QUADRICOTTERO

Obiettivi del progetto

- Analizzare i principi del volo verticale di un UAV quadricottero
- Analizzare le potenzialità di Arduino
- Utilizzare Arduino come strumento di logging e di controllo posizione rispetto al piano di volo
- Controllo a chiave simmetrica del piano di volo (controllo corruzione piano di volo)
- Visualizzazione della posizione corrente su PC

Fasi del progetto

- Corso teorico tramite video-lezioni remote
- Test individuale su corso teorico
- Esercizi individuali sull'utilizzo base di Arduino
- Montaggio di un drone in kit
- Collegamento dei vari shield Arduino e sistemazione di essi sul drone
- Progetto finale: realizzazione di un programma per Arduino che permetta di effettuare il logging, il controllo del piano di volo e l'invio della posizione corrente ad un PC

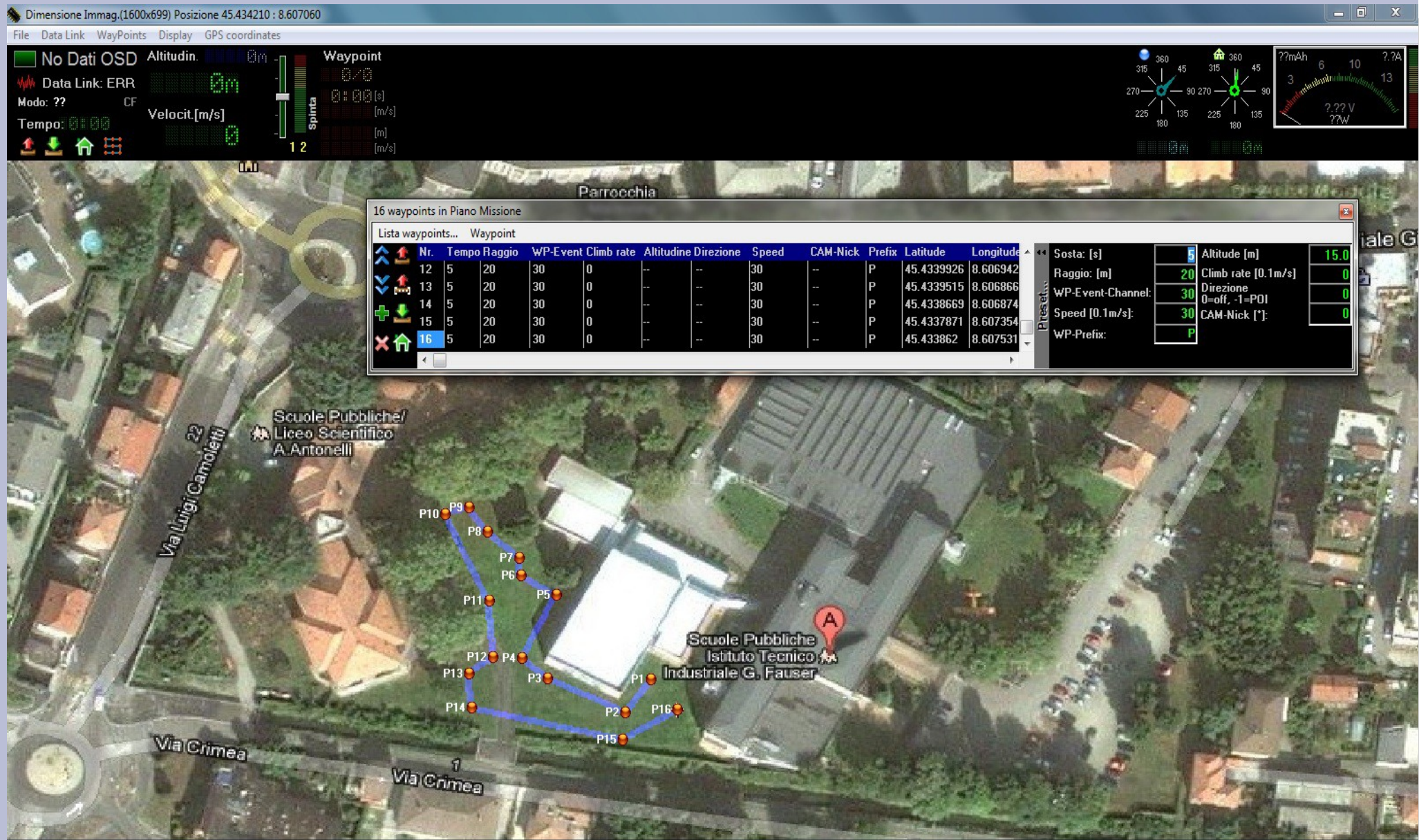
Oggetto della tesina - parte 1

- Creazione del piano di volo da rispettare tramite il software del costruttore del drone. Output dato in “pasto” a un programma creato da noi che genera un'hash MD5
- MD5: un tipo di hash. Dati in input una serie di byte si ottiene in output una stringa di lunghezza sempre uguale
- Arduino in seguito rigenera l'hash e la confronta per stabilire se il file del piano di volo è stato corrotto

Oggetto della tesina - parte 2

- Per controllare possibili corruzioni utilizzare la stessa chiave sia in fase di generazione del file che in fase di controllo (crittografia simmetrica) è sufficiente.
- Dato che il progetto è stato pensato anche per un utilizzo da parte della protezione civile, è stata realizzata una versione con chiave pubblica e chiave privata (crittografia assimetrica) per evitare manomissioni del file. Criptiamo quindi l'hash MD5 precedentemente generata, con la chiave privata

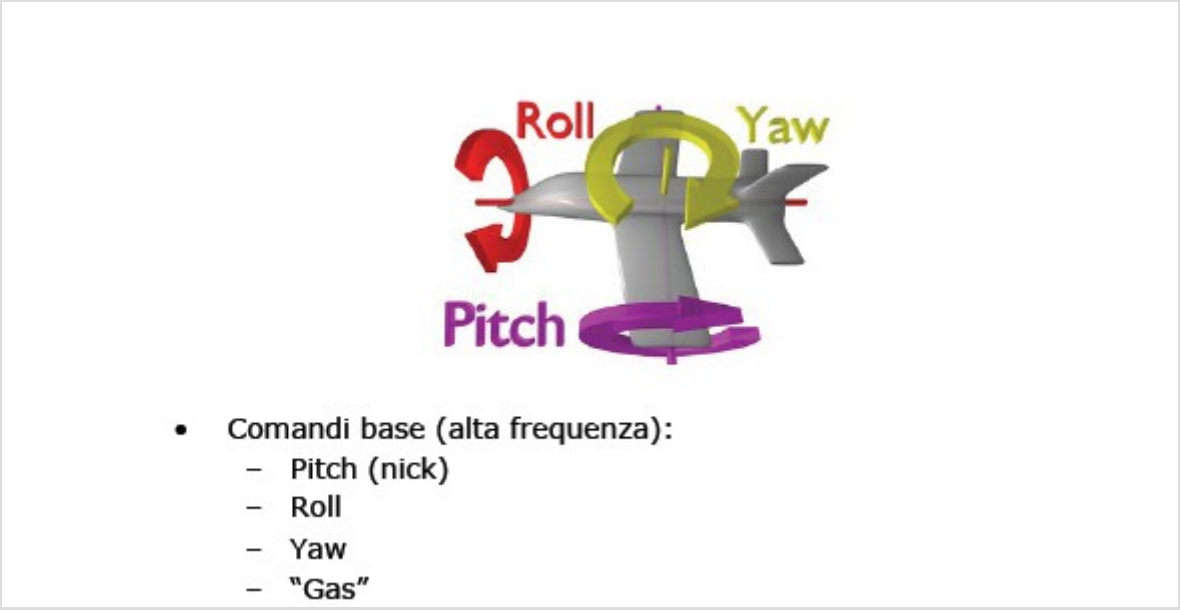
Oggetto della tesina - parte 3



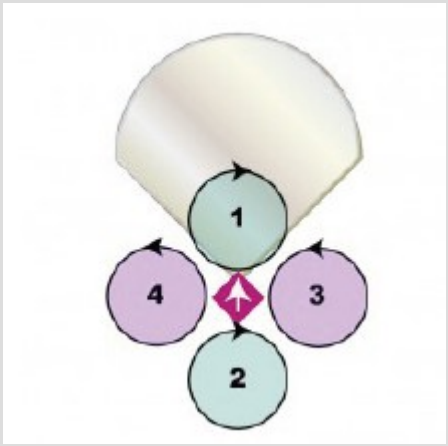
Principi per il volo - parte 1

- Volo e decollo VTOL (vertical take-off and landing)
- Volo automatico - Unmanned Aerial Vehicles (UAV)
- Eliche a tra loro controtanti
- Position hold (tramite GPS)
- Altitude hold
- Stabilizzazione tramite giroscopi e accelerometri

Principi per il volo - parte 2



Principio di volo



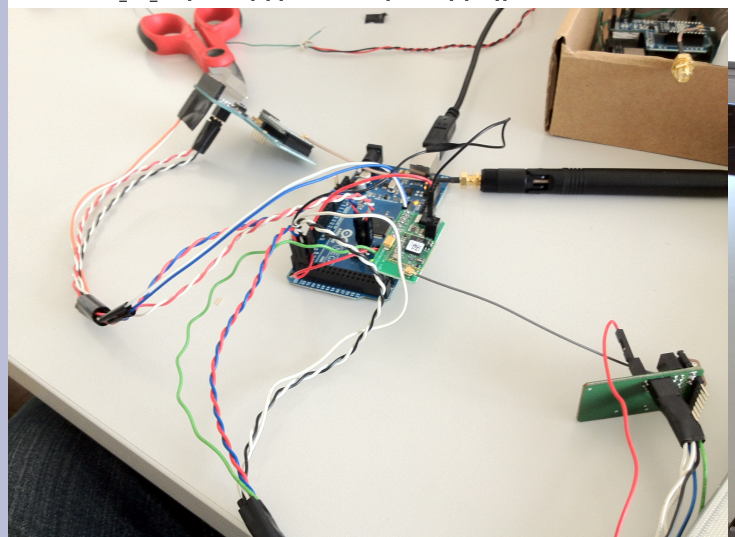
Configurazione eliche
In un quadricottero

Vantaggi UAV VTOL

- Stabilità in volo
- Volo verticale in poco spazio
- Nel caso di più di 4 rotori se si rompesse un rotore il drone continuerebbe a volare lo stesso, potendo quindi eseguire l'atterraggio in sicurezza
- Svantaggio: consumo

SVILUPPO

```
    endp++;
    percorso[i].raggio = strtod( endp, &endp );
    endp++;
    percorso[i].altezza = strtod( endp, &endp );
    endp++;
}

free( buffer_coordinate );

// conversione coordinate da gradi in metri rispetto alla posizione lat: 0°, long: 0°
for( i = 0; i < waypoints; i++ )
    converti_coordinate( &( percorso[i].posizione ) );

// calcolo delle equazioni che descrivono le regioni dello spazio entro cui il drone può muoversi liberamente senza che questi venga considerato
for( i = 1; i <= waypoints; i++ )
{
    if ( percorso[i].posizione.y == percorso[i-1].posizione.y )
    {
        t1_cur_x = percorso[i].posizione.x;
        t1_cur_y = percorso[i].posizione.y + percorso[i].raggio;

        t2_cur_x = percorso[i].posizione.x;
        t2_cur_y = percorso[i].posizione.y - percorso[i].raggio;

        t1_prec_x = percorso[i-1].posizione.x;
        t1_prec_y = percorso[i-1].posizione.y + percorso[i-1].raggio;

        t2_prec_x = percorso[i-1].posizione.x;
        t2_prec_y = percorso[i-1].posizione.y - percorso[i-1].raggio;

        percorso[i].allineamento = 0;
    } else if ( ( percorso[i].posizione.x == percorso[i-1].posizione.x ) && ( percorso[i].raggio == percorso[i-1].raggio ) )
    {
        percorso[i].inters.punti.x1 = percorso[i].posizione.x - percorso[i].raggio;
        percorso[i].inters.punti.x2 = percorso[i].posizione.x + percorso[i].raggio;
        percorso[i].allineamento = 1;
    } else if ( ( percorso[i].posizione.x == percorso[i-1].posizione.x ) )
    {
        t1_cur_x = percorso[i].posizione.x + percorso[i].raggio;
```



Generazione MD5 – parte 1

- Il programma da noi creato si occupa di calcolare calcola l'hash MD5 del piano di volo e di scriverlo su un file separato che sarà poi caricato sulla scheda micro SD insieme al piano di volo stesso
- Conversione del file del piano di volo esportato dal programma di configurazione fornito dal produttore del drone dal formato originale ad un formato più semplice e manipolabile da Arduino

Generazione MD5 – parte 2

- formato del file delle coordinate caricate sulla microSD:

Latitudine	longitudine	raggio	altezza	\n

```
Scegliere:
1)Conversione + md5
2)md5
3)uscita
Scelta: 1
Inserire file del percorso: percorso1.wpl
Caratteri: 156
Messaggio: 45,4339168;8,6074612;20;15
45,4338595;8,6073343;20;0
45,4339295;8,6071295;20;0
45,433834;8,6071051;20;0
45,4337958;8,6073295;20;0
45,4338468;8,6074855;20;0

MD5: 139D82A509EF2D85794C3DDB9D484911
Scegliere:
1)Conversione + md5
2)md5
3)uscita
Scelta:
```

Logging

- Controllo rispetto del piano di volo
- Scrittura della posizione corrente, accompagnata da un messaggio che indica il rispetto del piano di volo, nel file della micro SD

Data	Ora	Lat.	Lon.	Alt.	OK/ERR	Eq retta 1	Eq retta 2	Alt. prevista

Invio dati – parte 1

- Si fa la stessa cosa del logging inviando però i dati su una porta seriale tramite onde radio (protocollo ZigBee). Ovviamente si effettua un controllo CRC per evitare errori derivanti dal canale trasmissivo

preambolo	Id del drone	Lat.	Lon.	Alt.	Validità	Checksum	Coda	\0

- Preambolo e coda impostati in modo fisso rispettivamente a 'I' e 'F' per aiutare a delimitare correttamente il frame (1 byte ciascuno)

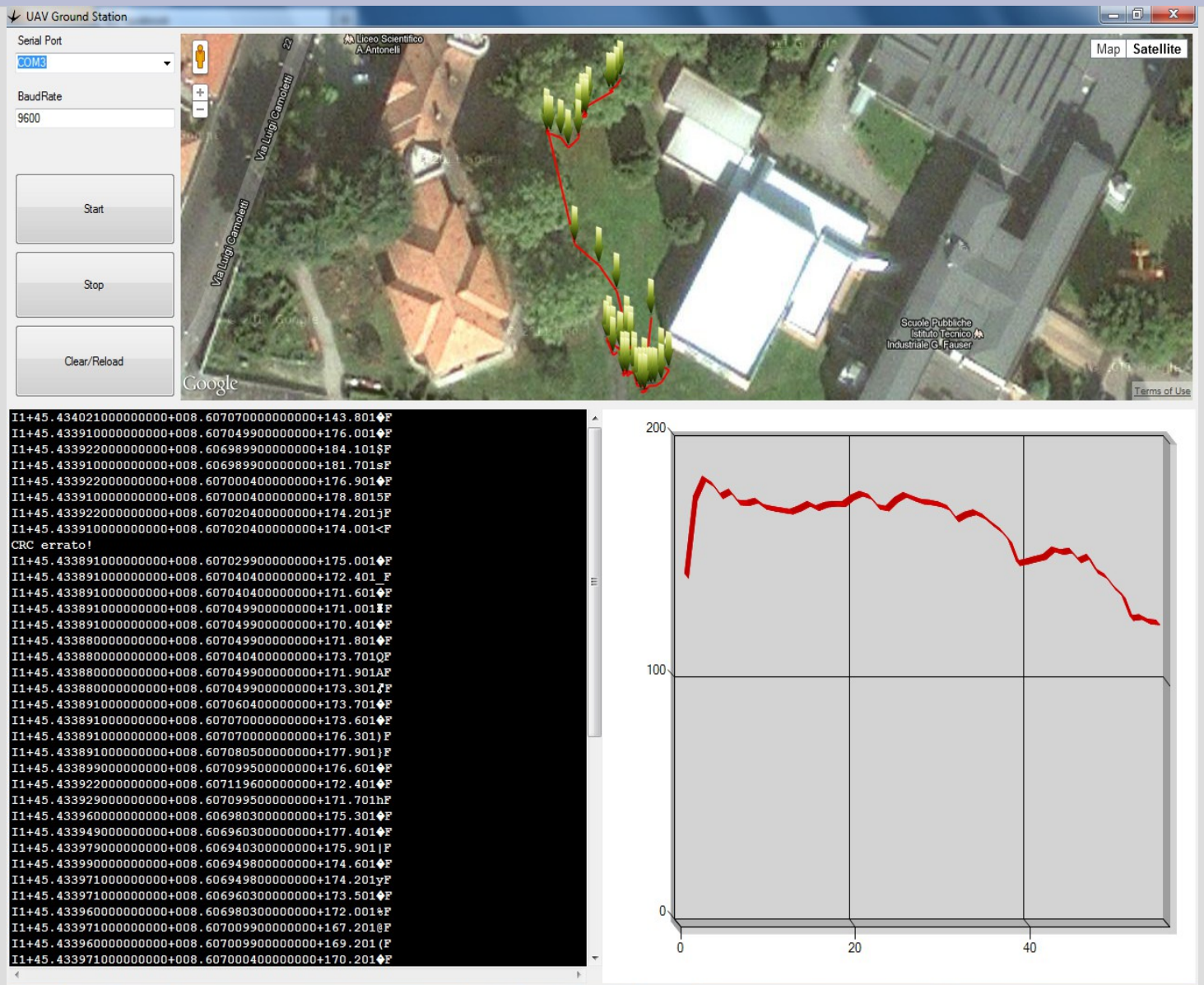
Invio dati – parte 2

- Identificativo del drone (1 byte)
- Latitudine (convertito da un numero in virgola mobile, 19 byte)
- Longitudine (come per la latitudine, 20 byte)
- Altitudine (7 byte)
- Campo validità, indica se il drone si trova in posizione corretta rispetto al piano di volo prestabilito (1) oppure no (0) (1 byte)

Invio dati – parte 3

- Checksum, calcolato mediante l'algoritmo 8CRC con polinomio generatore x8+x5+x4+1, serve per rilevare la presenza di errori nel canale radio utilizzato dai moduli XBee per comunicare (1 byte)
- \0, carattere terminatore della stringa (1 byte)
- Lunghezza totale del frame: 52 byte

Invio dati – parte 4



Invio dati – parte 5

- Esempi di cordinate utilizzate nel programma

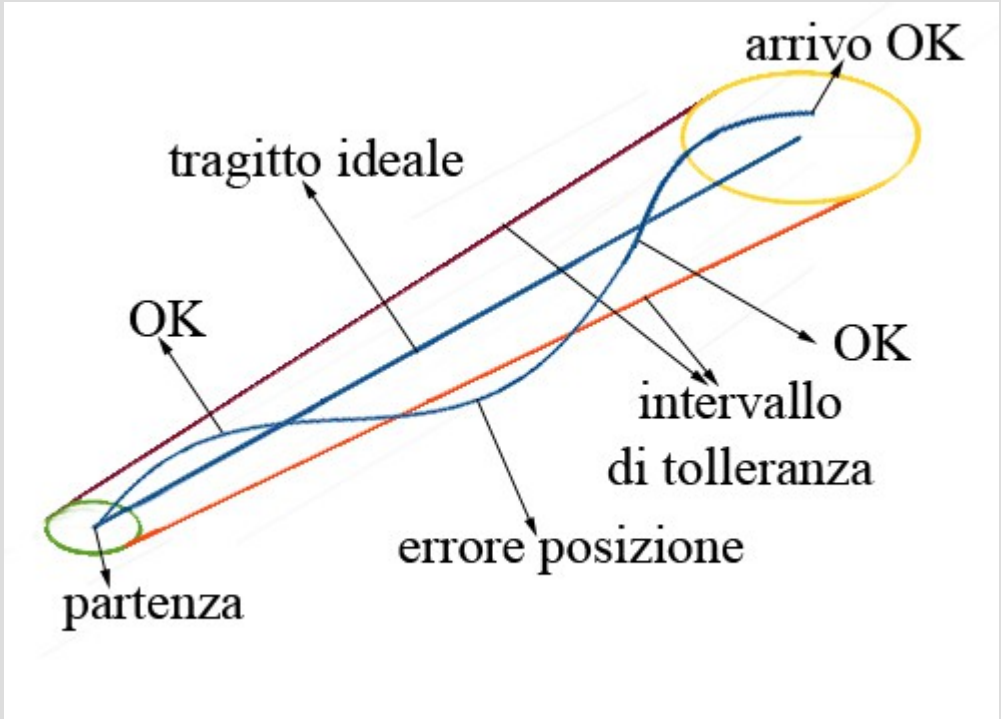
I1+45.434021000000000+008.607070000000000+143.801ãF
I1+45.433910000000000+008.607049900000000+176.001»F
I1+45.433922000000000+008.606989900000000+184.101$F
I1+45.433910000000000+008.606989900000000+181.701sF
I1+45.433922000000000+008.607000400000000+176.901ìF
I1+45.433910000000000+008.607000400000000+178.8015F
I1+45.433922000000000+008.607020400000000+174.201jF
I1+45.433910000000000+008.607020400000000+174.001<F
I1+45.433910000000000+008.607029900000000+173.201F
I1+45.433891000000000+008.607029900000000+175.001ÙF
I1+45.433891000000000+008.607040400000000+172.401_F
I1+45.433891000000000+008.607040400000000+171.601ÂF
I1+45.433891000000000+008.607049900000000+171.001F
I1+45.433891000000000+008.607049900000000+170.401ùF

Controllo piano di volo – parte 1

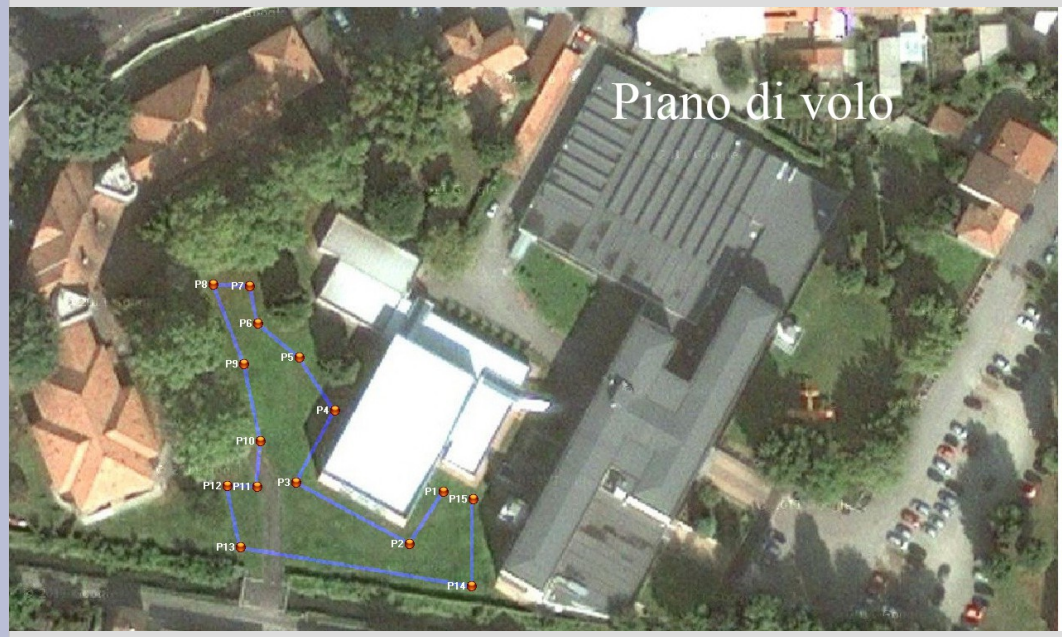
- Ogni waypoint è accompagnato da una tolleranza espressa in metri
- La latitudine e la longitudine vengono convertite da gradi a metri
- Si calcola quindi l'area di tolleranza per ogni waypoint
- Si calcolano le equazioni (tramite la trigonometria e la geometria analitica) delle rette che uniscono gli estremi delle circonferenze di tolleranza dei waypoint

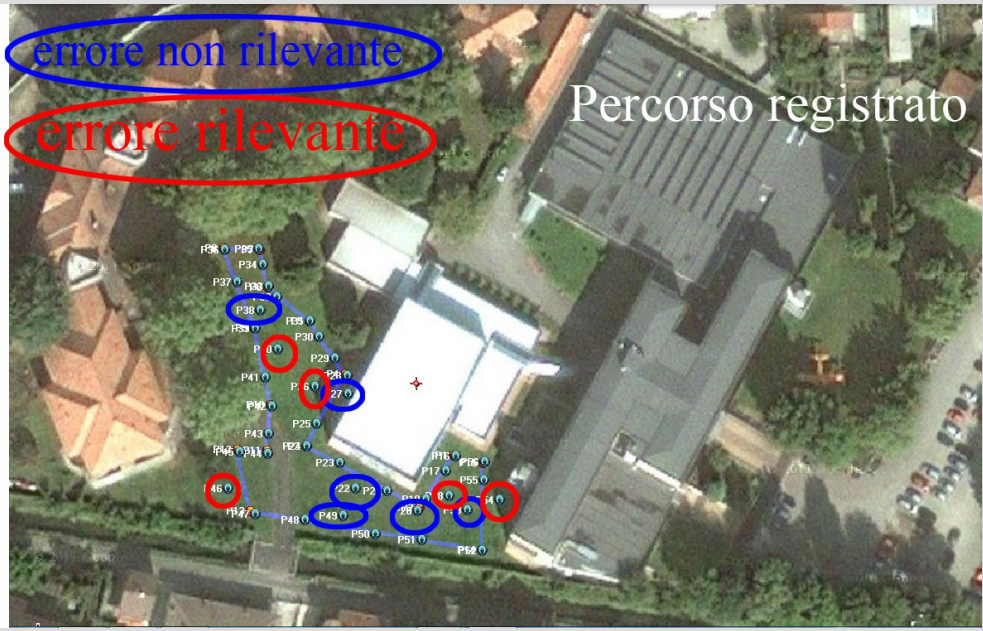
Controllo piano di volo – parte 2

- Ad ogni posizione ricevuta si verifica se questa è all'interno delle due rette, basta convertire la posizione in metri e sostituirla all'interno delle equazioni.

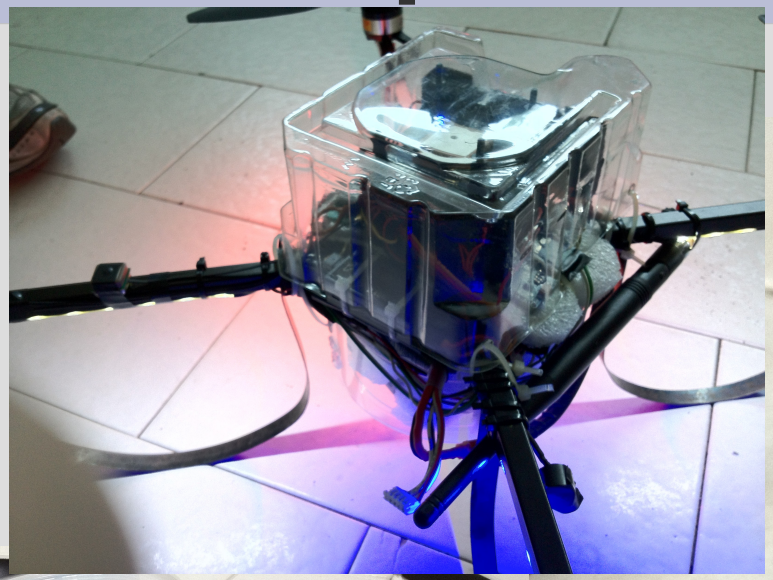


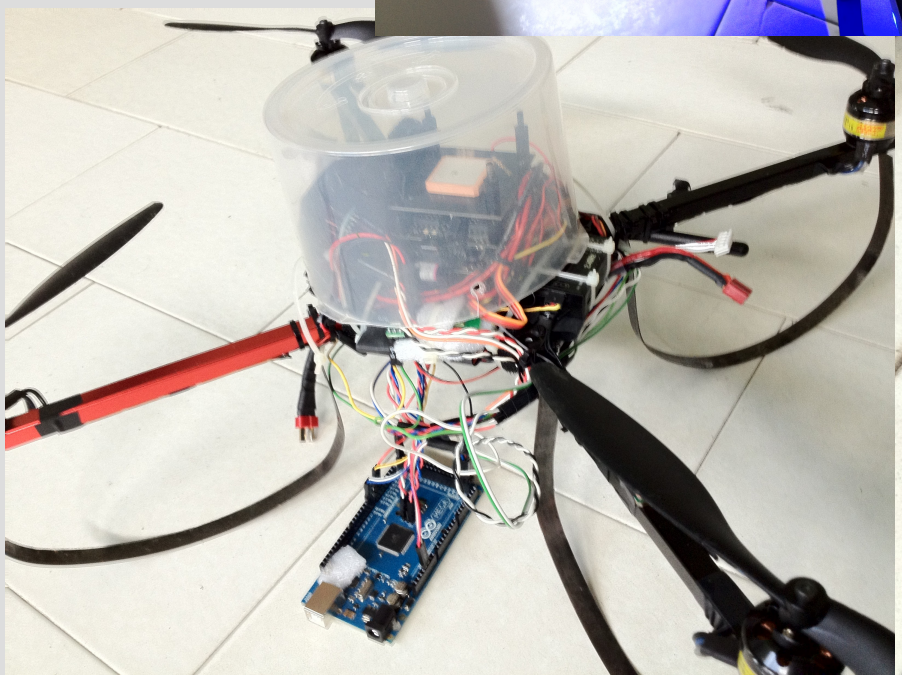
Controllo piano di volo – parte 3

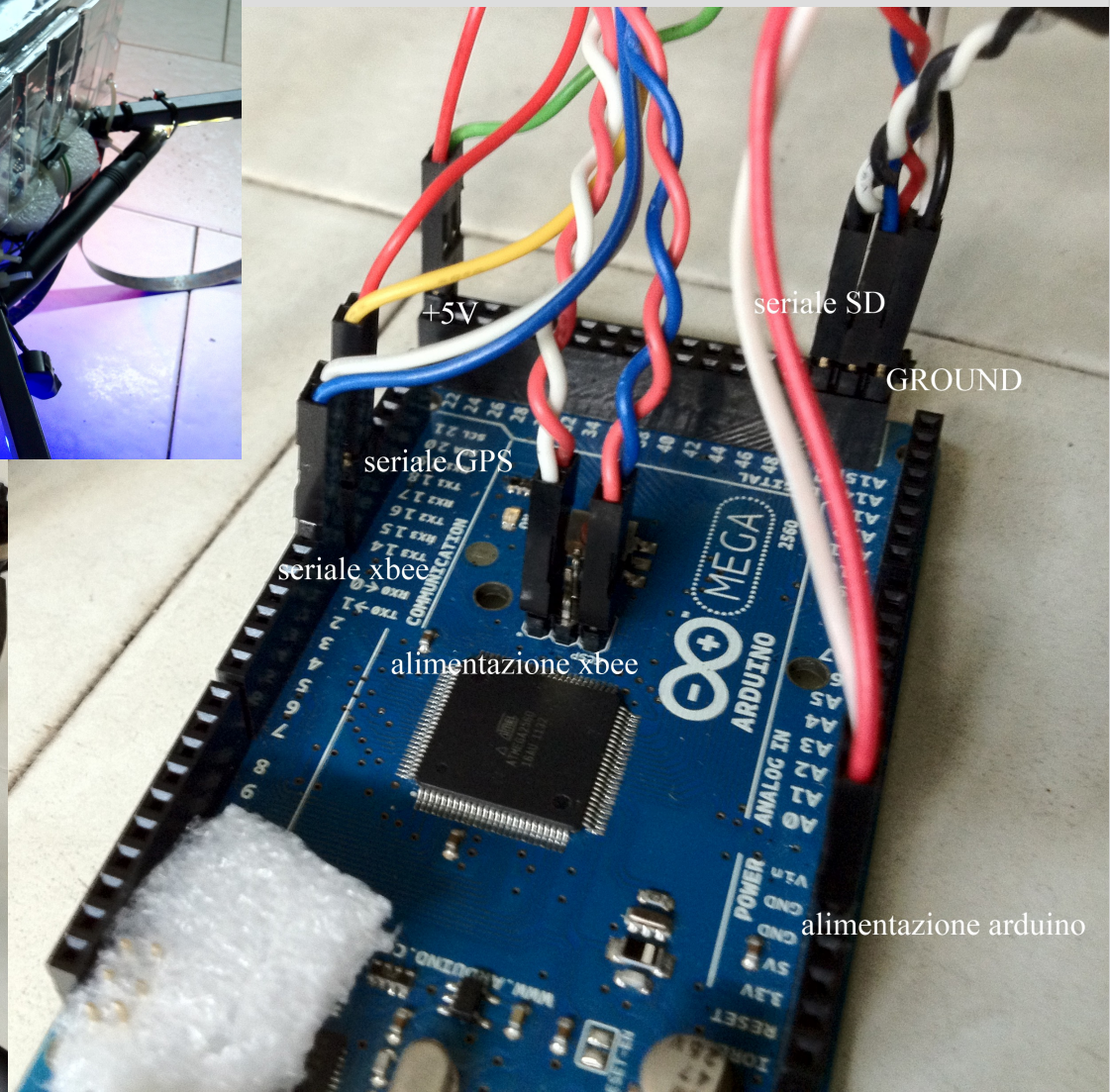




Modulo payload arduino parte 1







Modulo payload arduino parte 2

- Arduino mega “al centro di tutto” collegato a:
  - Lettore schede micro SD
  - XBee shield
  - GPS
- Arduino mega inserito in un'apposita cupola, da noi costruita, sotto il drone in modo da poterlo facilmente estrarre per riprogrammarlo
- Shield fissati sul drone in modo da poter ricevere facilmente il segnale radio (GPS e XBee) o in modo da essere facilmente accessibili (SD)

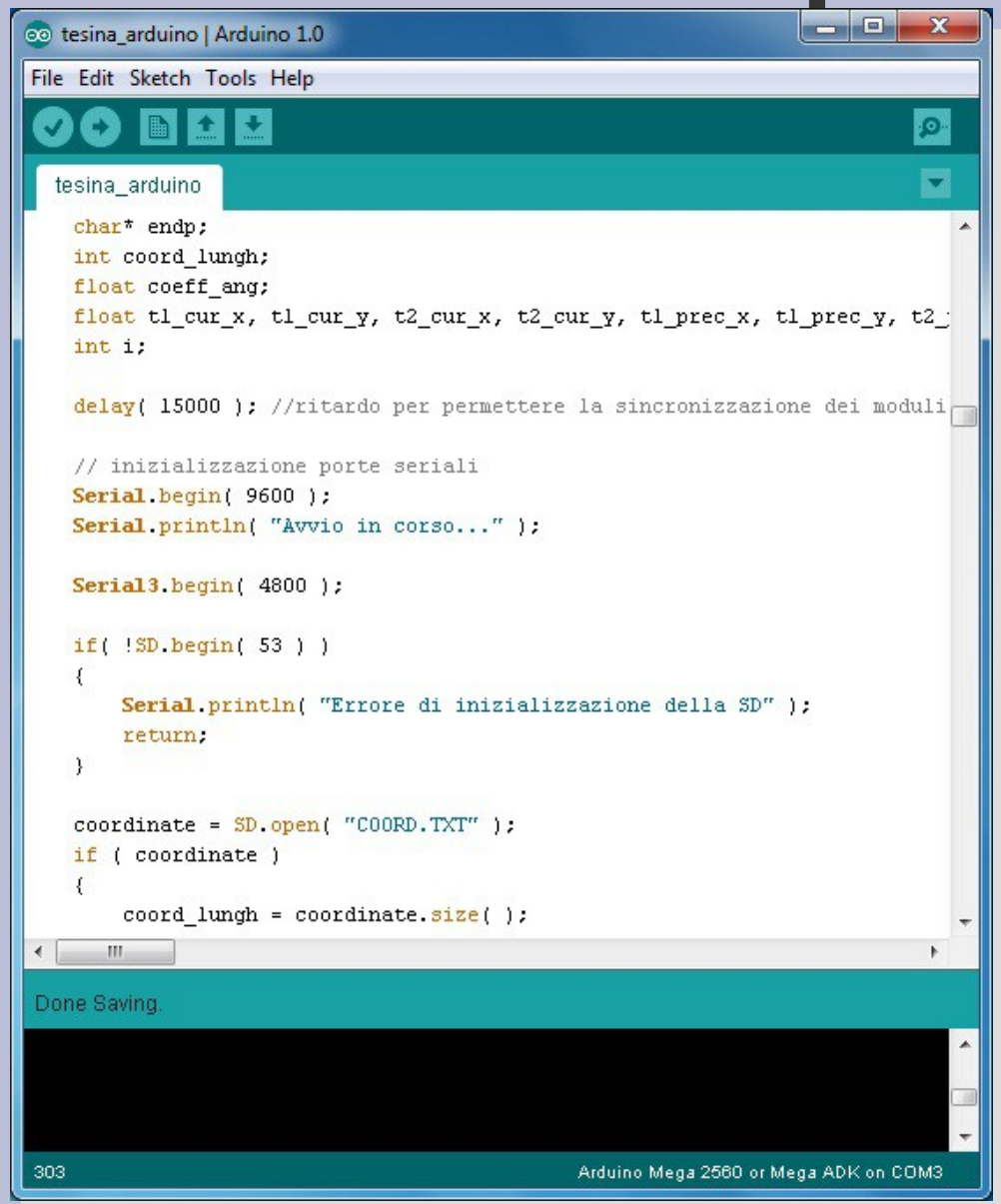
Modulo payload arduino parte 3

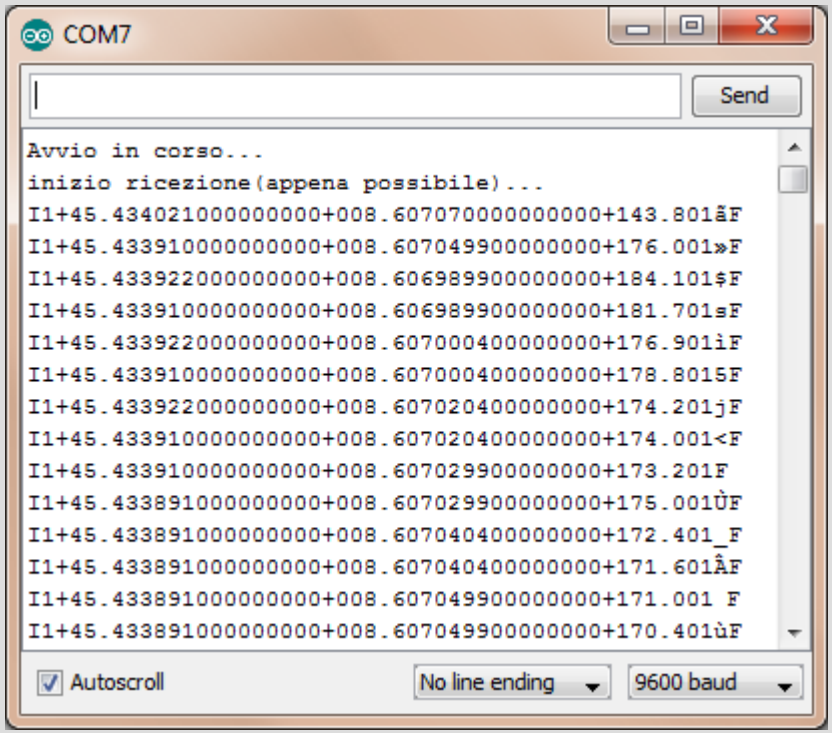
- Compiti arduino:
  - Comunicazione con gli altri componenti del payload
  - Verifica periodica della posizione GPS del drone e confronto con il piano di volo pre-impostato
  - Trasmissione di informazioni (attraverso un protocollo di comunicazione a pacchetti) al computer, utilizzando un canale radio dedi-cato(XBee)
  - Attività di registrazione periodica sulla scheda SD del percorso GPS del drone

Modulo payload arduino parte 4

- Programma codificato in linguaggio C/C++, esecuzione di un solo programma alla volta (monotasking)
- Due funzioni principali: setup() che viene eseguita una sola volta come funzione di inizializzazione e loop() che viene invocata ciclicamente
- Compilazione eseguita su PC tramite un apposito software multipiattaforma. Il programma viene in seguito caricato tramite USB o XBee

Modulo payload arduino parte 5





Ruoli funzione setup – parte 1

- inizializza i moduli Arduino di supporto (lettore microSD, canali seriali per XBee e GPS). “Aspettando” circa 15 secondi affinché siano tutti inizializzati
- caricamento del file delle coordinate con il piano di volo e calcolo hash MD5 (utilizzando la libreria AVRCrypto_Lib)
- confronto hash MD5 con quella calcolata dal programma realizzato su PC

Ruoli funzione setup – parte 2

- in caso confronto andato a buon fine, caricamento del piano di volo all'interno del programma altrimenti terminazione precoce dell'esecuzione
- conversione di tutte le coordinate dal sistema sessagesimale in metri calcolato come distanza dal punto 0°;0°, mediante le formule

$$\Delta S_{long} = (\arccos(\sin^2(lat) + \cos^2(lat) * \cos(-long))) \cdot 6372$$
$$\Delta S_{lat} = lat \cdot 6372$$

latitudine e longitudine sono espresse in radianti

Ruoli funzione setup – parte 3

- calcolo delle rette di tolleranza per ogni coppia di waypoint
- apertura del file di log

Ruoli funzione loop – parte 1

- Attende un secondo per la ricezione dei dati GPS dal modulo
- Controlla l'effettiva presenza di nuovi dati dal GPS
- In caso affermativo li acquisisce e li converte in metri come per le coordinate dei waypoint del piano di volo in modo da poter essere confrontati
- Scrittura sul file di log della data e ora di ricezione dei dati GPS, della latitudine, longitudine e altitudine rilevata, delle equazioni delle rette di tolleranza e dell'altitudine prevista

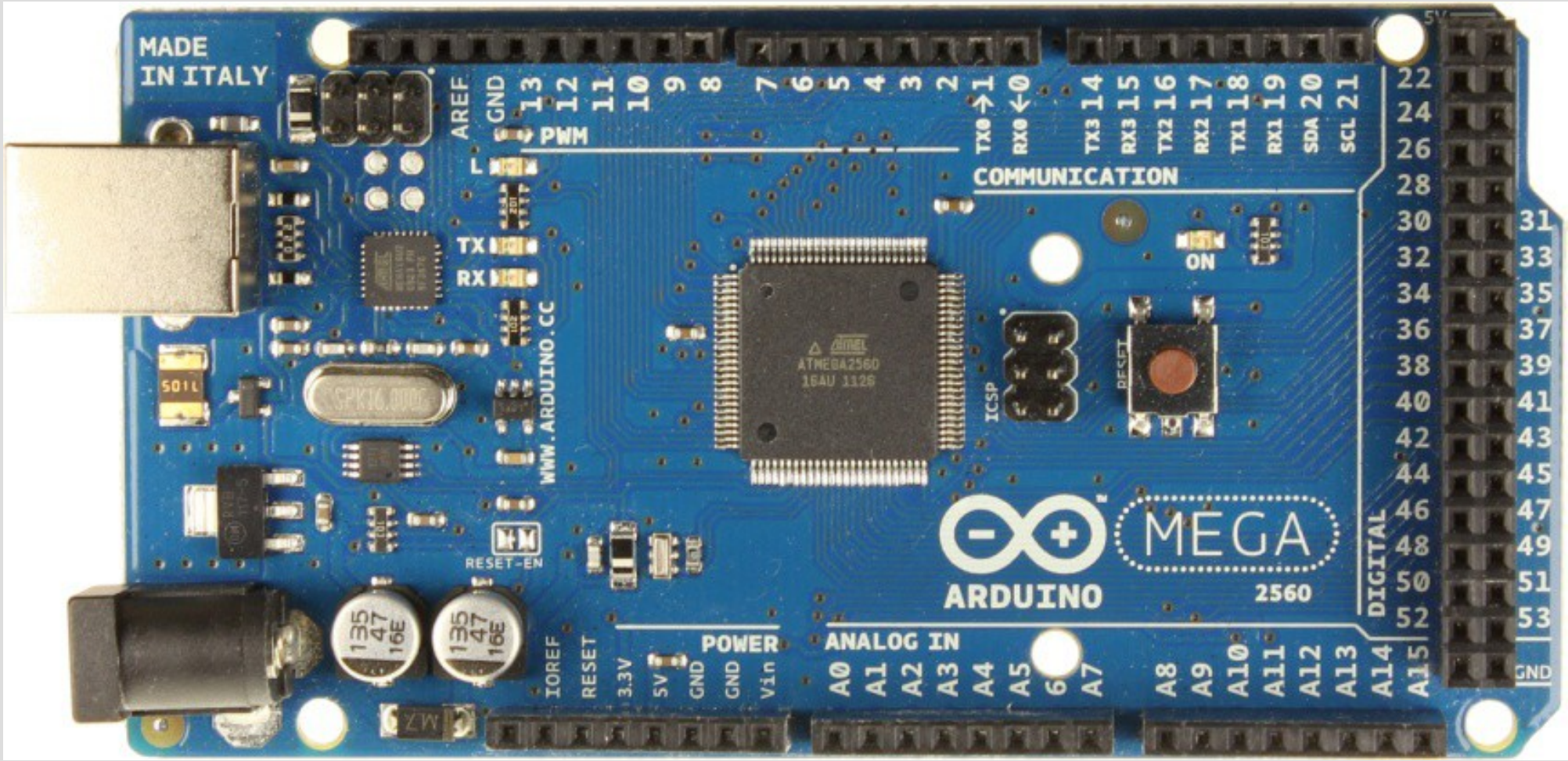
Ruoli funzione loop – parte 2

- Controllo vero e proprio della correttezza e invio del frame con alcune informazioni necessarie per la corretta visualizzazione grafica del programma su PC
- Evita di ritardare la scrittura sulla mircoSD (nessun tipo di buffer viene utilizzato) del file di log per evitare corruzione e perdita di dati in caso di interruzione dell'alimentazione

Arduino MEGA 2560 – parte 1

- CPU: ATmega2560 (freq. di clock a 16MHz)
- Memoria flash: 256KB di cui 8KB usati dal bootloader
- RAM: static RAM da 8KB
- 54 Pin digitali di I/O
- 4 Porte seriali: Serial (pin 0 e 1), Serial1 (pin 18 e 19), Serial2 (pin 16 e 17) e Serial3 (pin 15 e 14)

Arduino MEGA 2560 – parte 2

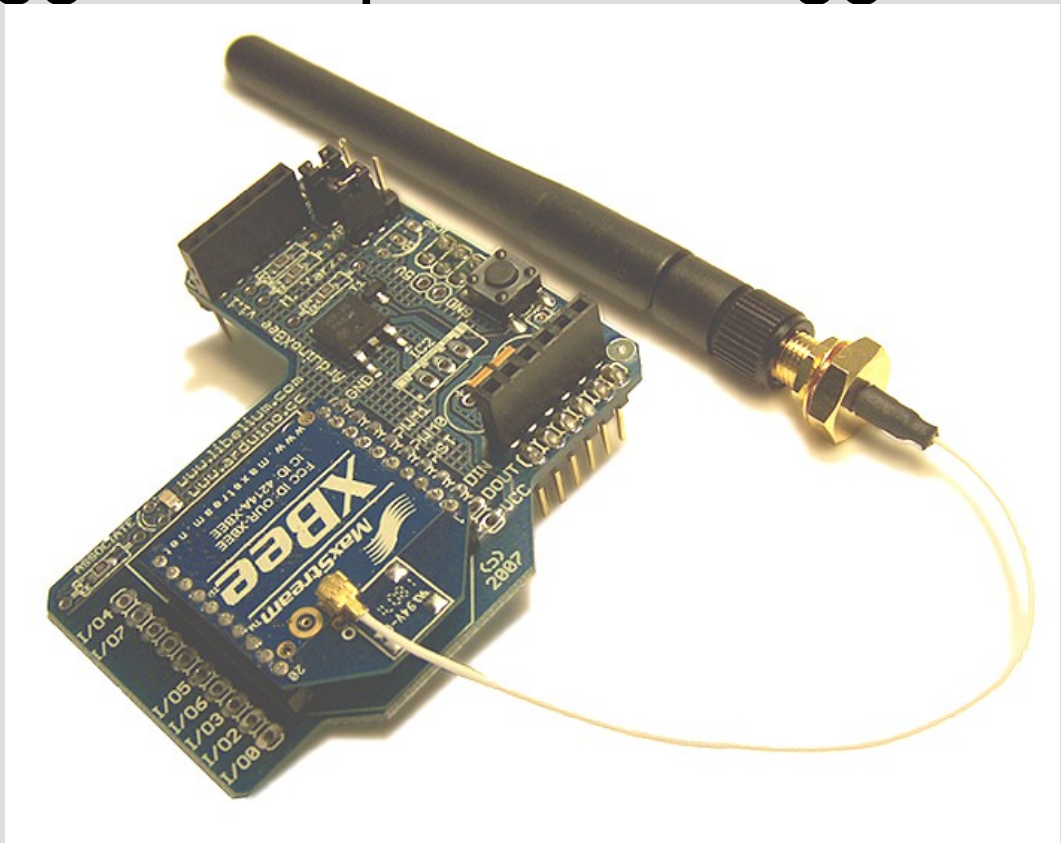


Modulo Xbee – parte 1

- modulo che permette la comunicazione senza fili di dispositivi, attraverso un canale di comunicazione seriale emulato
- Velocità del canale seriale configurabile (tra 1200bps e 115200bps)
- Possibilità di creare una PAN (Personal Area Network) con topologia a stella
- Frequenza di funzionamento del segnale radio di 2.4GHz

Modulo Xbee – parte 2

- Due modelli: XBEE, la versione base; XBEE Pro, versione evoluta, maggiore potenza, maggior raggio di copertura, maggiori consumi



Xbee explorer

- adattatore USB per il collegamento del PC al canale XBEE

Modulo microSD reader parte 1

- Comunicazione attraverso il protocollo seriale SPI sincrono (utilizza un segnale periodico di clock per sincronizzarsi)
- Supporto per schede microSD fino a 2GB (partizionate con file system FAT16)
- Programmazione attraverso la libreria SD per Arduino che fornisce primitive di base comuni

Modulo microSD reader parte 2

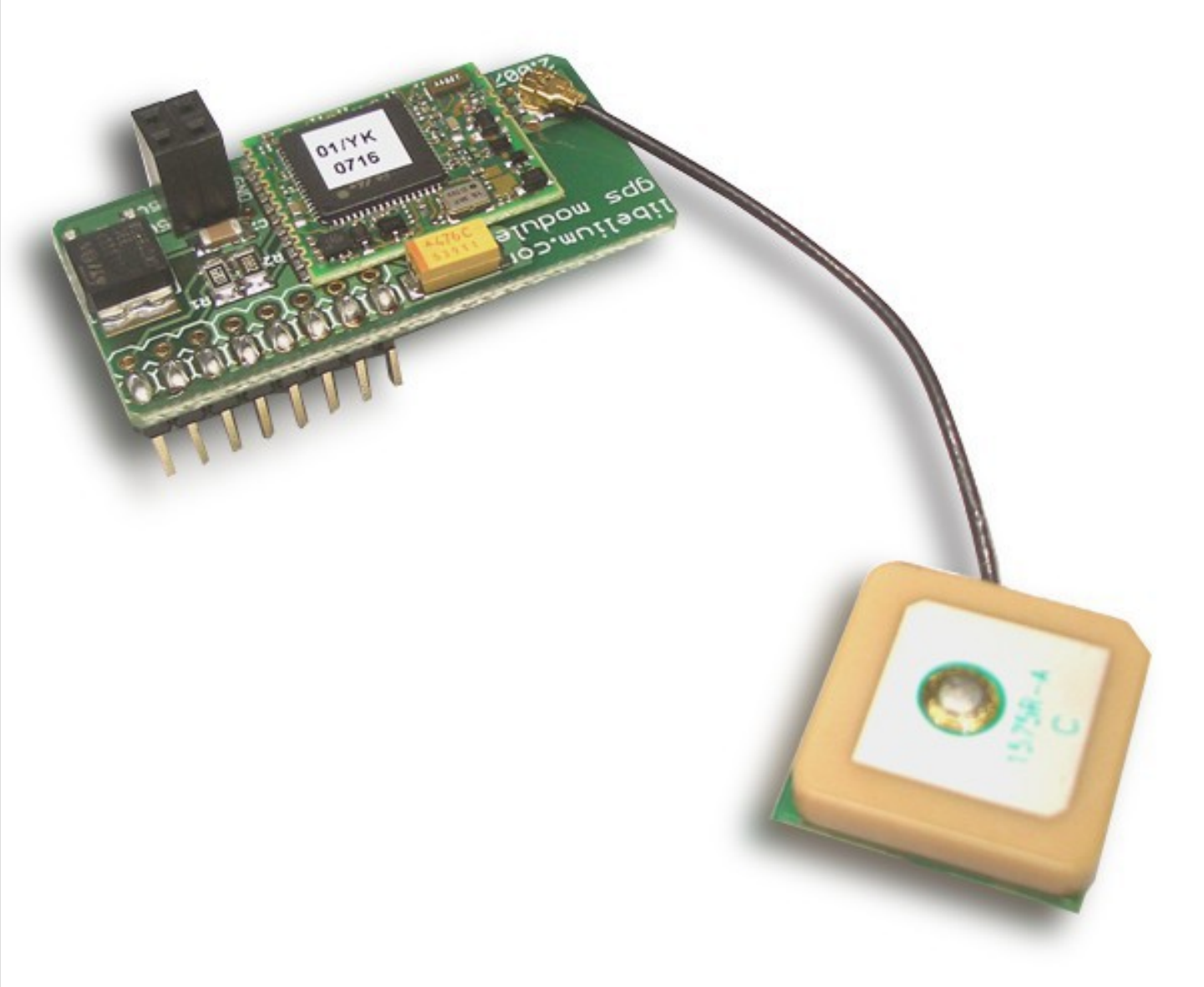
Modulo GPS – parte 1

- Modulo GPS prodotto da Libelium
- Comunicazione con i satelliti GPS attraverso il protocollo NMEA (National Marine Electronics Association), costituito da una serie di messaggi di testo
- Le stringhe NMEA sono disponibili su interfaccia seriale
- Velocità di default di 4800 baud

Modulo GPS – parte 2

- Basato su Vincotech A1080-b, Il modulo adatta i livelli elettrici del ricevitore Vincotech alla seriale Arduino
- Aggiornamento ogni secondo
- Utilizzo delle librerie TinyGPS per la programmazione del modulo
- Libreria TinyGPS che fornisce un’interfaccia per la manipolazione delle stringhe NMEA

Modulo GPS – parte 3



Prove sperimentali – parte 1

- Calibrazione iniziale del drone
  - Dopo aver finito di montare il drone assicurarsi che sia posto in una posizione dritta
  - Viene eseguita la calibrazione dell'accelerometro, del giroscopio e della bussola
  - Si testano dei motori: con l'MK-USB collegata alla FlightCtrl si esegue l'applicazione MikroKopterTool e si fa la prova dei motori facendo attenzione al verso in cui gira ogni elica

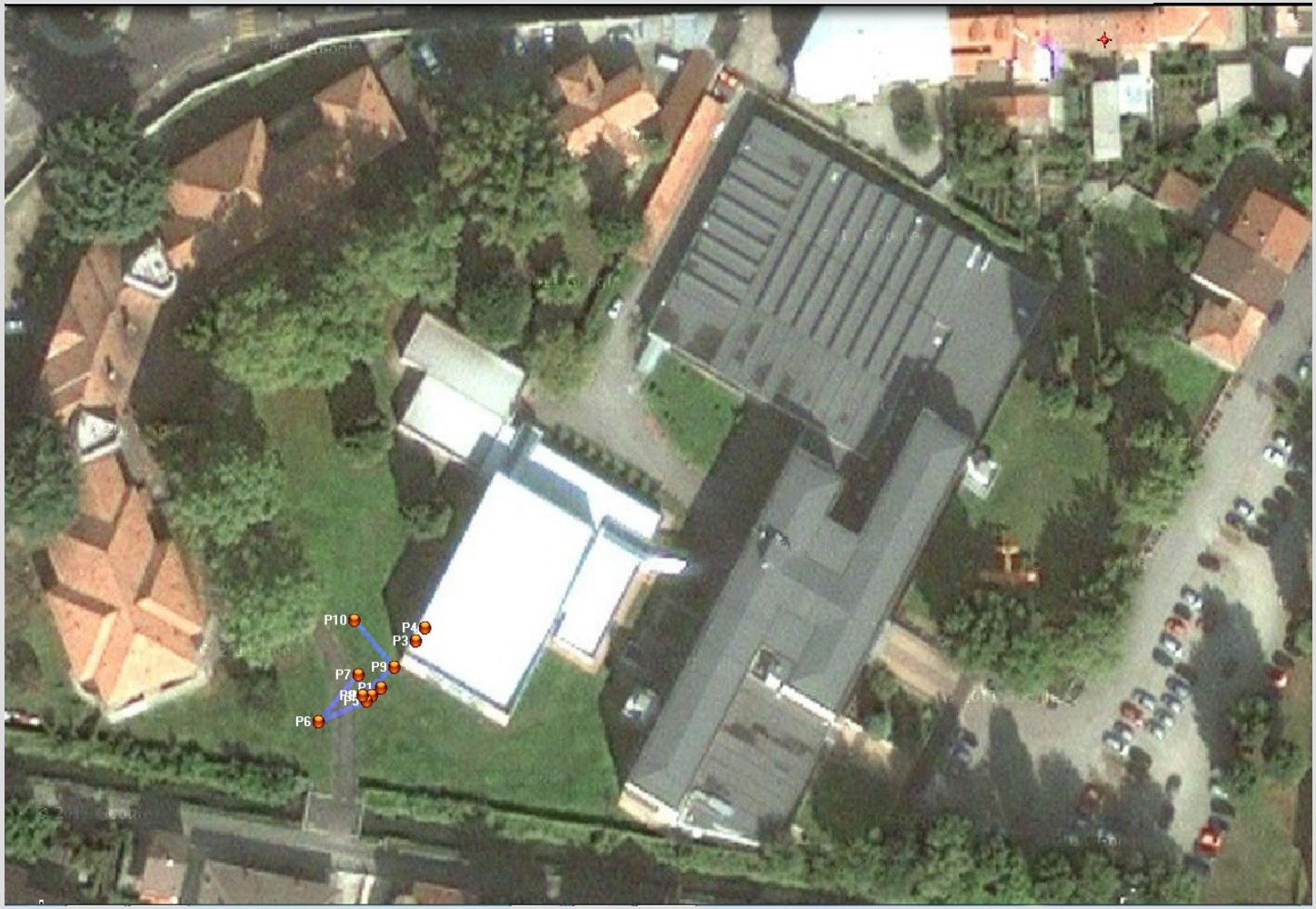
Prove sperimentali – parte 2

- Equilibrio del drone
  - Dopo la calibrazione si eseguono alcuni test facendo volare il drone raso terra per configurare la calibrazione del telecomando e imparare le prime manovre
  - Successivamente, in volo, si eseguono le altre prove per le quali è necessario l'uso del GPS, come ad esempio la funzionalità di AltitudeHold e PositionHold

Prove sperimentali – parte 3

Prove sperimentali – parte 4

- Coordinate registrate in volo



Prove sperimentali – parte 4

25/05/2012 11:34:1;+45.433941000000000;+008.607060400000000;+156.500;OK
25/05/2012 11:34:1;+45.433929000000000;+008.607040400000000;+162.200;OK
25/05/2012 11:34:4;+45.434010000000000;+008.607139600000000;+142.700;OK
25/05/2012 11:34:5;+45.434029000000000;+008.607159600000000;+139.100;OK
25/05/2012 11:36:0;+45.433922000000000;+008.607029900000000;+170.000;ERR
25/05/2012 11:37:1;+45.433891000000000;+008.606920200000000;+183.700;ERR
25/05/2012 11:39:4;+45.433960000000000;+008.607009900000000;+170.000;ERR
25/05/2012 11:39:5;+45.433929000000000;+008.607020400000000;+175.000;OK
25/05/2012 11:40:2;+45.433971000000000;+008.607090000000000;+145.200;OK
25/05/2012 11:40:4;+45.434040000000000;+008.607000400000000;+171.000;ERR

Problemi riscontrati nelle prove sperimentali – parte 1

- I problemi riscontrati riguardano soprattutto la durata della batteria molto bassa
- Di conseguenza il tempo di volo del drone varia in base al tempo che ci mettono gli altri componenti a regolarsi
- Soprattutto per quanto riguarda il GPS è essenziale il tempo che passa prima che si sincronizza con i satelliti e riesce a determinare una posizione abbastanza precisa

Problemi riscontrati nelle prove sperimentali – parte 2

- Questo tempo è di circa 4 minuti e tenendo conto che la batteria dura al massimo 12 minuti, per 1/3 del tempo totale di volo il drone potrebbe volare soltanto in modalità manuale
- Sempre per problemi relativi al GPS, in modalità PositionHold il drone inizia a ruotare intorno a se stesso finché si calibra e mantiene la posizione
- Il GPS collegato ad arduino, oltre ad impiegare pure 4 minuti ad “agganciare” i satelliti, è molto impreciso (fino a 45 metri di errore)

Problemi riscontrati nelle prove sperimentali – parte 3

- Precisione GPS

Problemi riscontrati nelle prove sperimentali – parte 4

- Un altro problema riguarda la stabilità del drone in altitudine quando è abbastanza basso (< 5 m da terra): infatti non riesce ad avere un’andatura stabile e continua a scendere fino a toccare terra per poi risalire alla posizione precedente. Questo problema probabilmente è causato dal sensore barometrico. Infatti ad un'altezza maggiore il problema si presenta più raramente.
- Molte volte, probabilmente a causa del sensore barometrico, il drone sale ad altezze molto elevate in poco tempo e in modo quasi incontrollabile

Autori tesina

- Barbu Adrian Alexandru
- Cardinale Claudio
- Dabbene Luca
- Fioratto Raffaele
- Schiavo Fabio