |
| | Cavetti F/F | Cavetti colorati 20cm | 2 |
| | GPS | Mediatek 10Hz | 23 |
| | Xbee | Xbee Pro s4b (X2) | 40 |
| Elettrica | Batterie | Li-po 6S 3Ah 20C | 44 |
| | Connettori | XT-60 | 2 |
| | Azionamento | Mystery 100A W.C | 27 |
| | Servo | Analog servo E-Max | 3,4 |
| | Motore | Robbe 3548/06 | 50 |
| | | **TOTALE** | **275,4** |

Quindi al prezzo totale c'è da sottrarre circa 70 euro. La rispettiva soluzione con Lego sarebbe costata sicuramente di più e non avrebbe avuto le specifiche dalle soluzione proposta.

## 1.2 - Descrizione della parte elettrica

La parte elettrica è composta dal motore e dal suo azionamento, dal servo per il timone (che sono gli attuatori del sistema) e da un circuito per ottenere il controllo della barchetta in modo autonomo o tramite il segnale proveniente dalla ricevente.

### 1.2.1 - Gli attuatori

Nello specifico gli attuatori servono per muovere la barchetta e guidarla/comandarla.

#### 1.2.1.1 Il motore

Il motore brushless ovvero senza spazzole ROXXY BL Outrunner 3548/06 Figura 4, è una macchina sincrona che rispetto ai motori asincroni ha il vantaggio finale di avere una coppia pressoché costante all'aumentare dei giri. Tale caratteristica è dovuta al fatto di fornire sempre una terna di tensioni trifase in modo da ottenere un certo angolo costante con il rotore, detto questo, è logico che serviranno sensori per capire dove si trova il rotore, oppure come nel nostro caso si usa un controllo sensorless ovvero senza sensori, di poco meno preciso se realizzato finemente, ma che per applicazioni di questo tipo, ovvero con specifiche non troppo stringenti risulta un buon compromesso. Il peso di soli 175g, le dimensioni 35 x 47mm la tensione di alimentazione di circa 20 volt, le 14 coppie polari, ed i 30A [1] lo fanno un candidato ideale sia come propulsore per aerei, che per propulsioni marine.


**Figura 4 Motore**

Il compito di fornire la giusta terna di tensioni è assolto dall'ESC ovvero dall'azionamento.

### 1.2.1.2 L'azionamento

L'ESC (Figura 5), in gergo modellistico, è stato scelto molto maggiore delle specifiche richieste solo perché l'altra specifica era il prezzo.

Infatti il Mystery RC 100A ha in dotazione un dissipatore da abbinare alle barche RC con ricircolo di acqua per il raffreddamento. Supporta una tensione massima di 22,2V ovvero di 6 celle Li-Poly oppure 18 NiMH o NiCd, quella minima di 7.2V per il funzionamento del BEC usato per alimentare l'arduino a 5V e corrente massima 3A[2]. Il segnale di controllo è il classico PWM per il controllo dei Servi.

**Figura 5 ESC**

### 1.2.1.3 Il Servo

Per il controllo del timone è stato usato un Servo modello HS-430BH vedi Figura 6 analogico e dalla risposta velocissima di 16 ms/60°.

**Figura 6 Servo timone**

Tale servo viene comandato tramite il classico PWM a 50Hz con un duty che varia di 2ms ovvero da circa il 6% all'8-9% proporzionale all'angolo che attua.

La libreria servo prevede proprio di fornire una variabile da 0 a 180 corrispondente all'angolo da ottenere.

#### 1.2.1.4 La scelta del controllo

Dato che il duty-cicle varia di poco è stato riscontrato un problema per il controllo. Infatti i pin dell'Arduino, una volta settati come uscita, sono a bassa impedenza e si brucerebbero se fossero messi a tensione e per questo non è possibile fare un parallelo con la ricevente. Quindi per controllare (allo stesso modo) l'azionamento e il servo, sia con l'Arduino, che con la ricevente e il suo radiocomando in abbinamento, è necessario leggere il duty proveniente dalla ricevente e poi tramite software decidere se comandare gli attuatori con la stessa lettura oppure tramite software che provvederà a fornire autonomamente un duty, per il controllo. Per leggere il duty esiste una funzione chiamata "pulseIn" dove è possibile leggere il tempo in cui un pin è a livello logico alto: essa non è però ottimizzata e con due funzioni, una per il servo e una per l'azionamento, và in crisi. Inoltre, non è possibile neanche abbinare lo stato logico in ingresso uguale a quello in uscita in quanto, mentre con un tempo di ciclo ridottissimo funziona, già con 5ms di ciclo non trasferisce correttamente lo stato, dato che in esecuzione del programma leggendo ad esempio l'ingresso alto, lo mantiene alto per tutto il ciclo e quindi il duty-cicle in uscita risulta alterato dal ciclo. Altre soluzioni per la guida autonoma o assistita come ArduPilot utilizzano un altro microcontrollore solo per leggere il duty proveniente dalla ricevente. Per questi motivi è stato progettato un circuito home-made in Figura 7 per effettuare la commutazione da radiocontrollo ad autonomo con porte logiche. Inoltre è stato usato un circuito buffer con un Op-Amp per amplificare il segnale di controllo da mandare agli attuatori e per amplificare la corrente del partitore con resistenze dell'ordine dei kΩ, fatto per la lettura della carica delle batterie, che non sarebbe bastato per caricare in modo opportuno la capacità del sample & hold dell'Arduino per la lettura di segnali analogici. Inoltre sono stati inseriti dei condensatori per livellare la tensione in lettura dalla batterie ed evitare oscillazioni

provenienti dall'azionamento di basso costo. Tale osservazione è stata fatta per l'alimentazione di tutti i componenti elettronici.

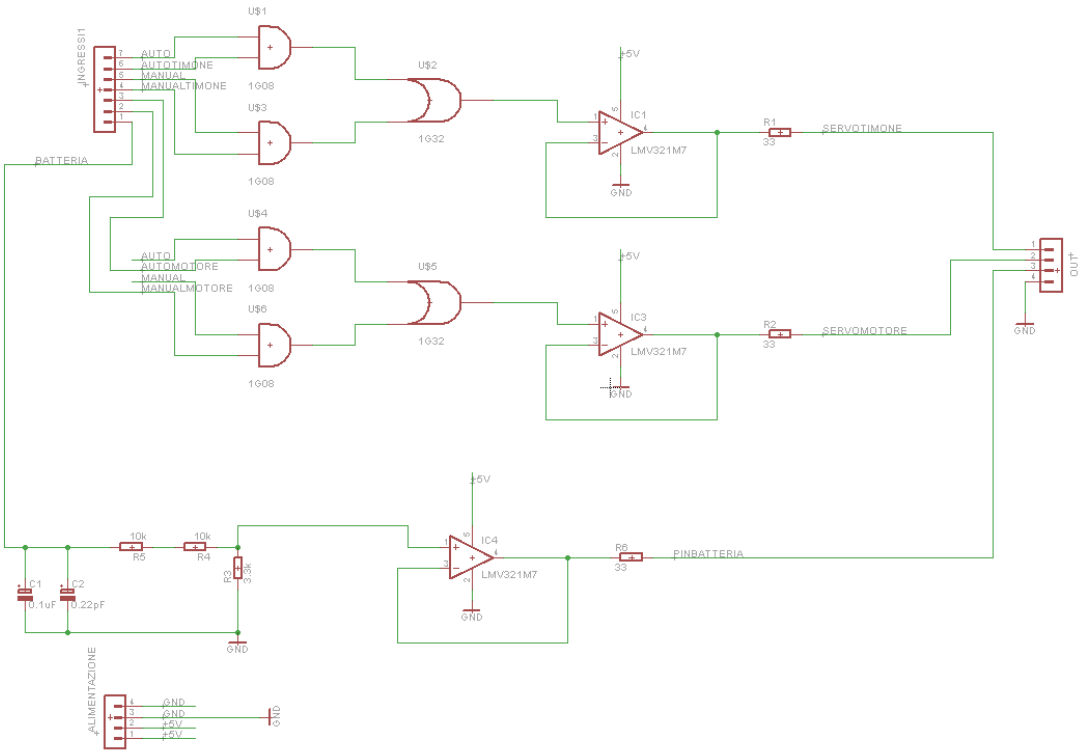


**Figura 7 Schema elettrico per il controllo**

Per la realizzazione è stata scelta la strada della bromografia che ha evitato fori e per minimizzare i costi dei componenti è stata adottata la tecnologia SMD di Figura 8.

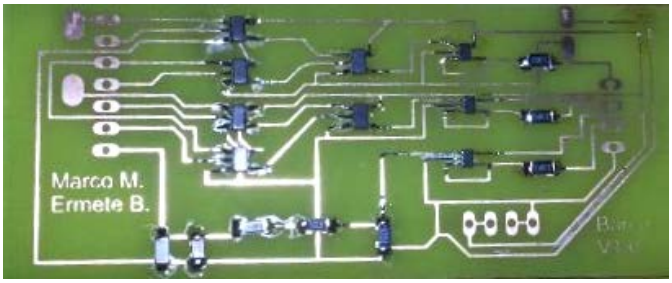


**Figura 8 Circuito per la scelta del controllo**

### 1.2.2 - La batteria

La batteria è stata scelta in modo da fornire la corrente richiesta, realizzando un buon rapporto peso/potenza e cercando il costo minimo. Sono stati scelti pacchi da usare in parallelo da 6 celle 6S Li-Poly da 3000mAh e con una scarica massima da 20C quindi 60A di scarica massima. Queste batterie hanno un costo di circa 20 euro.

## 1.3 - Descrizione della parte elettronica

La parte elettronica è il cuore e la mente del progetto. Il microcontrollore Arduino mega è sufficiente ad ottenere un ciclo di circa 7-8ms comunicante con il PC tramite l'Xbee. Per la navigazione è stato necessario un modulo GPS e una IMU entrambi a basso costo.

### 1.3.1 - Arduino mega

L'Arduino Mega 2560 è un progetto Open-Source basato sul microcontrollore ATmega2560. Dispone di 54 digitali pin di input / output (di cui 15 possono essere utilizzate come uscite PWM), 16 ingressi analogici, 4 UART (porte seriali hardware), un cristallo oscillatore a 16 MHz che funziona da clock, una connessione USB, un jack di alimentazione, un header ICSP, e un pulsante di reset. Contiene tutto il necessario per supportare il microcontrollore, per collegarlo in modo agevole a un computer con un cavo USB o alimentarla con un adattatore AC-DC o tramite una batteria per iniziare.

Arduino può percepire l'ambiente ricevendo input da una ampia gamma di sensori e può controllare motori e altri attuatori. Il microcontrollore sulla scheda è programmato utilizzando il linguaggio di programmazione Arduino (basato su Wiring) e l'ambiente di sviluppo Arduino (basato su Processing). [3]

In questo progetto Arduino Mega 2560, in Figura 9 utilizza vari sensori per verificare la posizione della barchetta e poi, con la risposta dei sensori i dati vengono elaborati e inviati agli attuatori per spostare la barca.

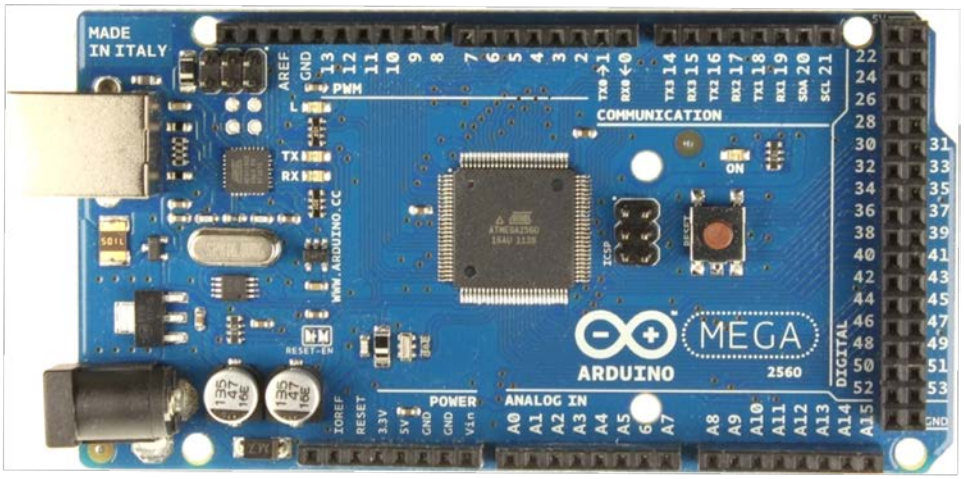


**Figura 9 Arduino Mega 2560**

### 1.3.2 - Xbee

XBee è un modulo radio illustrato in Figura 10. Questo dispositivo wireless è progettato per funzionare con il protocollo ZigBee. Questo modello funziona a una frequenza di 2,4 GHz e può inviare 250 kbps. Questa linea di dispositivi offre basso costo e basso consumo di energia. Il modulo scelto e utilizzato trasmette ad una distanza di 3,2 chilometri in campo aperto con una comunicazione bidirezionale. L'Xbee è stato configurato con un programma chiamato X-CTU per definire la velocità di comunicazione e tutti i parametri della rete guardando la guida[4]. X-CTU è un programma ed è possibile ricevere ed inviare dati in due codifiche diverse (HEX oppure ASCII).


**Figura 10 Modulo Xbee**

Questo modulo ha evitato l'implementazione di un protocollo ISO/OSI per la comunicazione WI-FI.

Effettuando una conversione Seriale/wifi non è stata necessaria alcuna libreria ed è stato collegato ad una porta seriale dell'Arduino e ad un convertitore USB/Seriale lato PC. Per altre informazioni si consulti [4].

### 1.3.3 - I sensori nella IMU

L'IMU GY-80 di Figura 11 integra a bordo un accelerometro, un magnetometro, un giroscopio ed un barometro. Ma non è necessario utilizzare il barometro dato che la barchetta non vola. L'IMU comunica con lo standard I²C, ci sono due pin di comunicazione SDA e SCL, presenti ovviamente anche sull'Arduino.

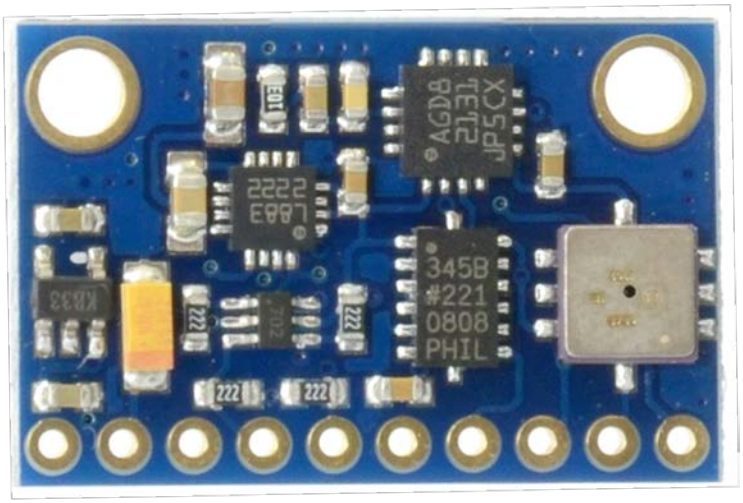


**Figura 11 IMU 10 DOF**

#### 1.3.3.1 La comunicazione I²C

Tutti i sensori dell'IMU GY-80 usano il protocollo di comunicazione I²C. Questo tipo di comunicazione ha il vantaggio di garantire il collegamento di più di un dispositivo collegando questi ultimi in parallelo al Bus.

Il protocollo I²C o anche I2C o IIC è seriale e sincrono, creato da Philips: è possibile scambiare dati utilizzando solo due fili: tramite uno si invia il clock e con l'altro il messaggio. Il clock è utilizzato per sincronizzare la comunicazione tra i dispositivi. In definitiva, con questo tipo di protocollo occorrono, per ogni sensore, due cavi per l'alimentazione e due cavi per la comunicazione e per questo è una buona soluzione

per i veicoli, poiché semplifica il cablaggio purché i dispositivi non siano troppo distanti tra loro (50cm massimo).

Per fare un Bus I²C, sono necessarie due resistenze da 5kΩ di pull-up collegate a +Vcc, una per ciascuno dei due segnali SDA ed SCL. Le resistenze di pull-up in Figura 12 sono usate per non lasciare il segnale "fluttuante" e quindi a livello sconosciuto.

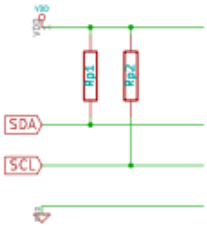


**Figura 12 Resistenza di pull-up**

La velocità di clock normalmente và dai 10kHz ai 100kHz, ma si possono trovare anche altre velocità sui dispositivi più recenti, come 400kHz e 3.4MHz. La velocità di comunicazione è limitata superiormente a questi valori, ma non esiste un limite inferiore; è anche possibile accendere un LED per verificare la trasmissione dati.

Nella comunicazione, il master controlla il segnale di clock sull'SCL e con l'SDA invia e riceve i dati. Quando viene iniziata una comunicazione si inviano sempre due segnali lo START prima del messaggio e lo STOP dopo il messaggio. Il segnale SCL deve andare basso per potere comunicare il dato dall'SDA. Quando l'SCL va alto e il segnale sull'SDA è cambiato, significa che il master sta iniziando a comunicare (START) o terminando (STOP) la comunicazione tra i dispositivi vedi Figura 13.

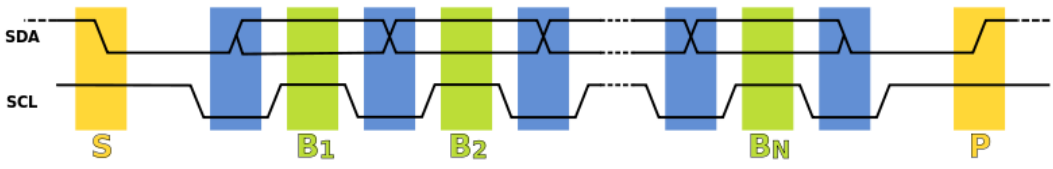


**Figura 13 Trasmissione I²C**

Di solito i segnali SCL e SDA sono a livello alto, a meno di iniziare o finire una comunicazione. Infatti per iniziare una comunicazione il master mette il segnale SDA al livello logico basso, mentre il livello logico di SCL è alto e inizia la comunicazione, questo è il comando START di Figura 14, e tutti i dispositivi sapranno che la comunicazione è iniziata con il dispositivo chiamato a comunicare.

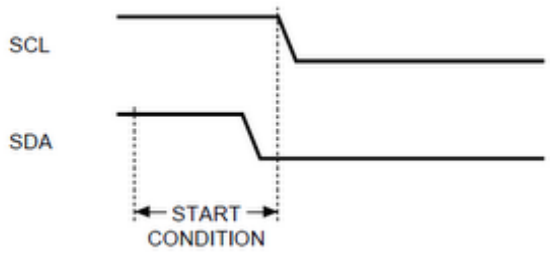


**Figura 14 START I²C**

Finita la trasmissione dei dati, lo stato di entrambi i segnali sono bassi, in questo modo il master può terminare la comunicazione. Per terminarla il master invia su SCL in livello logico alto e successivamente anche l'SDA và alto come mostrato in Figura 15. In questo modo si "comunica" che la comunicazione è terminata. Adesso i segnali SDA e SCL sono al livello alto e il master può far ripartire una comunicazione con un altro dispositivo con un altro segnale di START.

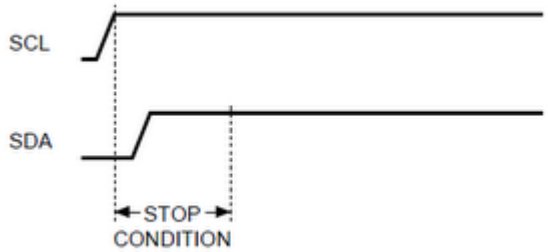


**Figura 15 STOP I²C**

Nella comunicazione tra il master e lo slave, il master manda un segnale di indirizzo subito dopo il segnale di START interpellando il dispositivo su tale indirizzo o registro. In questo modo tutti i dispositivi, dato che sono tutti sullo stesso Bus, sanno chi deve comunicare, successivamente il master riceve il messaggio dal dispositivo

associato al registro indicato dopo lo START, e riceve i pacchetti del bit più significativo al meno significativo.

In una comunicazione seriale, è necessario che lo slave comunichi al master se ha ricevuto correttamente la sequenza di dati. Per questo in determinati punti della comunicazione, il master aspetta un ACK e rilascia la linea SDA ed inizia a leggerne lo stato, lo slave trasmette il bit ACK (una ridondanza del messaggio per la verifica, come la somma di tutti i Byte inviati) esattamente come quando trasmette un bit. Durante un ciclo di SCL generato dal master, se la trasmissione è avvenuta correttamente e lo slave vuole trasmettere ACK, manterrà a livello logico basso SDA. In caso contrario lo slave manderà a livello logico alto SDA. Se il master non riceve ACK, potrà terminare la comunicazione con un comando di STOP richiedere lo stato del registro del dispositivo con un'altra con una nuova sequenza di START. [5]

In Arduino la comunicazione $I^2C$ avviene con una libreria che risulta semplice da usare[6].

#### 1.3.3.2 Accelerometro

ADXL345 Figura 16 è un piccolo componente, con basso consumo energetico, questo accelerometro mems triassiale alla risoluzione di 13 bit misura, fino a ± 16 g. Chiaramente questo dispositivo è sensibile anche all'accelerazione di gravità.

L'utente può scegliere la risoluzione della misura,ed è impostato sulla risoluzione di 13-bit a ±1g.


**Figura 16 L'accelerometro**

Per la comunicazione con questo dispositivo è stato scelto di utilizzare una libreria Open source disponibile in rete e facilmente reperibile. Per altri dettagli guardare il datasheet [7].

#### 1.3.3.3 Magnetometro

Il magnetometro HMC5883L in Figura 17, è un modulo per il rilevamento del campo magnetico con interfaccia digitale e viene utilizzato per applicazioni di navigazione dove c'è bisogno di una bussola di basso costo. Il suo pricipio di funzionamento è magneto-resistivo e ha un convertitore ADC che permette una precisione da 1 a 2 gradi come bussola. Questo è uno dei sensori utilizzati più sensibile ed affidabile a basso costo.

Integra un ADC a 12-Bit accoppiato con basso rumore AMR. Può aggiunge i 2 milli-Gauss di risoluzione.


**Figura 17 Il magnetometro**

Per la comunicazione con questo dispositivo si è scelto di utilizzare una libreria Open source disponibile in rete e facilmente reperibile. Per altri dettagli si consulti il datasheet[8].

### 1.3.3.4 Giroscopio

Il giroscopio L3G4200D di Figura 18 misura la velocità angolare nei 3 assi. Consente la scelta di 3 diverse scale ± 250 / ± 500 / ± 2000 dps. Ha una risoluzione di 16 bit, integra filtri passa basso ed un filtro passa alto con l'uso larghezza di banda selezionabile.

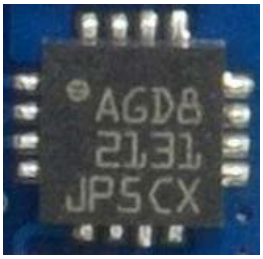


**Figura 18 Il giroscopio**

Per la comunicazione con questo dispositivo si è scelto di utilizzare una libreria Open source disponibile in rete e facilmente reperibile. Per altri dettagli si consulti il datasheet [9].

### 1.3.4 - GPS

Il MEDIATEK GPS 3329 in Figura 19 è un modulo ultra-compatto che lavora a 10Hz, la scelta del GPS è stata basata proprio sulla frequenza di aggiornamento, poiché la maggior parte dei GPS lavora a 1 Hz. Con tale frequenza è possibile avere migliori prestazioni del sistema, nonché un una risposta più rapida di correzione dell'errore. Inoltre si ha un'elevata sensibilità e capacità di monitoraggio in condizioni urbane.

**Figura 19 GPS Mediatek**

Il GPS comunica con il protocollo NMEA, è utilizzato per ricevere informazione tra dispositivi e PC, anche su diversi modelli di GPS per auto. Tramite questo protocollo di comunicazione il GPS invia dati di posizione, velocità e tempo ed altre variabili. La peculiarità di questo protocollo è quella di inviare sempre i dati indipendentemente da un segnale o carattere senza fermarsi. Invia un header iniziale per definire i tipi di dati che sono inviati, e successivamente i caratteri dell'informazione. L'informazione è indicizzata in questo ordine: latitudine, longitudine, altitudine, velocità, rotta, numero di satelliti, tempo, orario, precisione e tempo di risposta che con una libreria sono aggiornati.[10]

## 1.4 - Schema dei collegamenti

Lo schema dei collegamenti è rappresentato in Figura 20. Si è deciso, data la semplicità dei collegamenti, di integrare nello schema le fotografie dei componenti citati precedentemente.

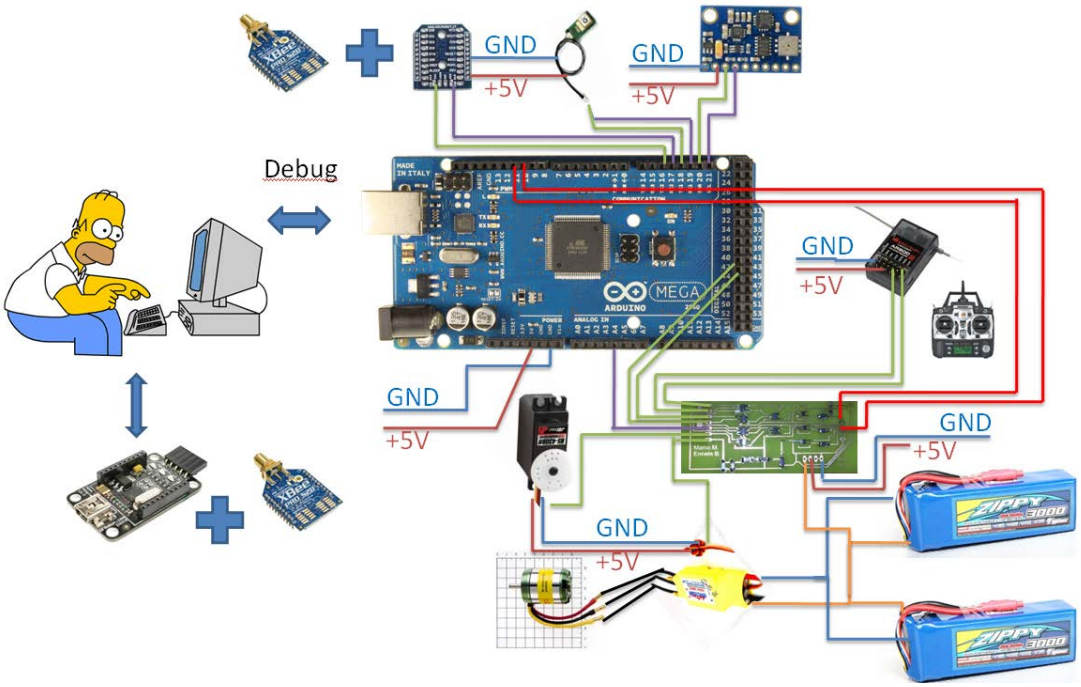


**Figura 20 Schema collegamenti**

L'Arduino è stato programmato tramite il PC con il quale è stato possibile anche effettuare alcune correzioni per ottenere un firmware stabile. Oltre a questo il PC

servirà, tramite l'interfaccia, a inviare e monitorare variabili tramite un piccolo convertitore USB/Seriale predisposto per l'alloggiamento della Xbee. Dal lato arduino l'Xbee sarà alloggiata su un traslatore di livello logico che integra un LDO per ottenere i 3,3V per alimentare l'Xbee connessa ai pin 16(TX) e 17(RX) dell'Arduino invertiti sul traslatore. Le due batterie in parallelo sono connesse all'ESC che alimenta anche tutti i dispositivi a 5V oltre a comandare il motore per la propulsione. Per monitorare la carica della batteria è stata portata sulla scheda per il controllo la tensione della batteria, che dopo averla limitata alla tensione misurabile dagli ingressi analogici è stata amplificata  e portata sul pin A4. Sia il motore che il Servo, alimentato a 5V, sono connessi alla scheda per la scelta del controllo che prenderà direttive in merito dai pin digitali 40 (se a livello logico 1 pilota manuale) o 42 (se a livello logico 1 pilota automatico). Il segnale PWM proveniente dalla ricevente, connessa automaticamente al radiocomando, fornirà il duty per il controllo manuale dai pin 9, per il timone, e 10, per il motore, l'Arduino modulerà il duty nel caso di pilota automatico, ma tali segnali di controllo saranno gestiti dalla scheda per il controllo e pertanto sono connessi ad essa. La piattaforma inerziale è alimentata a 5V, ma sulla scheda è presente un LDO per l'alimentazione degli integrati a 3,3V, e comunicando con il protocollo $I^2C$ dai pin 20 SDA e 21 SCL è presente anche un traslatore di livello logico con le relative resistenze di Pull-U. Il GPS è anch'esso alimentato a 5V ed è connesso alla seriale 1 dell'Arduino dai Pin 18(TX) e 19(RX). Al fine di ottenere collegamenti elettrici a prova di vibrazioni che saranno sicuramente presenti, pur trattandosi di una barca ma essendo un veicolo, tutti i collegamenti sono stati fermati con nastro adesivo e le schede, per evitare spiacevoli ossidazioni dall'umidità, sono state protette con un prodotto spray.

# 2 - Il Sistema operativo

La scelta è stata quella di programmare Arduino con il suo software per garantire la minima occupazione di memoria e snellire il ciclo. L'interfaccia utente è stata realizzata con le librerie Qt [11] che sono richieste ultimamente in ambito industriale e da molti datori di lavoro.

## 2.1 - L'idea di base

L'idea di base è quella di realizzare una barchetta che possa essere comandata sia con il radiocomando, sia in modo autonomo fornendole le coordinate GPS oppure registrando dei punti mentre è radiocomandata. Sono stati scelti quattro punti, A B C e D, che la barchetta dovrà raggiungere in modo sequenziale e che potranno essere ripetuti impostando i cicli. Se la barchetta entrerà nell'intervallo di latitudine e longitudine definito dalla variabile RaggioGPS il programma cambierà il riferimento da raggiungere. È stata inoltre definita una variabile per consentire alla barca di aspettare a ripartire in ogni punto per un certo intervallo di tempo, per fare questo è necessario imporre prima del tempo di attesa la scrittura dello stato di quiete del timone e del motore. Inoltre è presente un altro punto: la Home, che dovrà essere impostato come riferimento alla fine dei cicli oppure, nel caso richiesto, se la barchetta perde il segnale dal PC, in tal caso il pilota automatico dovrà essere disabilitato con, appunto, la variabile Pilota. La comunicazione è stata fatta sincrona con caratteri iniziali per il settaggio, modifica dei parametri di controllo e la richiesta di invio dati per monitorare delle variabili effettuata tramite i moduli Xbee. L'implementazione del codice è la seguente:

```
DifLat = RifLat - Lat;        //calcolo le distanze dal punto per trovare l'angolo

DifLon = RifLon - Lon;        //calcolo le distanze dal punto per trovare l'angolo

DifLonASS = abs(DifLon); //valore asoluto della distanza

DifLatASS = abs(DifLat);
```

```
  if ((Cicla == -1) && (DifLonASS <= RaggioGPS) && (DifLatASS <= RaggioGPS))
{
 Pilota = 0;
}
if ((DifLonASS <= RaggioGPS) && (DifLatASS <= RaggioGPS))
{
 if ((RifLat == LatD) && (RifLon == LonD) && (Cicla >= 1))
 {
  RifLat = LatA;
  RifLon = LonA;
  Cicla--;
  if (DelayD != 0)
  {
  Timone.write(90);
  Motore.write(0);
  delay(DelayD);
  }
 }
 else if ((RifLat == LatD) && (RifLon == LonD) && (Cicla == 0))
 {
  RifLat = LatH;
```

```
    RifLon = LonH;
    Cicla--;
    if (DelayD != 0)
    {
    Timone.write(90);
    Motore.write(0);
    delay(DelayD);
    }
}
  else if ((RifLat == LatC) && (RifLon == LonC))
  {
    RifLat = LatD;
    RifLon = LonD;
    if (DelayC != 0)
    {
    Timone.write(90);
    Motore.write(0);
    delay(DelayC);
    }
  }
  else if ((RifLat == LatB) && (RifLon == LonB))
  {
```

```
      RifLat = LatC;
      RifLon = LonC;
      if (DelayB != 0)
      {
      Timone.write(90);
      Motore.write(0);
      delay(DelayB);
      }
    }
    else if ((RifLat == LatA) && (RifLon == LonA))
    {
      RifLat = LatB;
      RifLon = LonB;
      if (DelayA != 0)
      {
      Timone.write(90);
      Motore.write(0);
      delay(DelayA);
      }
    }
}
```

La rotta desiderata sarà definita dal vettore dato dalle coordinate del punto di arrivo rispetto alla posizione rilevata dal GPS della barca calcolandone l'angolo dal nord

geografico. Sarà necessario sapere l'orientazione dal nord della barca, ma a causa della declinazione magnetica e del verso del campo magnetico terreste alla nostra latitudine è stato necessario l'utilizzo della piattaforma inerziale in abbinamento al filtro di Kalman per la stima dello stato.

## 2.2 - Stima dello stato per ottenere $\theta b$

Per determinare la rotta reale è necessario determinare l'angolo di orientazione $\theta b$ dal nord preso da +180° a -180°. A tale scopo, si utilizza il magnetometro triassiale che fornisce una misura del campo magnetico nei tre assi della piattaforma inerziale. Tale misura non è sufficiente per ottenere una stima corretta dell'angolo di orientazione rispetto al nord. In questo paragrafo verrà analizzata la soluzione proposta per effettuare tale stima.

### 2.2.1 - Declinazione magnetica

La declinazione magnetica rappresenta l'angolo di variazione tra il nord geografico e quello magnetico. In Figura 21 è rappresentata una carta della declinazione.

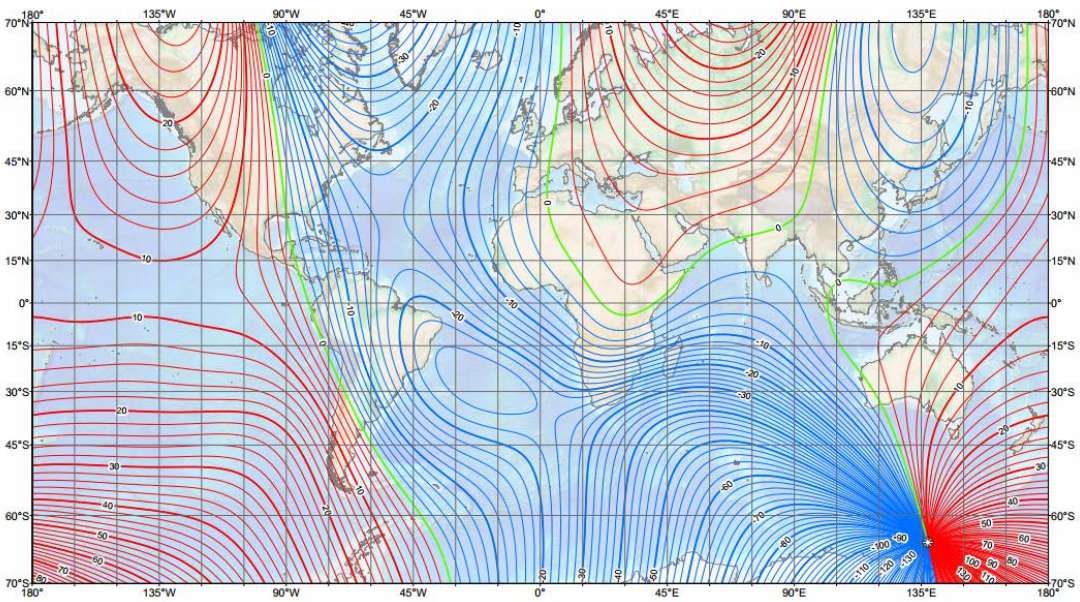


**Figura 21 Mappa della declinazione magnetica**

Infatti il polo nord magnetico e il polo nord geografico non sempre sono coincidenti. Non solo: nella storia, oltre all'inversione dei poli magnetici, è stata registrata una grossa variazione dei vettori magnetici nelle varie zone della terra che rappresentano una vera e propria mappa in continua lenta variazione con le varie zone ed i rispettivi angoli di differenza[12]. La situazione attuale è rappresentata in Figura 21.

Le righe verdi rappresentano il punto di declinazione magnetica pari a zero, quelle rosse con un angolo positivo e quelle blu con un angolo negativo. Come è possibile vedere in foto e verificandolo con una delle tante mappe o siti che offrono questo servizio disponibili sul web [13], è possibile vedere che l'Italia si trova in una posizione prossima allo zero, infatti risulta avere una declinazione magnetica di circa +2°. Tale valore è stato sommato alla stima, ma ipotizzando di mandare la barchetta molto distante dalla nostra latitudine sarebbe necessario integrare una libreria già ottimizzata per questa correzione, ma che si è scelto di non integrare per il momento. Si è scelto di sommare alla stima dello stato di $\theta b$ una variabile di declinazione magnetica.

### 2.2.2 - Il campo magnetico terrestre

Il campo magnetico terrestre può essere modellato come una calamita molto potente posizionata al centro della terra come mostrato in Figura 22.

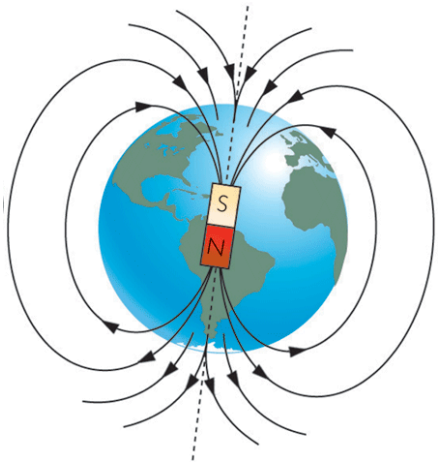


**Figura 22 Campo magnetico terrestre**

Proprio come farebbe un magnete, che disperde le linee di campo nello spazio provenienti ed uscenti dal Polo Sud ed entranti nel Polo Nord, anche la Terra a seconda della distanza e dell'inclinazione rispetto all'asse tra Polo Nord e Polo Sud magnetico manifesta vettori di campo magnetico che saranno orizzontali all'equatore, ma verticali ai poli. Alla nostra latitudine è stata registrata una pendenza di circa 60° verso il basso come è intuibile dalla figura.

Questo comportamento modifica l'angolo di stima dal nord nel caso in cui la barchetta avesse un'orientazione diversa, ma possibile, con certi angoli di beccheggio o rollio diversi da zero. Per questo è stato necessario l'utilizzo di matrici di rotazione per proiettare il vettore del campo magnetico misurato dal magnetometro sul piano parallelo all'acqua.

### 2.2.3 - Calibrazione e accettazione

Pur trattandosi di un magnetometro costituito dallo stesso wafer di silicio, le misure di campo magnetico sui tre assi risultavano leggermente diverse. È stata necessaria pertanto, come accade di consueto negli strumenti di misura, una taratura della bussola. Sono stati quindi fatti stampare valori registrati dal magnetometro sui tre assi ruotandolo in tutti i versi e con questi valori è stato fatto un grafico 3D in Figura 23.

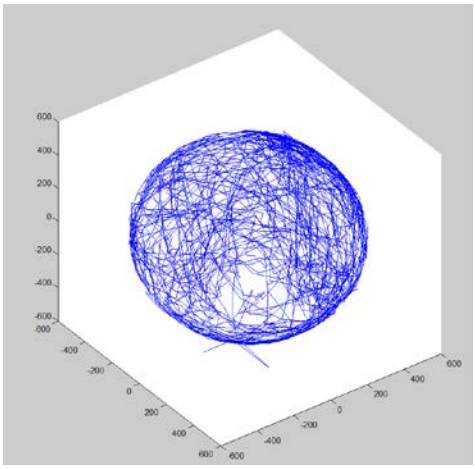

**Figura 23 Sferoide di calibrazione bussola**

Le unità di misura, essendo valori digitali a fondo scala, non hanno una precisa dichiarazione a meno di moltiplicarli per la sensitività del magnetometro. Come è possibile vedere dalla Figura 23 alcuni punti non sono realmente quelli misurati, ma ogni tanto il magnetometro sbaglia su qualche asse andando fuori dallo sferoide che descrive il vettore magnetico. Per questo è stata fatta un'accettazione dei valori all'interno dell'intersezione data dall'equazioni di due sfere, una interna e una esterna. Successivamente, trovati i massimi e minimi su ogni asse, è stata sommata una variabile costante per la calibrazione della bussola. Il codice è il seguente:

```
if((scaled.XAxis > -600) && (scaled.XAxis < 600))
  {
    xxx = scaled.XAxis + 19;
  }
  if((scaled.YAxis > -600) && (scaled.YAxis < 600))
  {
    yyy = scaled.YAxis + 99;
  }
  if((scaled.ZAxis > -600) && (scaled.ZAxis < 600))
  {
    zzz = scaled.ZAxis -30;
  }
  //Accettazione di valori di misura con coppia di sferoidi
  Rmagnetico = sqrt((xxx*xxx) + (yyy*yyy) + (zzz*zzz));
  if((Rmagnetico > sferamin) && (Rmagnetico < sferamax) )
{
```

```
  YYY = yyy;

  XXX = xxx;

  ZZZ = zzz;

}
```

### 2.2.4 - Stima del beccheggio e rollio

Per ottenere l'angolo di beccheggio e rollio è stata utilizzata la libreria Kalman.h il cui funzionamento è descritto successivamente. Tale strumento consente di stimare lo stato di due variabili in ingresso al sistema e, per alleggerire il carico computazionale, e verificare che sia possibile la computazione nei tempi massimi, si è scelto di usare un filtro di Kalman semplificato ad una matrice 2x2 ripetuta per i casi necessari.

Sono stati quindi istanziati kalAngleX e kalAngleY e sono stati calcolati gli angoli dalle misure di accelerazione considerando il vettore di accelerazione di gravità.

```
accXangle = (atan2(Yg,Zg)*RAD_TO_DEG);

 accYangle = (atan2(Xg,Zg)*RAD_TO_DEG);
```

Tale misura, durante la navigazione, accuserebbe la dinamica di accelerazione durante le virate nelle componenti X e Y e quindi anche il corrispondente angolo risulterebbe alterato. Per la misura sui giroscopi è stata fatta una correzione dei segni per ottenere gli angoli desiderati e tale misura fornisce solo una scala di valori.

```
double gyroXrate = gyroX/114;

 double gyroYrate = -gyroY/114;

 double gyroZrate = -gyroZ/114;
```

La misura dei giroscopi è infatti da moltiplicare per la Sensitività rispetto al fondo scala e tale valore è stato preso da datasheet [7] . Inoltre, usando solo questa misura il

rumore, e la costante che risulta dall'integrale per passare dalla velocità angolare all'angolo farebbero sicuramente derivare la misura a valori scorretti.

Gli angoli desiderati risultano i seguenti:

```
kalAngleX = kalmanX.getAngle(accXangle, gyroXrate, (double)(micros()-timerR)/1000000);
timerR = micros();

kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, (double)(micros()-timerP)/1000000);
timerP = micros();
```

Il timer aggiorna una variabile in modo che il filtro conosca l'intervallo di tempo su cui calcolare la derivata all'interno del filtro. Tali angoli sono proprio beccheggio e rollio desiderati.

### 2.2.5 - Ottenere $\theta b$

Ricapitolando: è stata fatta la taratura della bussola in modo da ottenere misure sugli assi X, Y e Z uguali di massimi e minimi; è stata fatta un'accettazione dei valori con 2 sfere una interna e una esterna per evitare letture errate, ma il solo valore dell'angolo dal nord con un certo beccheggio o rollio falserebbe il reale angolo in quanto il vettore campo magnetico terrestre alla nostra longitudine è di 60°, quindi è stata fatta una stima del beccheggio e rollio con il filtro di Kalman che limita il rumore dei giroscopi e non risente delle dinamiche di moto sugli accelerometri. Per ottenere la stima di orientazione dal nord è stata quindi effettuata una proiezione delle misure sul piano parallelo alla superficie dell'acqua con le matrici di rotazione in terna fissa. Per il solo beccheggio si ha la rotazione di Figura 24 indicando con p (pitch) il valore dell'angolo di beccheggio:

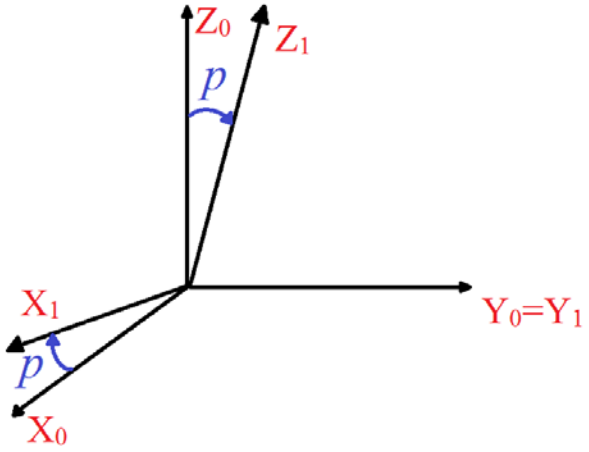


**Figura 24 Rotazione sul beccheggio**

la cui matrice di rotazione risulta:

$$T_p^0 = \begin{bmatrix} \cos p & 0 & -\sin p \\ 0 & 1 & 0 \\ \sin p & 0 & \cos p \end{bmatrix}$$

Sul rollio la rotazione delle terne è descritta dalla Figura 25.

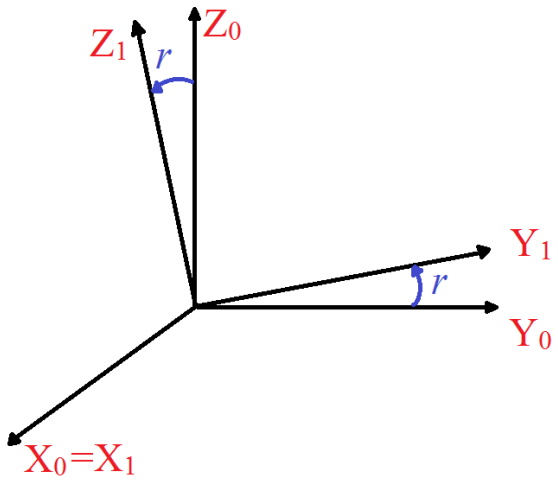


**Figura 25 Rotazione sul rollio**

La cui matrice di rotazione (la r rappresenta l'angolo di rollio):

$$T_r^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r & -\sin r \\ 0 & \sin r & \cos r \end{bmatrix}$$

La matrice di rotazione totale risulta essere:

$$T_T^0 = T_r^0 + T_p^0 = \begin{bmatrix} 1+\cos p & 0 & -\sin p \\ 0 & 1+\cos r & -\sin r \\ 0 & \sin r & \cos r + \cos p \end{bmatrix}$$

E dato che servivano solo $X_0$ e $Y_0$ per individuare dove si trova la proiezione della misura è stato implementato il codice:

CYYY = (YYY * (1+ cos_roll) - (ZZZ * sin_roll));

CXXX= (XXX * (1+cos_pitch) - (ZZZ * sin_pitch));

E con la tangente calcolare l'angolo dal nord:

TetaK = (atan2(CYYY,CXXX)* RAD_TO_DEG);

Questa misura era però abbastanza affetta da errori e l'oscillazione era elevata in previsione che controllasse il timone, quindi, analogamente a quanto fatto precedentemente è stato riapplicato il filtro di Kalman, tra la velocità angolare in Z e la stima di orientazione dal nord precedentemente proiettata sul piano orizzontale stimando rollio e beccheggio con il medesimo metodo. ed è stata notevolmente migliorata la stima:

kalAngleZ = kalmanZ.getAngle(TetaK, gyroZrate, (double)(micros()-timerZ)/1000000);
timerZ = micros();

ottenendo una piccola oscillazione di mezzo grado circa.

Nel codice sono state inoltre effettuate prove di filtraggio per verificare che il filtro di Kalman non necessiti di ulteriori filtri in ingresso.

### 2.2.6 - Schema a blocchi, stima e controllo

Una ulteriore e immediata visione dei collegamenti per la stima dell'angolo di rotta e del controllo è presente in Figura 26. L'accelerometro triassiale fornisce una misura

dell'accelerazione di gravità nei tre assi comprensiva di disturbi derivanti dal moto e dalla dinamica della barca.

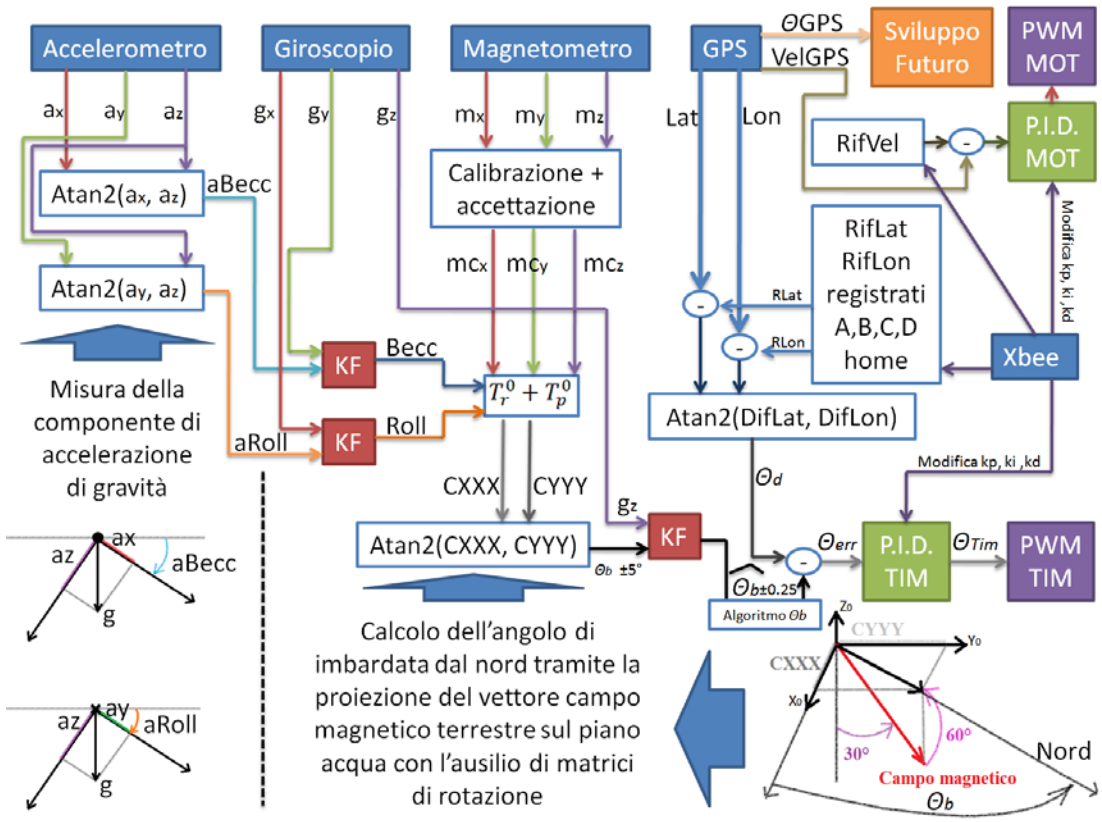


**Figura 26 Schema a blocchi**

Tali misure sono state trasformate con la funzione Atan2 che restituisce l'angolo di beccheggio e rollio (aBecc ed aRoll) affetto da rumori di misura e disturbi dovuti alle accelerazioni durante i transitori di moto. L'inserimento del filtro di Kalman, che effettua una fusione con il giroscopio nelle rotazioni corrispondenti, annulla i disturbi del moto misurati degli accelerometri e porta a zero l'errore di deriva dei giroscopi. Ottenendo quindi beccheggio e rollio(Becc e Roll) molto precisi 0.03° circa di oscillazione sulla misura statica. Effettuata la calibrazione comprensiva di accettazioni della misura per il magnetometro triassiale (da mx, my, mz a mcx, mcy, mcz) è stato proiettato sul piano acqua (piano $X_0$ e $Y_0$) il vettore campo magnetico (in rosso) che alla nostra latitudine punta 60° verso il basso (http://magnetic-declination.com/) attraverso l'ausilio delle matrici di rotazione e del beccheggio e rollio precedentemente stimati. Successivamente è stata riutilizzata la funziona atan2 per il calcolo dell'angolo dal nord ottenendo una oscillazione di circa 5°. Per questa

ragione è stato necessario riapplicare un filtro di Kalman analogo ai precedenti per la stima della rotta della barca aggiungendo la velocità di rotazione in z data dal giroscopio ed ottenendo una oscillazione in condizioni statiche di meno di 0,25°. Per il PID applicato al timone è stata ottenuta la rotta desiderata, (il riferimento da inseguire) tramite la funzione Atan2 dalla differenza tra la posizione della barca e le coordinate registrate di riferimento che la barca deve seguire chiamate A, B, C, D e H di home precedentemente descritte nell'algoritmo per l'inseguimento dei punti. La stima della rotta della barca è posta in retroazione al sistema come misura dell'uscita con l'aggiunta dell'algoritmo per virare sempre nel verso più conveniente descritto precedentemente. Il PID esegue una correzione sul timone che va a zero inclinazione quando la barca va dritta. Dalla xbee è possibile modificare online i parametri di controllo ed è possibile fornire il riferimento di velocità che il PID del motore deve inseguire misurando la velocità tramite il GPS che è stata posta in retroazione. Tale misura potrebbe essere ulteriormente filtrata inserendo la misura di accelerazione lungo la direzione della rotta fornita dall'accelerometro integrando un nuovo filtro di Kalman. Anche la stima della rotta della barca potrebbe essere migliorata inserendo la variabile di rotta data dal GPS con un ulteriore filtro di Kalman. Entrambi questi ultimi filtri di Kalman sarebbero essenzialmente diversi sia nella parte di predizione che nelle matrici del sistema rispetto a quelli adottati attualmente.

I tre filtri di Kalman implementati risultano identici per la formulazione del problema di filtraggio e quindi per le matrici A, B, C e D. Anche per quanto riguarda le variabili in ingresso e per la stima ottenuta della variabile di uscita risultano morfologicamente identici. In ingresso ai filtri è necessario dare l'angolo e la velocità angolare su tale angolo in ogn'uno dei filtri di Kalman Impiegati, mentre la stima in uscita è proprio l'angolo non affetto da rumore di misura dei singoli sensori MEMS e esente da disturbi del moto generati durante le traiettorie inseguite dal controllo.

Nel dettaglio si ha un predittore composto da:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + B_k\dot{\theta}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k = \begin{bmatrix} \theta + \Delta t(\dot{\theta} - \dot{\theta}_{bias}) \\ \dot{\theta}_{bias} \end{bmatrix}$$

$$\hat{y}_{k|k-1} = [1 \quad 0]\hat{x}_{k-1|k-1} + v_t$$

Ottenendo quindi tutte le matrici che compongono il sistema.

Nel caso del filtro per il rollio la variabile descritta come $\theta$ risulta essere aRoll mentre $\dot{\theta}$ è la velocità angolare misurata dal giroscopio $g_x$. In uscita si ha l'angolo del rollio esente dai disturbi generati dal moto.

Nel caso del filtro per il beccheggio la variabile descritta come $\theta$ risulta essere aBecc mentre $\dot{\theta}$ è la velocità angolare misurata dal giroscopio $g_y$. In uscita si ha l'angolo del beccheggio esente dai disturbi generati dal moto.

Nel caso del filtro per l'imbardata la variabile descritta come $\theta$ risulta essere l'angolo generato dalla misura del campo magnetico proiettato sul piano acqua $X_0$,$Y_0$ ottenuto mediante l'utilizzo di matrici di rotazione, mentre $\dot{\theta}$ è la velocità angolare misurata dal giroscopio $g_z$. Questo ultimo filtro ha come uscita proprio l'angolo di rotta desiderato.

## 2.3 - Il filtro di Kalman

Il filtro di Kalman è lo stimatore ottimo dello stato. Con questo si intende che, matematicamente, non è possibile ottenere una stima di una delle variabili che compongono lo stato, o di una combinazione di esse, in modo migliore (nel senso MMSE) rispetto al filtro di Kalman. Nel caso specifico è stata adottata una "libreria" dato che si è ritenuto di avvalersi sempre delle ultime risorse disponibili per evitare di fare un lavoro già fatto da altre persone(anche per questo è difficile che una persona si cimenti programmando in Assembler), anche se è sempre necessario sapere cosa si sta implementando ed è sempre necessario avere le stesse nozioni ed essere buoni programmatori almeno quanto l'autore della "libreria". Infatti è normale che inizialmente non sia possibile compilare e solo dopo aver analizzato il codice è possibile venire a capo della soluzione al problema. Nel caso specifico l'header kalman.h preso spunto dalla IMU open ARDUIMU i cui codici sono presenti su githab, effettua una stima dello stato degli angoli di beccheggio e rollio con le istanze

dichiarate nel codice di Arduino. Con la misura dell'angolo dato dalla misura dell'accelerazione di gravità nelle componenti Z,Y per il rollio e Z,X per il beccheggio che riporta a zero l'errore di integrazione dei giroscopi che forniscono la misura, affetta da rumore a media nulla, della velocità di rotazione angolare sugli assi opposti. Sono stati quindi istanziati 3 differenti filtri di Kalman che funzionano in modo analogo. Uno per stimare il beccheggio, uno per stimare il rollio, e dalla combinazione di questi due con le relative matrici di rotazione un'altro per stimare l'angolo dal nord. Per alleggerire il calcolo si è scelto in primis di implementare una stima non completa di tutto il vettore delle variabili da stimare ed in più si è scelto di inserire dove possibile un termine di osservazione che eliminasse lo stato che non era necessario allo scopo ovvero l'angolo da stimare.

Nel particolare un sistema LTI. è descritto dalle equazioni di Gauss-Markov:

$$\begin{cases} \dfrac{dx(t)}{dt} = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

Dato che è necessario stimare lo stato ovvero la matrice x degli ingressi è opportuno considerare la prima eq. che in forma matriciale diventa:

$$\hat{x}_t = A\hat{\mathrm{x}}_{\mathrm{t}-1}(t) + Bu(t) + W(Q)$$

in quanto si stima lo stato attuale rispetto ai campioni precedenti ed è stato inserito un termine W di rumore gaussiano a media nulla e di varianza Q non nulla sulla diagonale. Da questo passo, detto di predizione, si evince che sarà necessario inizializzare delle variabili o nel modo migliore per far convergere le stima al valore più rapidamente possibile oppure a zero nel caso si ipotizzi che il sistema da stimare possa dare inizialmente valori non congrui alla stima. Il passo zero della libreria prevede proprio di inizializzare la matrice di covarianza e il bias a zero.

Le variabili in ingresso alla libreria, che restituisce l'angolo, sono definite come:

Angolo[°] = Istanza.getAngle(newAngle[°], newRate[°/s], dt[s]);

ovvero l'angolo ottenuto dagli accelerometri, la velocità di rotazione su tale angolo, e il tempo di esecuzione del ciclo. Pur essendo queste le variabili in ingresso alla libreria, il vettore dello stato da stimare è stato scelto dall'angolo, che è quello che realmente ci interessa, e dal bias del giroscopio. Questo permette di stimare la deriva del giroscopio, sottraendo tale valore alla velocità angolare che, integrata, fornirà un valore dell'angolo senza tale deriva mettendo insieme i sensori e annullando i vari disturbi sulle accelerazioni, presenti in un veicolo, per il calcolo dell'angolo da atan2 ed eliminando la deriva dei giroscopi.

Inoltre nel caso della libreria, dato che è possibile istanziare la stima di più variabili si ha una generalizzazione che comporta una leggera pesantezza di scrittura matematica:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + B_k\dot{\theta}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k = \begin{bmatrix} \theta + \Delta t(\dot{\theta} - \dot{\theta}_{bias}) \\ \dot{\theta}_{bias} \end{bmatrix}$$

Il vettore *B* rappresenta il tempo di esecuzione ed è conosciuto.

È possibile implementare semplicemente con:

```
rate = newRate - bias;
angle += dt * rate;
```

Come è previsto dalla teoria del filtro di Kalman [14] il passo successivo(stima a priori dell'errore di covarianza) è quello di aggiornare la matrice di covarianza ovvero:

$$P_{k|k-1} = AP_{k-1}A^T + W(Q_k)$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_{bias}} \end{bmatrix} \Delta t =$$

$$= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_{bias}} \Delta t \end{bmatrix}$$

Ovviamente i termini della matrice sono calcolati da quelli precedenti e se inizializzati a zero, saranno zero solo al primo ciclo. Implementata con il codice:

```
P[0][0] += dt * (dt*P[1][1] - P[0][1] - P[1][0] + Q_angle);
P[0][1] -= dt * P[1][1];
P[1][0] -= dt * P[1][1];
P[1][1] += Q_gyroBias * dt;
```

Ora è necessario effettuare l'aggiornamento e il passo successivo è quello di calcolare il guadagno di Kalman. Per farlo occorre calcolare prima:

$$S_k = HP_{k|k-1}H^T + R = [1 \quad 0]\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + R = P_{00_{k|k-1}} + var(\theta)$$

Dove H è il vettore di osservazione ed R è la varianza del rumore di misura considerata costante. Nel codice:

```
S=P[0][0]+R_angle;
```

Il guadagno di Kalman si calcola con la formula:

$$K_k = P_{k|k-1}H^T S_k^{-1}$$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} S_k^{-1} = \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{S_k}$$

E quindi in codice dato che S diventa un numero in questo caso:

```
K[0] = P[0][0] / S;
K[1] = P[1][0] / S;
```

Ora è possibile aggiornare la stima dello stato calcolando prima l'innovazione:

$$\tilde{\boldsymbol{y}}_k = z_k - H\hat{x}_{k|k-1} = z_k - [1 \quad 0]\begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k|k-1} = z_k - \theta_{k|k-1}$$

Fatta con il codice:

```
y = newAngle - angle;
```

Si noti è che questa operazione era possibile anche precedentemente senza alterare il codice e quindi il risultato. Adesso è effettuato l'aggiornamento dello stato:

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k \tilde{\boldsymbol{y}}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{\boldsymbol{y}}_k = \begin{bmatrix} \theta \\ \dot{\theta}_{bias} \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \tilde{\boldsymbol{y}} \\ K_1 \tilde{\boldsymbol{y}} \end{bmatrix}_k$$

Con il codice:

```
angle += K[0] * y;
bias += K[1] * y;
```

Ora è necessario riaggiornare la matrice di covarianza, data dalla semplificazione a posteriori $K_k S_k K_k^T = P_{k|k-1} H K_k^T$, con il guadagno di Kalman ed ottenerla alla esecuzione successiva con la relativa correzione della stima dell'errore:

$$P_{k|k} = (I - K_k H) P_{k|k-1} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{10} & K_1 P_{11} \end{bmatrix}$$

```
P[0][0] -= K[0] * P[0][0];
P[0][1] -= K[0] * P[0][1];
P[1][0] -= K[1] * P[0][0];
P[1][1] -= K[1] * P[0][1];
```

Per concludere i 3 parametri che sono descritti come costanti sono:

```
const double Q_angle = 0.001;
const double Q_gyroBias = 0.003;
const double R_angle = 0.03;
```

questi descrivono la varianza dell'accelerometro, o meglio della combinazione di due misure dell'accelerometro con atan2; la varianza del giroscopio e la varianza del

rumore di misura. Questi termini alle volte, in alcuni semiconduttori mems di qualità, sono indicizzati con un registro su $I^2C$ ed è possibile richiamarli e modificarli ad ogni ciclo di calcolo.

## 2.4 - L'interfaccia

L'interfaccia è stata fatta con Qt creator, ottimo strumento per creare interfacce utente. La comunicazione è bidirezionale per monitorare alcune variabili, in ricezione dati, e per inviare i settaggi e modificare online i valori dei guadagni del PID. L'Xbee è collegato ad un convertitore USB/Seriale connesso al PC mentre un traslatore di livello logico da 3V a 5V funziona da interfaccia di comunicazione sulla barchetta per Arduino. Un menù a tendina è stato inserito per scegliere la modalità di inserimento dei punti di riferimento, che potranno essere scelti sia da PC sia registrando le coordinate di dove si trova la barchetta radiocomandata. Inoltre è possibile scegliere il tipo di controllo, se automatico o manuale con il radiocomando, se attivare o disattivare il PID motore dando un riferimento direttamente al motore per tarare il PID timone al meglio, effettuare un reset, far tornare la barchetta alla home se perde il segnale con il PC oppure continuare la navigazione, dare un OffSett al timone e alla bussola per la declinazione, o ancora disattivare o attivare l'inner loop e settare l'antiwindup sia del timone che del motore. Inoltre è possibile tarare il raggio entro il quale cambia riferimento, quanti cicli deve fare la barchetta e la sosta in secondi per ogni punto. La comunicazione può essere scelta con vari passi temporali ma con 50ms è ottima e non perde il sincronismo. La scelta di monitorare la batteria è indispensabile per ottenere la massima capacità di carica nel tempo, infatti se le batterie Li-Poly scendano sotto una certa soglia si danneggerebbero e a forza di arrivare al limite diminuirebbero la loro capacità rispetto a quella nominale. Le variabili monitorate possono essere viste in Figura 26.

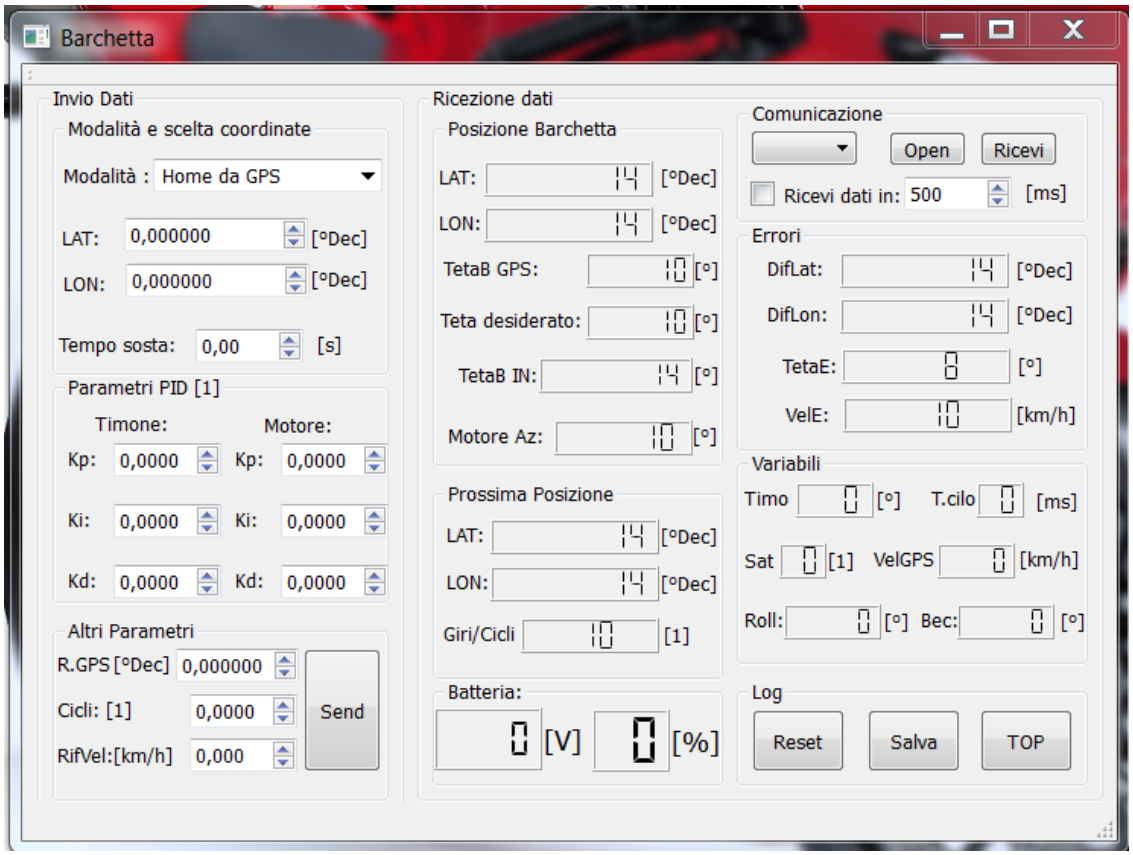


**Figura 27 Interfaccia utente**

Inoltre è possibile registrare alcune variabili facendo un log salvandolo normalmente per avere più dati a disposizione e se particolarmente buono salvarlo con il nome Top.

Anche volendo inserire il codice implementato per l'interfaccia sarebbe totalmente inutile in quando il linguaggio prevede l'inserimento di eventi richiamati dai pulsanti tramite metodi creati appositamente dall'ambiente di sviluppo e risulterebbe incomprensibile.

Inoltre per la stessa ragione vengono creati vari formati e strutture in automatico su file diversi.

# 3 - Il controllo

Il controllo è effettuato con 2 PID. uno per il timone e uno per la velocità.

La soluzione di adoperare il PID. sulla velocità è semplificativa ed evita la generazione di traiettoria per cercare di semplificare il più possibile i calcoli da effettuare sul microcontrollore.

Il controllore PID. è stato uno dei primi controllori implementati ed il funzionamento è semplice se paragonato ad altri controllori, ma data questa semplicità non è possibile controllare anche come mandare a zero l'errore e risulta quindi troppo semplice per alcuni sistemi. Tale controllore è l'acronimo di proporzionale, integrale e derivativo infatti genera un segnale proporzionale all'errore, più un termine proporzionale all'integrale del segnale errore, più un termine proporzionale alla derivata del segnale errore. Ci sono dei guadagni da applicare ai tre termini chiamati generalmente Kp per il termine proporzionale, Ki per in termine integrale, Kd per il termine derivativo. Per fare ciò è stata usata una libreria PID_V1 che utilizza l'aggiornamento di una variabile temporale ogni qualvolta si esegue il calcolo dell'uscita. Se la differenza tra il tempo e tale variabile risulta maggiore del tempo definito per il calcolo del PID si entra nel calcolo dell'uscita e nel riaggiornamento di tale variabile. Si tratta quindi di un sistema soft-real-time in quanto si ha il ciclo real-time non vincolato perfettamente al tempo. Infatti considerando che il ciclo possa essere di circa 7-8ms al più il pid verrà eseguito al tempo predisposto di computazione più il tempo di ciclo. L'approssimazione è fatta con il metodo di Eulero in avanti. Variando il Kp si varia la risposta e prontezza del sistema che potrebbe anche diventare instabile se troppo grande oppure vedere un tempo di assestamento molto lungo e con diverse oscillazioni. Variando il Ki si ottiene la reiezione continua dell'errore, ovvero se il valore di riferimento e di uscita differiscono di una quantità costante, la parte integrale va aumentando e contribuisce maggiormente con il trascorrere del tempo fino a che l'errore e quindi l'integrale cambia di segno. Per ottenere un effetto di anti windup, ed evitare che in taratura l'integrale vada ad assumere valori mastodontici che influirebbero negativamente su un cambio di riferimento, è stata modificata la libreria inserendo un massimo e

minimo per questo termine contribuendo con un piccolo mattoncino alla comunità Open source. Per quanto riguarda il termine derivativo Kd, proporzionale alla derivata dell'errore, è stata modificata la libreria per consentire o non consentire la disattivazione dell'inner loop. Infatti tale termine nel PID classico determina una maggiore prontezza del sistema ma porta facilmente all'instabilità, con l'innerloop in realtà va a diminuire la prontezza del sistema ma possono essere scelti valori di Kp più alti che compensano il classico ruolo del Kd.

### 3.1.1 - PID timone

Il PID sul timone è stato fatto inserendo come ingresso il valore filtrato della bussola $\theta b$, come uscita il valore da dare in scrittura alla libreria servo in gradi che controllerà il timone (nel codice Autotimone), come riferimento il valore di $\theta d$ desiderato che proviene dall'arcotangente tra il punto di posizione rilevato dal GPS a bordo del veicolo e il punto di arrivo. I parametri sono quelli menzionati precedentemente. Ovviamente a seconda di come è stato collegato il servo al timone sarà necessario utilizzare 90+Autotimone oppure 90-Autotimone e con 90 si intendono 90° ovvero il punto medio del timone. Nel codice è stata anche inserita un'altra variabile di offset, considerando che la barchetta è artigianale e che non sempre i 90° indicano la perfetta posizione del timone in modo che la barchetta non abbia una certa deriva.

Per ottenere delle virate sempre verso l'angolo più corto è stata fatta una correzione dei parametri di ingresso come segue:

```
TetaE = TetaD - TetaB;            //calcolo l'errore

  if ((TetaB < 0) && (TetaD > 0) && (TetaE > 180)) //condizione affinché la barca vada verso l'angolo più corto

  {

   InputTimone = (360 + TetaB);     // Ingresso al PID con un valore maggiore di 180° oppure minore di -180°
```

```
  }

   if((TetaB > 0) && (TetaD < 0) && (TetaE < -180))

  {

    InputTimone = (-360 + TetaB);

  }

   if (((TetaB < 0) && (TetaD < 0)) || ((TetaB > 0) && (TetaD > 0)) || ((TetaB < 0)
&& (TetaD > 0) && (TetaE < 180)) || ((TetaB > 0) && (TetaD < 0) && (TetaE > -
180)))

   {

     InputTimone = TetaB;  // l'ingresso per tutti i casi in cui l'angolo non attraversa i
-180+180 o viceversa

   }
```

### 3.1.2 - PID motore

Il PID per il motore è stato fatto inserendo come ingresso il valore di velocità rilevato dal GPS, come uscita il valore compreso da 0° a 90° da dare in scrittura alla libreria servo che controllerà il segnale da dare all'azionamento. Infatti la libreria prevede di utilizzare proprio un riferimento analogo sia per il controllo dei motori sia per il controllo dei Servi. Il riferimento sarà dato da PC, e tale valore sarà comprensivo di una variabile di offset in quanto l'azionamento da barca non prevede la rotazione in senso opposto e per questo è necessario che parta da 90.

## 3.2 - Analisi log

L'analisi dei log è stata fatta usando Matlab. Le prove preliminari sono state svolte dando un riferimento costante al motore per la taratura del timone e cercando di tarare i parametri del PID timone.

### 3.2.1 - Errore di orientazione

Il primo grafico che mostra l'errore di orientazione in [°] dal riferimento in azzurro al valore stimato di $\theta b$ è in Figura 27 dove è possibile notare un errore costante.

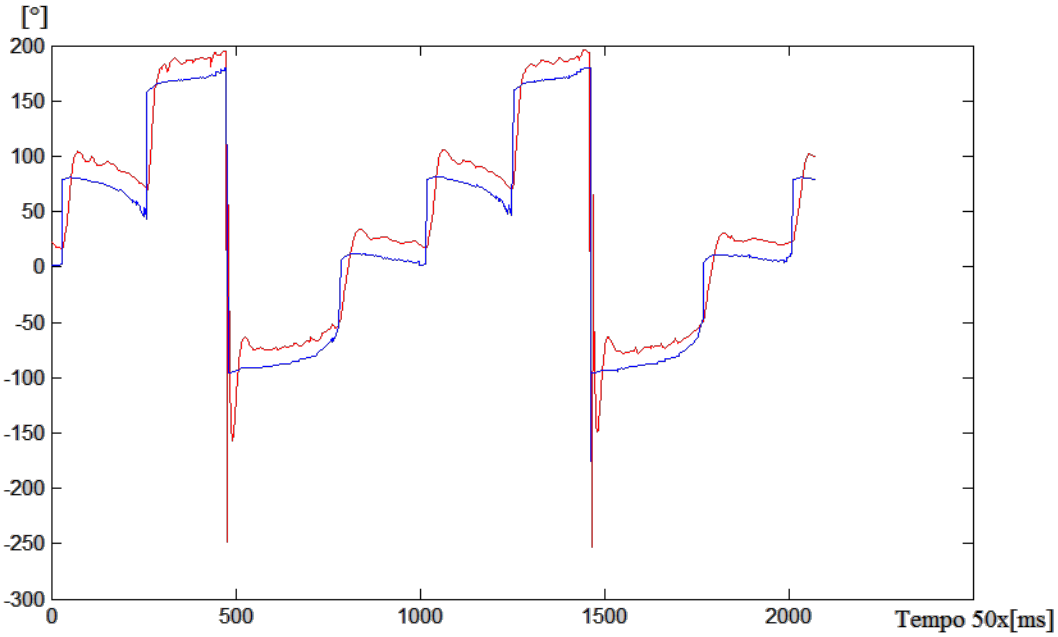


**Figura 28 Errore di orientazione esperimento A**

L'asse x dei grafici rappresenta il tempo, ma dato che la richiesta delle variabili da monitorare è di 50ms ogni punto rappresenta la misura ad intervalli di 50ms, per un totale di durata di questo primo esperimento di 100s circa. Tale errore genera un riferimento che viene ricalcolato ad ogni ciclo del programma e se la rotta reale fosse perfettamente quella desiderata si otterrebbe un segnale di riferimento a gradini con quattro livelli corrispondenti agli angoli di rotta desiderati. In fase di taratura è stato necessario includere un anti windup dato che è stato necessario utilizzare anche il parametro integrale per eliminare proprio l'errore costante.

Nel secondo esperimento sono stati aumentati i valori del termine integrale ed è stato annullato l'errore costante, è stato inoltre inserito un termine derivativo con innerloop che diminuiva la prontezza del sistema, ed anche se ad occhio nudo la barchetta percorreva la traiettoria abbastanza bene dai grafici ottenuti dal log si può osservare una certa oscillazione prima dell'assestamento sul riferimento. Inoltre nella Figura 28 si può notare come il riferimento in ingresso cambi e vada anche oltre i +180° e sotto i -180° in accordo con la stesura del programma per ottenere sempre una virata dalla parte più conveniente  quando il riferimento è prossimo ai ±180°.

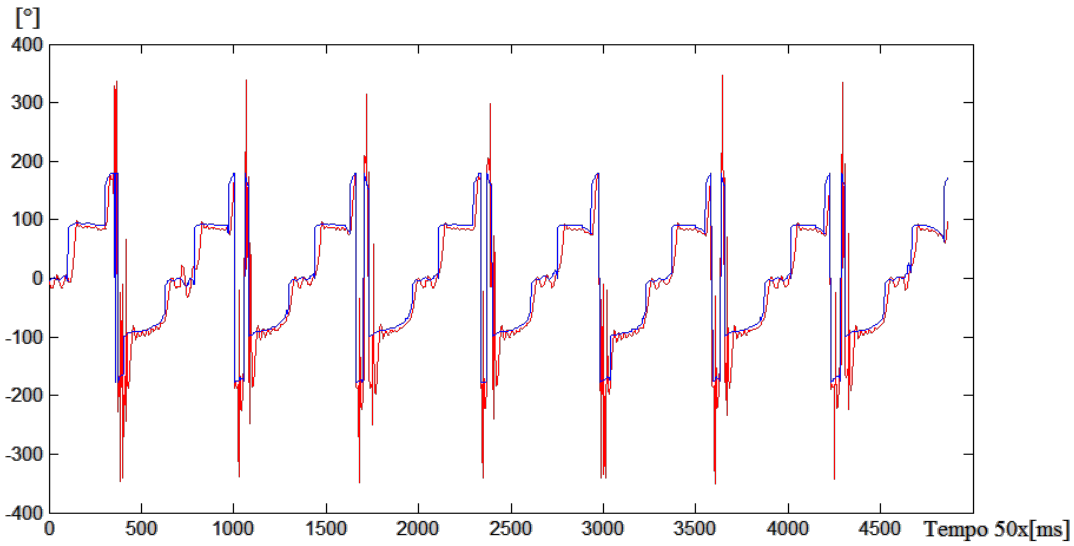


**Figura 29 Errore di orientazione esperimento B**

In questo esperimento è stata fatta una più lunga analisi del comportamento per verificarne anche la ripetibilità che sembra essere molto buona. Infatti permetterebbe di ottenere una taratura anche più fine dei parametri del PID che in caso contrario non sarebbe possibile.

### 3.2.2 - Errore sulla mappa

Un tipo di grafico interessante è quello che mostra i punti seguiti dal GPS sulla latitudine e longitudine in gradi decimali. I punti possono essere posti anche molto vicini tra loro dato che il GPS scelto riesce ad ottenere una discreta risoluzione in condizioni di buona visibilità dei satelliti che possono arrivare a 13. Il minimo quadrato effettuato dalla barchetta dando i punti di riferimento è stato di circa 2 metri

di lato. Il riferimento in azzurro presente nelle figure non è la vera e propria traiettoria che la barchetta dovrebbe seguire.

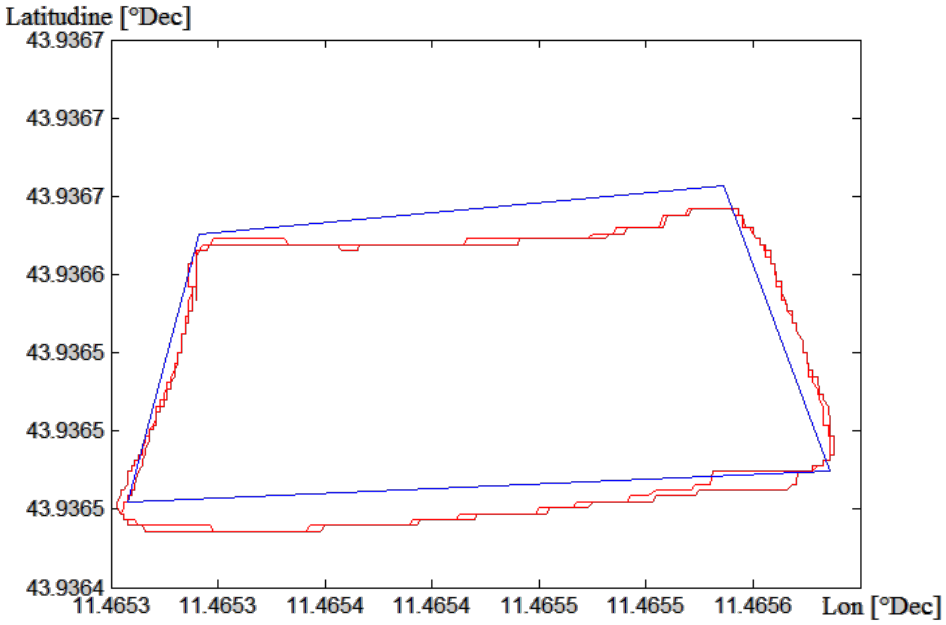


**Figura 30 Errore di traiettoria esperimento A**

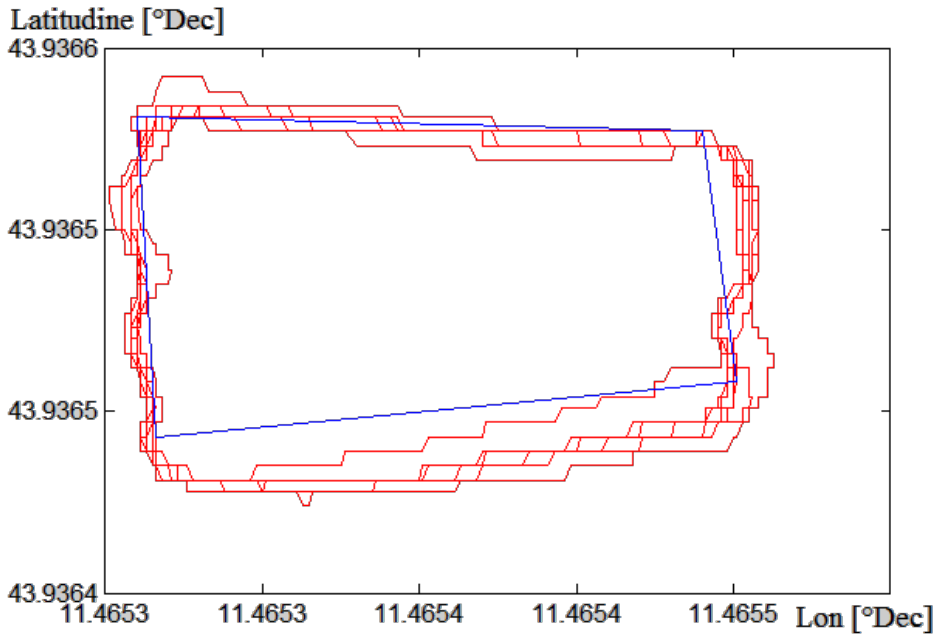


**Figura 31 Errore di traiettoria esperimento B**

Infatti per snellire il ciclo ed ottenere delle stime dello stato più affidabili si è scelto di non generare una traiettoria, ma di andare verso i vertici dei punti ad una velocità

costante. Come è possibile vedere dalla Figura 30 in accordo con la Figura 28 intorno al riferimento pari all'angolo di 0° sono presenti delle oscillazioni che sono ben visibili dai dati del log ma che ad occhio nudo sono risultate solo leggermente percettibili in fase di test.

### 3.2.3 - L'errore di velocità

Entrambi i test sul timone sono stati eseguiti con un riferimento di velocità pari a 5km/h ed in riferimento all'esperimento A è possibile vedere in corrispondenza dei cambi di riferimento un aumento dell'errore di velocità dovuto ai cambi di direzione.

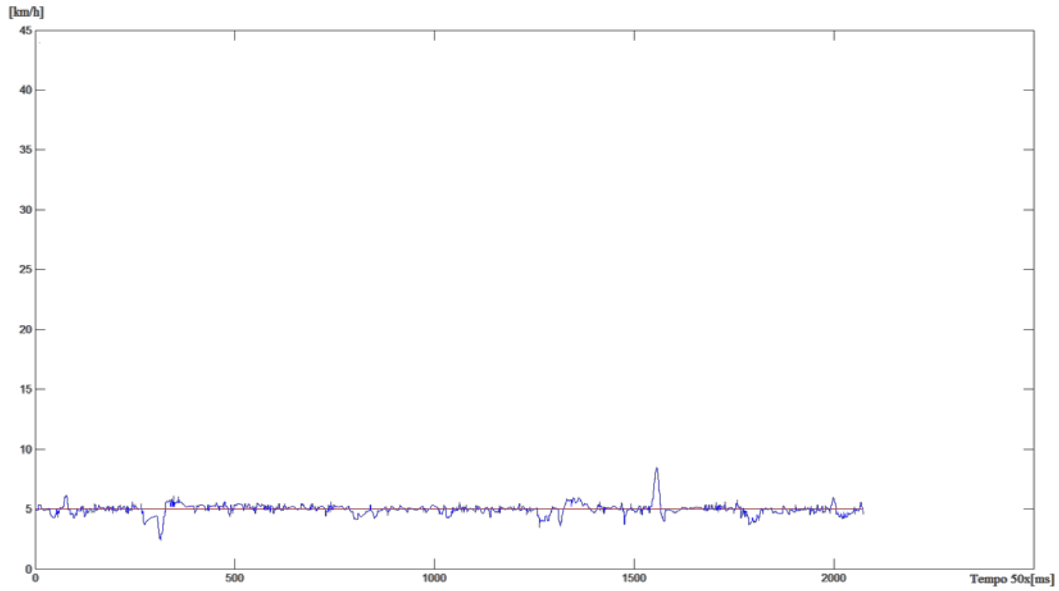


**Figura 32 Errore di velocità**

La taratura dei PID di sistemi non lineari, come nel caso di eliche e timoni, porta a dover fare una tabella di cambio dei valori dei guadagni in modo da ottenere valori di taratura differenti corrispondenti ai vari riferimenti in ingresso. Infatti con velocità di guida diverse i valori dei parametri trovati differiscono di molto.

# Conclusioni e sviluppi

Con questo lavoro sono state affrontate molte delle tematiche più importanti per la realizzazione di un veicolo autonomo. È stato analizzato il veicolo nelle parti che lo compongono e sulle scelte di tali parti è stato minimizzato il costo, ottenendo l'obiettivo prefissato. La stima dello stato ha dato ottimi risultati con l'applicazione del filtro di Kalman e sono state comprese e migliorate alcune delle librerie open souce adottate nel progetto ricondividendole a nostra volta. Evitando la programmazione tramite Matlab di Arduino è stato ottenuto un tempo di ciclo di 8 ms. L'interfaccia utente è stata di estrema utilità per comprendere il ruolo dei parametri di taratura e cambiare online i valori. La risposta del sistema, anche se artigianale e con non poche preoccupazioni per l'interferenza dei fattori ambientali, è risultata estremamente ripetibile e consentirà in futuro una più precisa taratura dei parametri, anche se attualmente è possibile ritenersi più che soddisfatti essendo riusciti ad ottenere un errore di rotta di circa lo 0,5% (2° su 360°) ed escludendo i cambi di direzione di circa il 5% sulla velocità.

Gli sviluppi futuri sono limitati solo dall'immaginazione. Avendo a disposizione beccheggio e rollio verrà sicuramente fatta una prova per la realizzazione di un aliscafo dinamico realizzato con alette orizzontali stampate in 3D e attuate per mezzo di servi digitali, più precisi e rapidi di quelli analogici. Modelli di questo tipo potrebbero essere usate per l'invio di piccoli pacchetti tra le città costiere in analogia a quanto vorrebbe fare Amazon con i quadricotteri creando una rete dove ogni mezzo conosce la posizione dell'altro e riuscendo anche a monitorare ben oltre i 3km di distanza sfruttando lo ZigBee. Inoltre è stato verificato che Arduino ha delle grosse potenzialità e non spaventerebbe neanche l'implementazione di controlli più onerosi e complicati. Per la stima dello stato può essere integrato un ulteriore filtro di Kalman per la stima della velocità, e posizione ed anche integrare la stima della rotta data dal GPS.

# BIBLIOGRAFIA

[1] Robbe. www.robbe.de. [Online]. http://www.robbe.de/roxxy-bl-outrunner-3548-06.html

[2] www.ebay.it. [Online]. http://www.ebay.it/itm/Mystery-RC-100A-Forward-Backward-Brushless-ESC-W-Water-Cooling-for-boat-V2-1-/231056550729?pt=UK_ToysGames_RadioControlled_JN&hash=item35cc0b2b49

[3] Arduino. (2008) http://arduino.cc/.

[4] G. Bernardo, *Easy Bee*. Robotitaly, 2011.

[5] [Online]. https://sites.google.com/site/websitedelguru/home/elettronica/il-protocollo-i2c

[6] Arduino. [Online]. http://arduino.cc/en/reference/wire#.UwSC4vl5Mfg

[7] A. Devices, *Digital accelerometer ADXL345*. 2012.

[8] H. HMC5883L, *3-Axis Digital Compass IC HMC5883L*. Plymouth, 2011.

[9] STMicroelettronics, *L3G4200D Mems motion sensor*. 2010.

[10] MEDIATEK -3329 Datasheet.

[11] N. e. altri. QTcreator. [Online]. http://qt-project.org/

[12] Declinazione magnetica. [Online]. http://en.wikipedia.org/wiki/Magnetic_declination

[13] Declinazione magnetica su mappa. [Online]. http://magnetic-declination.com/

[14] L. Chisci, *Appunti del corso "Stima e identificazione"*. Firenze, 2013.

]

[15 Hobbyking, *www.hobbyking.com.*

]

[16 (2010) XBee®/XBee-PRO® ZB RF Modules.

]

[17 Ebay. [Online]. http://www.ebay.it/itm/Mystery-RC-100A-Forward-Backward-

] Brushless-ESC-W-Water-Cooling-for-boat-V2-1-

/231056550729?pt=UK_ToysGames_RadioControlled_JN&hash=item35cc0b2

b49

# Allegato codice Arduino

```
#include <Servo.h>
#include <Wire.h>
#include <HMC5883L.h>
#include <GPS_MTK.h>
#include <PID_v1.h>
#include <math.h>
#include <ADXL345.h>
#include <L3G4200D.h>
#include <Kalman.h>
#include <Barchetta.h>

int Pilota = 0;            //variabile per pilota automatico o no
int Torna = 0;             //variabile che va a 1 se perde segnale
int Mammone = 0;           //variabile per tornare alla home se
perde segnale
int PidMot = 1;            //variabile per attivare il PID motore
int PidMotK = 0;           //variabile per attivare il pid con
kalman
int Autoreset = 0;         //variabile per resettare
int IL = -1;               //variabile per attivare l'innerloop

//Calcolo del tempo di ciclo
float Time;
int t1;
int t2;
uint32_t TimerSend = millis(); //per monitorare il ciclo

//Parametri filtro di kalmam
Kalman kalmanX;   // istanza per filtro di kalman su beccheggio
Kalman kalmanY;   // istanza per filtro di kalman su rollio
Kalman kalmanZ;   // istanza per filtro di kalman su bussola
Kalman kalmanVel; //istanza per filtro di kalman su velocità
double kalAngleX, kalAngleY, kalAngleZ, VelGPSK, accYangle,
accXangle;
//tempi per il calcolo della derivata per kalman
uint32_t timerP;
uint32_t timerR;
uint32_t timerZ;
uint32_t timerV;

//Accellerometro
ADXL345 acc;                    //inizializzo l'accellerometro
double pitch, roll, Xg, Yg, Zg;
const float alpha = 0.5;        //Filtro passa basso bebug
double fXg = 0;
double fYg = 0;
double fZg = 0;
double fYgNC, AccK;

//Bussola
HMC5883L compass;            //inizializza la bussola

//Gyroscopio
L3G4200D gyro;               //inizializzo il gyro
double gyroX, gyroY, gyroZ;  //variabili di vel angolare
```

```
//Parametri Servo
Servo Timone;                //inizializza il timone servo
Servo Motore;                //inizializza il motore servo
int leggipwm = 1;            //debug per leggere servo
float TonTimone, TonMotore, TimoneRicevente, MotoreRicevente;

//PIN USATI
int TimonePin = 10;          //pin servo timone
int MotorePin = 11;          //pin servo motore
int TimoneRiceventePin = 8;  //pin ricevente timone
int MotoreRiceventePin = 9;  //pin ricevente motore
int BatteriaPin = 4;         //pin lettura batteria
int AUTOTIMONE = 40;         //pin autopilota timone
int AUTOMOTORE = 41;         //pin autopilota motore
int MANUALTIMONE = 42;       //pin pilota manuale timone
int MANUALMOTORE = 43;       //pin pilota manuale timone

//Coorinate punti da seguire
double LatA;
double LonA;
double LatB;
double LonB;
double LatC;
double LonC;
double LatD;
double LonD;
double LatH;
double LonH;
//ritardi sui punti
float DelayA = 0;
float DelayB = 0;
float DelayC = 0;
float DelayD = 0;
float TimerPoint = 0;

//Variabili in ricezione
float LatQT, LonQT;
int Scelta;
int Send;

//Variabili Per inseguire i punti
float RifLat;                //riferimento di latitudine e
longitudine
float RifLon;                //riferimento di latitudine e
longitudine
double DifLat, DifLon;       //differenza di latitudine e
longitudine per trovare il TetaD con atan2
double DifLatASS, DifLonASS; //valore assoluto della differenza
float RaggioGPS = 0.000020;  // raggio entro il quale la barchetta
passa alle coordinate successive
float Cicla = 5;             //quanti volte deve seguire i punti
float CiclaQT;               //varibile per stabilire cicli e
settare altri paramentri

//variabili per lettura stato batteria
float Batteria;

// Variabili GPS
```

```
float Lat;                    // Variabili GPS
float Lon;                    // Variabili GPS
int Sat;                      // numero satelliti
double VelGPS;                // Velocità rilevata dal gps

//variabili per il calcolo del raggio
double TetaDrad;         //teta desiderato in radianti
double TetaD;            //teta desiderato
float TetaB;             //teta Barca o busssola qualdirsivoglia
double TetaE;            //Angolo di errore
double TetaK;            //Angolo per kalman
float Declinazione = 2;  //declinazione magnetica
float TetaGPS = 0;       //rotta dal gps da integrare con TetaB

//Variabili per la bussola accettazione e tatatura
double Rmagnetico; //Modulo vettore magnetico
float XXX;         //accettazione con sfere tarata
float YYY;
float ZZZ;
float xxx;         //taratura e limite di misura
float yyy;
float zzz;
int sferamin = 420; //raggio interno dello sferoide
int sferamax = 585; //raggio esterno dello sferoide
//per calcolarli una volta sola
float cos_roll;
float sin_roll;
float cos_pitch;
float sin_pitch;
double CYYY;
double CXXX;

//Parametri parchetta
float DistTP = 0.03; //distanza timone pinne

float TimoneSegnale= 0;    //inizializzazione da guidata con
radiocomando
float MotoreSegnale = 0;   //inizializzazione da guidata con
radiocomando

//parametri PID
double InputTimone;  //ingresso al pid timone
double Autotimone;   //uscita del pid timone
float TimoneZero = 0; //lo zero del timone
// TetaD             //setpoint timone
float AntiTim = 10;
float AntiMot = 20;
double kpTim = 2;    //proporzionale timone
double kiTim = 0;    //integrale timone
double kdTim = 0;    //derivativo timone
double kpMot = 5;    //proporzionale motore
double kiMot = 0;    //integrale motore
double kdMot = 0;    //derivativo motore
//VelGPS             //ingresso al pid motore
double Automotore;   //uscita al pid motore
double AutomotoreSet;//uscita pid motore +90
double VelRif = 6;   //rifermino di velocità in Km/h
double Min = -45;    //Minimo timone
double Max = 45;     //Massimo timone
```

```
double MinM = 0;     //Min Motore
double MaxM = 90;    //Max motore
PID motorePID(&VelGPS, &Automotore, &VelRif,kpMot,kiMot,kdMot,
DIRECT,&AntiMot,&IL);
PID timonePID(&InputTimone, &Autotimone, &TetaD,kpTim,kiTim,kdTim,
DIRECT,&AntiTim,&IL);


void setup()
{
  pinMode(AUTORESET, OUTPUT);   //pin autoreset settato come uscita
  digitalWrite(AUTORESET,LOW);  //pin autoreset basso
  Serial.begin(115200);         //seiale debug USB
  Serial1.begin(38400);         // inizializza la seriale del GPS
  Serial2.begin(38400);         // inizializza seriale Xbee
  Serial3.begin(38400);

  pinMode(MANUALTIMONE, OUTPUT); //uscita per la scelta del
controllo
  pinMode(MANUALMOTORE, OUTPUT);
  pinMode(AUTOTIMONE, OUTPUT);
  pinMode(AUTOMOTORE, OUTPUT);


  pinMode(TimoneRiceventePin, INPUT);    //setta il pin ricevente
timone
  pinMode(MotoreRiceventePin, INPUT);    //setta il pin ricevente
Motore
  Timone.attach(TimonePin);              //servo timone
  Motore.attach(MotorePin);              //servo motore

  Motore.write(90);                      //sett il mezzo DA TESTARE
  Timone.write(90);                      // set servo to mid-point

  delay(1000);          //attendi la carica dei condensatori sui
sensori

  Wire.begin();         //per la I^2C

  compass = HMC5883L(); //istanza per la bussola

  setupHMC5883L();      //setup the HMC5883L

  gyro.enableDefault(); //gyroscopio setuppe

  acc.begin();          //accellerometro setuppe

  GPS.Init();           //inizializza il GPS

  BAR.Init();           //inizialiuzzo la comunicazione

//punti all'uni per il debug angoli terrazza
/*
LatA = 43.800460;
LonA = 11.245148;
LatB = 43.800491;
LonB = 11.245148;
LatC = 43.800441;
LonC = 11.245266;
```

```
LatD = 43.800403;
LonD = 11.245246;
LatH = 43.800472;
LonH = 11.245245;
*/
/*
//Punti a casa per dedug
LatA = 43.950955;//Parcheggio giudo
LonA = 11.399841;
LatB = 43.950773;//Davanti al cancello campo
LonB = 11.400552;
LatC = 43.951159;//Davanti ad ingresso orti
LonC = 11.400922;
LatD = 43.951621;//Parcheggio montagnole
LonD = 11.400831;
LatH = 43.951374;//Davanti ai cassonetti
LonH = 11.400595;
*/
//Lago di vicchio
LatA = 43.936445;
LonA = 11.465644;
LatB = 43.936275;
LonB = 11.465941;
LatC = 43.936105;
LonC = 11.465644;
LatD = 43.936275;
LonD = 11.465347;
LatH = 43.936445;
LonH = 11.465400;

//riferimento di latitudine e longitudine iniziale
RifLat = LatA;
RifLon = LonA;

motorePID.SetMode(AUTOMATIC);
timonePID.SetMode(AUTOMATIC);
delay(100); //per attendere una buona stima dei sensori

//settaggio iniziale filtro kalman
  acc.read(&Xg, &Yg, &Zg);
  accYangle = (atan2(Xg,Zg)*RAD_TO_DEG);
  accXangle = (atan2(Yg,Zg)*RAD_TO_DEG);
  kalmanX.setAngle(accXangle); //
  kalmanY.setAngle(accYangle);
  timerP = micros();
}

void loop()
{
  t1 = millis();  //calcolo del ciclo
  if (Pilota == 0)//Pilota manuale
  {
    digitalWrite(MANUALTIMONE,HIGH);
    digitalWrite(MANUALMOTORE,HIGH);
    digitalWrite(AUTOTIMONE,LOW);
    digitalWrite(AUTOMOTORE,LOW);

    /* Varie prove fatte per la lettura del duty della ricevente
     *
```

```
     *
    pinMode(TimonePin, OUTPUT);        //setta il pin per il servo
timone
    pinMode(MotorePin, OUTPUT);        //setta il pin per il servo
motore
    Timone.attach(TimonePin);          //servo timone
    Motore.attach(MotorePin);          //servo motore

    TimoneSegnale = digitalRead(TimoneRiceventePin);   // read the
input pin Ricevente
    digitalWrite(TimonePin, TimoneSegnale);    // setta il timone
pin come il timone segnale
    MotoreSegnale = digitalRead(MotoreRiceventePin);   // read the
input pin motore
    digitalWrite(MotorePin, MotoreSegnale);    // setta il timone
pin come il timone segnale

     if(leggipwm == 1)
    {
      TonTimone = pulseIn(TimoneRiceventePin, HIGH);
      TimoneRicevente = ((TonTimone - 1085) * 0.21077);
      Timone.write(TimoneRicevente);
      leggipwm = 0;
    }
     if(leggipwm == 0)
    {
      TonMotore = pulseIn(MotoreRiceventePin, HIGH);
      MotoreRicevente = ((TonMotore - 1085) * 0.21077);
      Motore.write(MotoreRicevente);
      leggipwm = 1;
    }
    */
  }
    //leggo il GPS
    //i casi sono associati a vari tipi di errori di lettura sulla
libreria
    //anche questa libreria è stata corretta e ricondivisa.
    switch(GPS.Read())
    {
      case 0:
    Lat = ((float)GPS.Lattitude/1000000);
    Lon = ((float)GPS.Longitude/1000000);
    VelGPSK = (GPS.Speed*0.036);//km/h
   // VelGPSK = (GPS.Speed)*0.01;//m/s
    Sat = ((int)GPS.Sats);
    TetaGPS = (GPS.Heading/100);
    break;
     case 1:
     Serial.print ("GPS no fix");//debug
     break;
     case 2:
     Serial.print ("GPS errorA");
     break;
     case 3:
     Serial.print ("GPS errorB");
     break;
     case 4:
     Serial.print ("GPS timeout");
     break;
```

```
    }

    //Leggo la batteria
    Batteria = analogRead(BatteriaPin)*0.0256858217;

   //leggo la bussola
    MagnetometerScaled scaled = compass.ReadScaledAxis();
    //Aceettazione misura del campo magnetico intervalli e
correzione di calibrazione
    if((scaled.XAxis > -600) && (scaled.XAxis < 600))
    {
      xxx = scaled.XAxis + 19;
    }
    if((scaled.YAxis > -600) && (scaled.YAxis < 600))
    {
      yyy = scaled.YAxis + 99;
    }
    if((scaled.ZAxis > -600) && (scaled.ZAxis < 600))
    {
      zzz = scaled.ZAxis -30;
    }

//Accettazione di valori di misura con coppia di sferoidi
    Rmagnetico = sqrt((xxx*xxx) + (yyy*yyy) + (zzz*zzz));
    if((Rmagnetico > sferamin) && (Rmagnetico < sferamax) )
  {
    YYY = yyy;
    XXX = xxx;
    ZZZ = zzz;
  }

//leggo l'accellerometro
 acc.read(&Xg, &Yg, &Zg);
 Xg = Xg+0.05;
 Yg = Yg+0.03;

//leggo il gyroscopio
 gyro.read();
 gyroX = gyro.g.x;
 gyroY = gyro.g.y;
 gyroZ = gyro.g.z;

//Calcolo il Beccheggio e rollio con kazman
 //moltiplico per la sensitività i valori letti dal gyroscopio
 double gyroXrate = gyroX/100;
 double gyroYrate = -gyroY/114;
 double gyroZrate = -gyroZ/114;


 //fYg = (Yg - (((VelGPS*VelGPS) * tan(Autotimone)) /
DistTP)*cos_roll); //CORREZIONE DI ACCELLERAZIONE con centro di
istantanea rotazione
 accXangle = (atan2(Yg,Zg)*RAD_TO_DEG);
 accYangle = (atan2(Xg,Zg)*RAD_TO_DEG);
 kalAngleX = kalmanX.getAngle(accXangle, gyroXrate,
(double)(micros()-timerR)/1000000); // Calcolo del filtrozzo di
kalman
 timerR = micros();
```

```
 kalAngleY = kalmanY.getAngle(accYangle, gyroYrate,
(double)(micros()-timerP)/1000000); // calcolo del filtrozzo di
kalman
 timerP = micros();
 roll = kalAngleX*DEG_TO_RAD;
 pitch = kalAngleY*DEG_TO_RAD;
/*
    //filtro passa basso di prova per vedere se migliorava la stima
    fXg = Xg * alpha + (fXg * (1.0 - alpha));
    fYgNC = Yg * alpha + (fYg * (1.0 - alpha));
    fZg = Zg * alpha + (fZg * (1.0 - alpha));
    fYg = (fYgNC - ((VelGPS*VelGPS * tan(Autotimone)) / DistTP));
//CORREZIONE DI ACCELLERAZIONE con centro di istantanea rotazione
    //Roll  Pitch Con accellerometro senza kalman
    roll  = atan2(-fYg, fZg);
//radianti
    pitch = atan2(fXg, sqrt(fYg*fYg + fZg*fZg));
//radianti
*/
 //Calcolo del TetaB rispetto al nord sul sistema di riferimento
scelto
  cos_roll = cos(roll);
  sin_roll = sin(roll);
  cos_pitch = cos(pitch);
  sin_pitch = sin(pitch);
  CYYY = (YYY * cos_roll - (ZZZ * sin_roll));    //Proiezione della
componente y del vettore campo magnetico sul sistema di riferimento
immaginario "Piano terra"
  CXXX= (XXX * cos_pitch - (ZZZ * sin_pitch));   //Proiezione della
componente x del vettore campo magnetico sul sistema di riferimento
immaginario "Piano terra"
  TetaK = (atan2(CYYY,CXXX)* RAD_TO_DEG);        // Tiro fuori
l'angolo dal nord
  AccK = 9.81*(((Zg-(cos_pitch*cos_roll)))-(Xg-
(sin_pitch*cos_roll)));
  kalAngleZ = kalmanZ.getAngle(TetaK, gyroZrate, (double)(micros()-
timerZ)/1000000); // calcolo del filtrozzo di kalman
  timerZ = micros();
  VelGPS = (kalmanVel.getAngle(VelGPSK, AccK, (double)(micros()-
timerV)/1000000))*0.036; // calcolo del filtrozzo di kalman
  timerV = micros();
  TetaB = kalAngleZ+Declinazione;


//DEBUG
/*
    Serial.print("  Lat:");
    Serial.print(Lat,6);
    Serial.print("  TetaD:");
    Serial.print(TetaD);
    Serial.print("  TetaB:");
    Serial.print(TetaB);
    Serial.print("  BussolaGPS");
    Serial.print((float)GPS.Heading/100.0);
    Serial.print("  Autotimone:");
    Serial.print(90-Autotimone);
    Serial.print("   AutoPID");
    Serial.print(Autotimone);
    Serial.print("  TetaE");
```

```
Serial.print(TetaE);
Serial.print("       VelGPS:");
Serial.print(VelGPS);

Serial.print("  Automotore:");
Serial.print(90+Automotore);

Serial.print("Cicla:");
Serial.print(Cicla);
Serial.print("  KpTim:");
Serial.print(kpTim);
Serial.print("  Vbatt:");
Serial.print(Batteria);

Serial.print("  DifLatASS:");
Serial.print(DifLatASS,6);
Serial.print("  RifLat:");
Serial.print(RifLat,6);
Serial.print("  Riflon:");
Serial.print(RifLon,6);
Serial.print("  Tempo:");

Serial.print("   Diflat");
Serial.print(DifLat,6);
Serial.print("   Diflon");
Serial.print(DifLon,6);
Serial.print("  Sat:");
Serial.println(Sat);

Serial.print("  Autotimone:");
Serial.print(90-Autotimone);
Serial.print("   AutoPID");
Serial.print(Autotimone);

Serial.print("  TetaB:");
Serial.println(TetaB);

Serial2.print("  TetaB:");
Serial2.println(TetaB);
Serial.print("  CXXX:");
Serial.print(CXXX);
Serial.print("  CYYY:");
Serial.print(CYYY);
Serial.print("   Pich:");
Serial.print(pitch*RAD_TO_DEG);
Serial.print("   Roll:");
Serial.print(roll*RAD_TO_DEG);
Serial.print("  XXX:");
Serial.print(XXX);
Serial.print("  YYY:");
Serial.print(YYY);
Serial.print("  ZZZ:");
Serial.println(ZZZ);

Serial.print("  XXX:");
Serial.print(accXangle);
Serial.print("  YYY:");
Serial.print(accYangle);
Serial.print("   Pich:");
```

```
    Serial.print(kalAngleY);
    Serial.print("   Roll:");
    Serial.print(kalAngleX);

   // Serial.print("  TetaB:");
   // Serial.print(TetaB);
   // Serial.print("  XXX:");
   // Serial.print(accXangle);
   // Serial.print("  YYY:");
   // Serial.print(accYangle);
   // Serial.print("   Pich:");
   // Serial.print(kalAngleY);

   Serial.print("  GyroX");
   Serial.print(gyroXrate);
   Serial.print(" Zacc:");
   Serial.print(Zg);
   Serial.print(" Yacc:");
   Serial.print(Yg);
   Serial.print(" AngleZY:");
   Serial.print(accXangle);
   Serial.print("   Roll:");
   Serial.print(kalAngleX);

   Serial.print("         GyroY");
   Serial.print(gyroYrate);
   Serial.print(" Zacc:");
   Serial.print(Zg);
   Serial.print(" Yacc:");
   Serial.print(Xg);
   Serial.print(" AngleZX:");
   Serial.print(accYangle);
   Serial.print("   Roll:");
   Serial.print(kalAngleY);


  //Serial.print(" Zg:");
  //Serial.print(Zg);
  //Serial.print(" Xg:");
  //Serial.print(Xg);

  //Serial2.print("  VelGPSkal");
  //Serial2.println(VelGPS);


  //t2 = millis();
  //Serial.print("   Tc:");
  //Serial.println(t2-t1);
  */

//Aquisizione dati da xbee
  switch(BAR.Read())
    {
      case 0:
    LatQT = ((float)BAR.LatQT);
    LonQT = ((float)BAR.LonQT);
    RaggioGPS = ((float)BAR.RaggioGPS);
    VelRif = (BAR.Vmax);
    CiclaQT = (BAR.Cicla);
```

```
    Scelta = ((int)BAR.Scelta);
    TimerPoint = (BAR.Send*1000);
    kpTim = BAR.KpT;//
    kiTim = BAR.KiT;//
    kdTim = BAR.KdT; //
    kpMot = BAR.KpM;//
    kiMot = BAR.KiM;//
    kdMot = BAR.KdM;//
    Torna = BAR.Torna;
    break;
     case 1:
     Serial2.print ("no fix");
     break;
     case 2:
     Serial2.print ("errorA");
     break;
     case 3:
     Serial2.print ("errorB");
     break;
     case 4:
     Serial2.print ("timeout");
     break;
    }


//Settaggio variabili da aquisizione dati
if (Scelta == 0)
{
  LatH = Lat;
  LonH = Lon;
  Scelta = 10;
}
 if (Scelta == 1)
{
  LatA = Lat;
  LonA = Lon;
  RifLat = LatA;
  RifLon = LonA;
  DelayA = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 2)
{
  LatB = Lat;
  LonB = Lon;
  DelayB = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 3)
{
  LatC = Lat;
  LonC = Lon;
  DelayC = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 4)
{
  LatD = Lat;
```

```
  LonD = Lon;
  DelayD = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 5)
{
  LatH = LatQT;
  LonH = LonQT;
  Scelta = 10;
}
 if (Scelta == 6)
{
  LatA = LatQT;
  LonA = LonQT;
  RifLat = LatA;
  RifLon = LonA;
  DelayA = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 7)
{
  LatB= LatQT;
  LonB = LonQT;
  DelayB = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 8)
{
  LatC = LatQT;
  LonC = LonQT;
  DelayC = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 9)
{
  LatD = LatQT;
  LonD = LonQT;
  DelayD = TimerPoint;
  Scelta = 10;
}
 if (Scelta == 11)
{
  Pilota = 0;
  Scelta = 10;
}
 if (Scelta == 12)
{
  Pilota = 1;
  Scelta = 10;
}
if (Scelta == 13)
{
  Cicla = CiclaQT;
}
if (Scelta == 14)
{
  PidMot = 1;
}
if (Scelta == 15)
```

```
{
  PidMot =0;
}
if (Scelta == 16)
{
  PidMotK =1;
}
if (Scelta == 17)
{
  PidMotK = 0;
}
if (Scelta == 18)
{
  digitalWrite(AUTORESET,HIGH);
}
if (Scelta == 19)
{
  Mammone = 0;
}
if (Scelta == 20)
{
  Mammone = 1;
}
if (Scelta == 21)
{
  TimoneZero = CiclaQT;
}
 if (Scelta == 22)
{
  Declinazione = CiclaQT;
}
if (Scelta == 23)
{
  AntiMot = CiclaQT;
}
if (Scelta == 24)
{
  AntiTim = CiclaQT;
}
 if (Scelta == 25)
{
  IL = -1;
}
if (Scelta == 26)
{
  IL = 1;
}

if (PidMotK == 0)
{
VelGPS = VelGPSK;
}

//trasformazione delle variabili in float per inviarle
float inputT = InputTimone;
float VelGPSS = VelGPS;
float TetaDS = TetaD;
float kpTimS = kpTim;
float kiTimS = kiTim;
```

```
 float kdTimS = kdTim;
 float kpMotS = kpMot;
 float kiMotS = kiMot;
 float kdMotS = kdMot;
 float CiclaS = Cicla;
 float RaggioGPSS = RaggioGPS;
 float Satt = Sat;
 float PICH = kalAngleX;
 float ROLL = kalAngleY;
 float VELGPS = VelGPSK;
 float AutotimoneS = Autotimone;
 float AutomotoreS = Automotore;

 //invio delle variabili
 BAR.SendDati(&Lat, &Lon, &RifLat, &RifLon, &TetaGPS, &TetaDS,
&CiclaS, &AutomotoreS, &inputT, &AutotimoneS, &Satt, &PICH, &Time,
&VELGPS, &ROLL, &Batteria, &Sat);

  if (Pilota == 1)//variabile autopilota
  {
    digitalWrite(MANUALTIMONE,LOW);  //uscite per la scelta del
controllo per il pilota automatico
    digitalWrite(MANUALMOTORE,LOW);  //uscite per la scelta del
controllo per il pilota automatico
    digitalWrite(AUTOTIMONE,HIGH);   //uscite per la scelta del
controllo per il pilota automatico
    digitalWrite(AUTOMOTORE,HIGH);   //uscite per la scelta del
controllo per il pilota automatico


    DifLat = RifLat - Lat;         //calcolo le distanze dal punto
per trovare l'angolo
    DifLon = RifLon - Lon;         //calcolo le distanze dal punto
per trovare l'angolo
    DifLonASS = abs(DifLon);       //Valore assoluto per la distanza
dal riferimento
    DifLatASS = abs(DifLat);       //Valore assoluto per la distanza
dal riferimento

   // strutturina per i punti devo passare da A a B Poi a C e D e se
voglio tornare a casa invece di cilare.
       if ((Cicla == -1) && (DifLonASS <= RaggioGPS) && (DifLatASS
<= RaggioGPS))
    {
      Pilota = 0;//arrivato alla home passa in pilota manuale
    }

    if ((DifLonASS <= RaggioGPS) && (DifLatASS <= RaggioGPS))
    {
      if ((RifLat == LatD) && (RifLon == LonD) && (Cicla >= 1))
      {
        RifLat = LatA;
        RifLon = LonA;
        Cicla--;
        if (DelayD != 0)
        {
        Timone.write(90);
        Motore.write(0);
        delay(DelayD);
```

```
        }
      }
      else if ((RifLat == LatD) && (RifLon == LonD) && (Cicla == 0))
      {
        RifLat = LatH;
        RifLon = LonH;
        Cicla--;
        if (DelayD != 0)
        {
        Timone.write(90);
        Motore.write(0);
        delay(DelayD);
        }
    }
      else if ((RifLat == LatC) && (RifLon == LonC))
      {
        RifLat = LatD;
        RifLon = LonD;
        if (DelayC != 0)
        {
        Timone.write(90);
        Motore.write(0);
        delay(DelayC);
        }
      }
      else if ((RifLat == LatB) && (RifLon == LonB))
      {
        RifLat = LatC;
        RifLon = LonC;
        if (DelayB != 0)
        {
        Timone.write(90);
        Motore.write(0);
        delay(DelayB);
        }
      }
      else if ((RifLat == LatA) && (RifLon == LonA))
      {
        RifLat = LatB;
        RifLon = LonB;
        if (DelayA != 0)
        {
        Timone.write(90);
        Motore.write(0);
        delay(DelayA);
        }
      }

    }

    TetaDrad = (atan2(DifLon, DifLat));
    TetaD = TetaDrad * RAD_TO_DEG;       // Passaggio da radianti a
gradi
 // TetaB = in fondo                    // bussola in TetaB
    TetaE = TetaD - TetaB;               //calcolo l'errore


    if ((TetaB < 0) && (TetaD > 0) && (TetaE > 180)) //condizione
affinche la barca vada verso l'angolo più corto
```

```
   {
     InputTimone = (360 + TetaB);                    // do in
ingresso un valore maggiore di 180° oppure minore di -180° in modo
che calcoli il pid in modo buono
   }
    if((TetaB > 0) && (TetaD < 0) && (TetaE < -180))
   {
     InputTimone = (-360 + TetaB);
   }
    if (((TetaB < 0) && (TetaD < 0)) || ((TetaB > 0) && (TetaD > 0))
|| ((TetaB < 0) && (TetaD > 0) && (TetaE < 180)) || ((TetaB > 0) &&
(TetaD < 0) && (TetaE > -180)))
    {
      InputTimone = TetaB;                           // l'ingresso
per tutti i casi in cui l'angolo non attraversa i -180+180 o
viceversa
    }

    //settaggio PID
    timonePID.SetTunings(kpTim, kiTim, kdTim);
    motorePID.SetTunings(kpMot, kiMot, kdMot);
    timonePID.SetSampleTime(50);
    motorePID.SetSampleTime(50);
    timonePID.SetOutputLimits(Min, Max);
    motorePID.SetOutputLimits(MinM, MaxM);
    timonePID.Compute();
    motorePID.Compute();

    Timone.write(90-Autotimone+TimoneZero); //scrivo il valore di
uscita del pid per il servo che aziona il timone.

    if (PidMot == 1 )
    {
    Motore.write(90+Automotore); //scrivo il valore di uscita del
pid per il servo che aziona il motore.
    }
    if (PidMot == 0)
    {
      Motore.write(90+VelRif);
    }
}

 //ritorna alla home se non ricevi nulla..
  if ((Torna == 1) && (Mammone = 0))
  {
    RifLat = LatH;
    RifLon = LonH;
    Cicla = -1;
  }
    t2 = millis();
    Time = t2-t1;

 delay(Autoreset);//con watchdog da implementare con modifica
compilatore.
}
void setupHMC5883L(){
  //Setup HMC5883L e vedere errori
  int error;
  error = compass.SetScale(1.3); //setta la scala.
```

```
  if(error != 0) Serial.println(compass.GetErrorText(error));
//guarda gli errori e stampali
  error = compass.SetMeasurementMode(Measurement_Continuous);
//setta la misura continua
  if(error != 0) Serial.println(compass.GetErrorText(error));
//guarda gli errori e stampali
}
```