DIETI UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

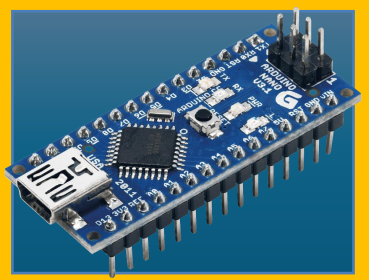
**LAUREA IN INGEGNERIA MECCATRONICA**

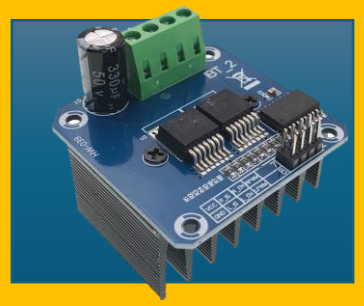
# ARCHIMEDE

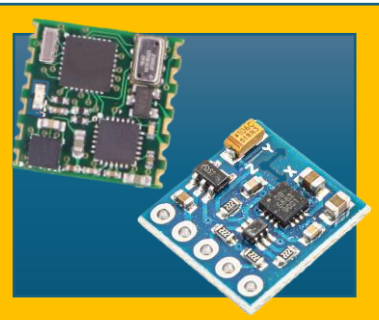
**Candidato Giulio Vestri P27000143**

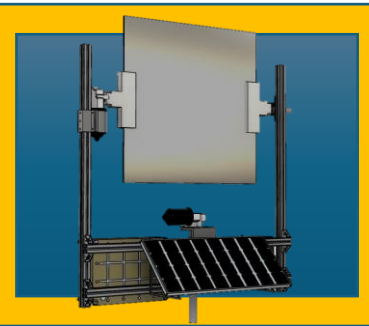
**Relatore Prof. Vincenzo Lippiello**

**Correlatore Dott. Alessandro De Crescenzo**









Startup innovativa e
spin-off del laboratorio
di robotica PRISMA Lab

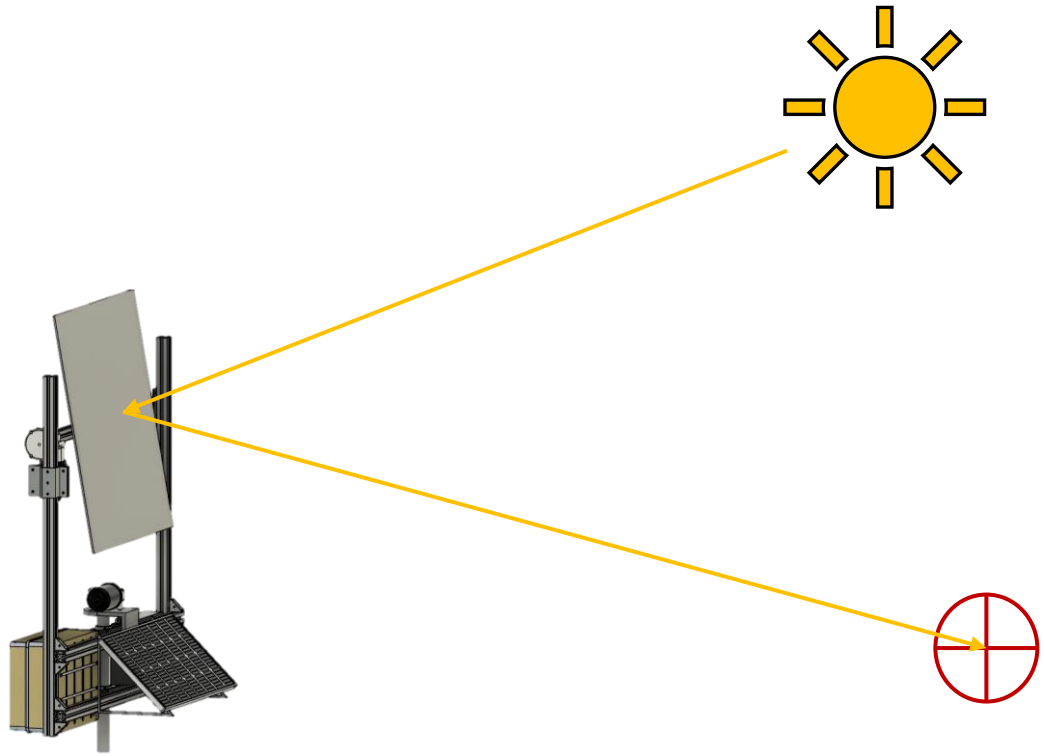
Specializzata nello
sviluppo di soluzioni
robotiche avanzate

# Neabotics

## Service Robotics Solutions

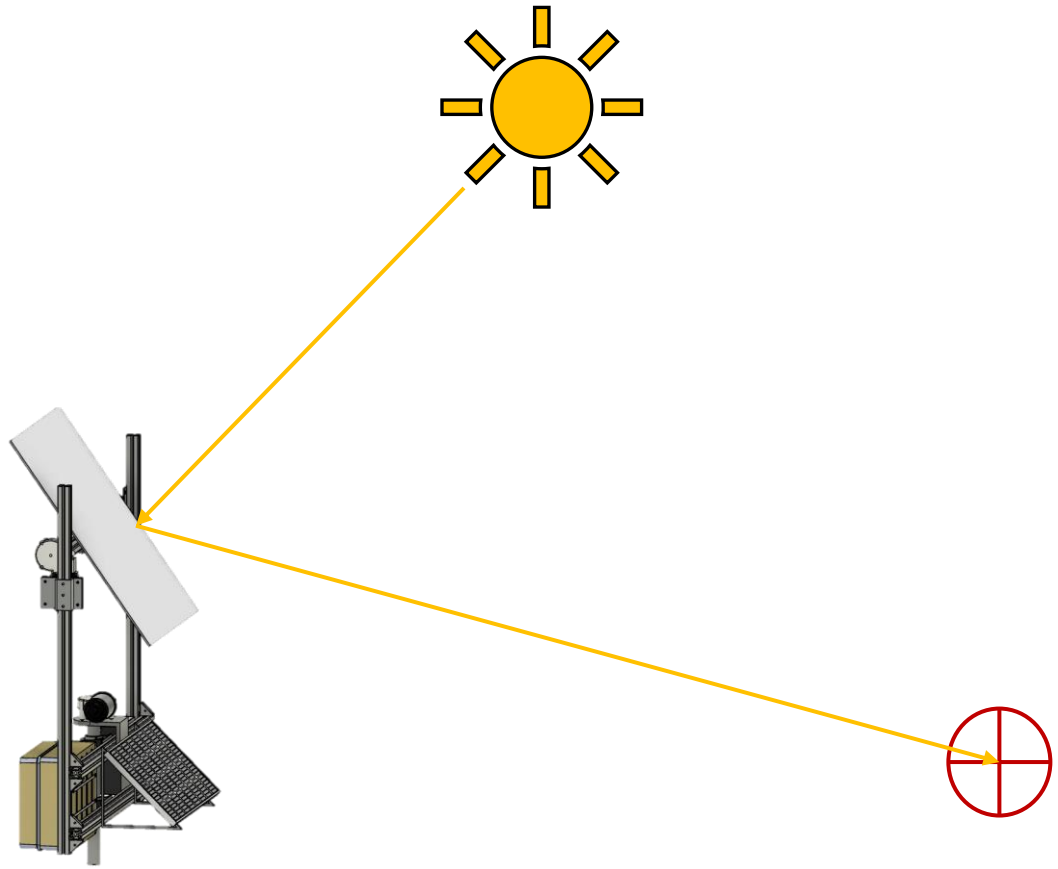
# Specifiche di progetto

Il sistema doveva essere in grado di riflettere i raggi del sole verso un target selezionabile, modificando il proprio orientamento in base al movimento del sole, al fine di garantire l'illuminazione del target durante l'arco della giornata.



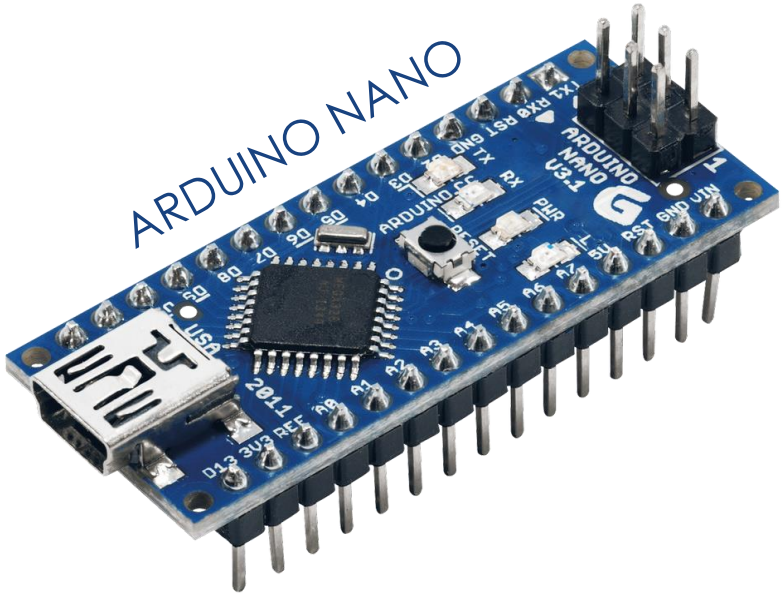
# Specifiche di progetto

Il sistema doveva essere in grado di riflettere i raggi del sole verso un target selezionabile, modificando il proprio orientamento in base al movimento del sole, al fine di garantire l'illuminazione del target durante l'arco della giornata.



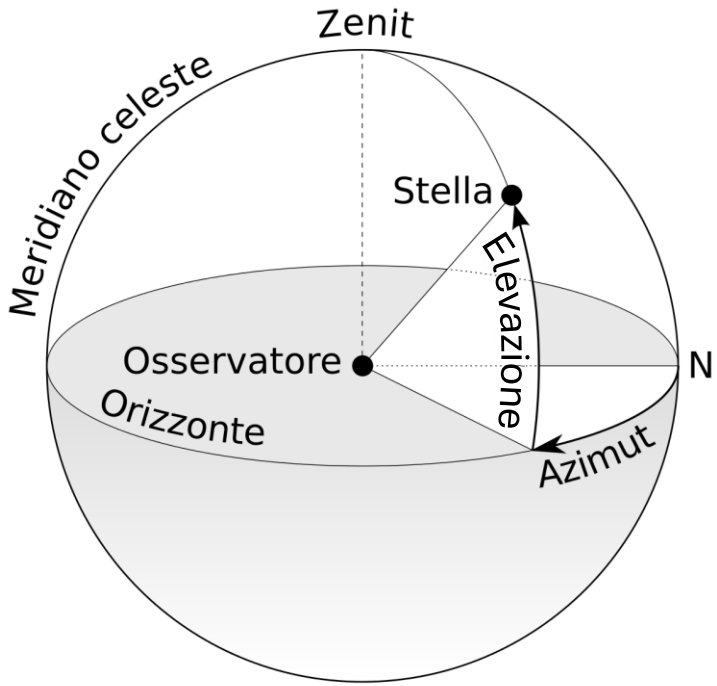
# Steps di progettazione

- Tracciamento della posizione solare
- Selezione coordinate del target
- Calcolo dell' orientamento desiderato
- Sensori e attuatori
- Struttura fisica del sistema
- Filtraggio del rumore
- Controllo di posizione
- Modalità Low-Power



# Tracciamento della posizione solare

Per tracciare il sole ho usato la libreria solarCalculator.h.
Essa restituisce i valori degli angoli Azimut ed Elevazione che permettono di individuare un qualsiasi oggetto all'interno della sfera celeste.




Le grandezze prese variano a seconda della posizione, inserita manualmente come latitudine e longitudine e della data, calcolata in UTC.

Azimut → Angolo rispetto al nord magnetico [0°,360°]
Elevazione → Angolo formato con l'orizzonte [+90°, -90°]

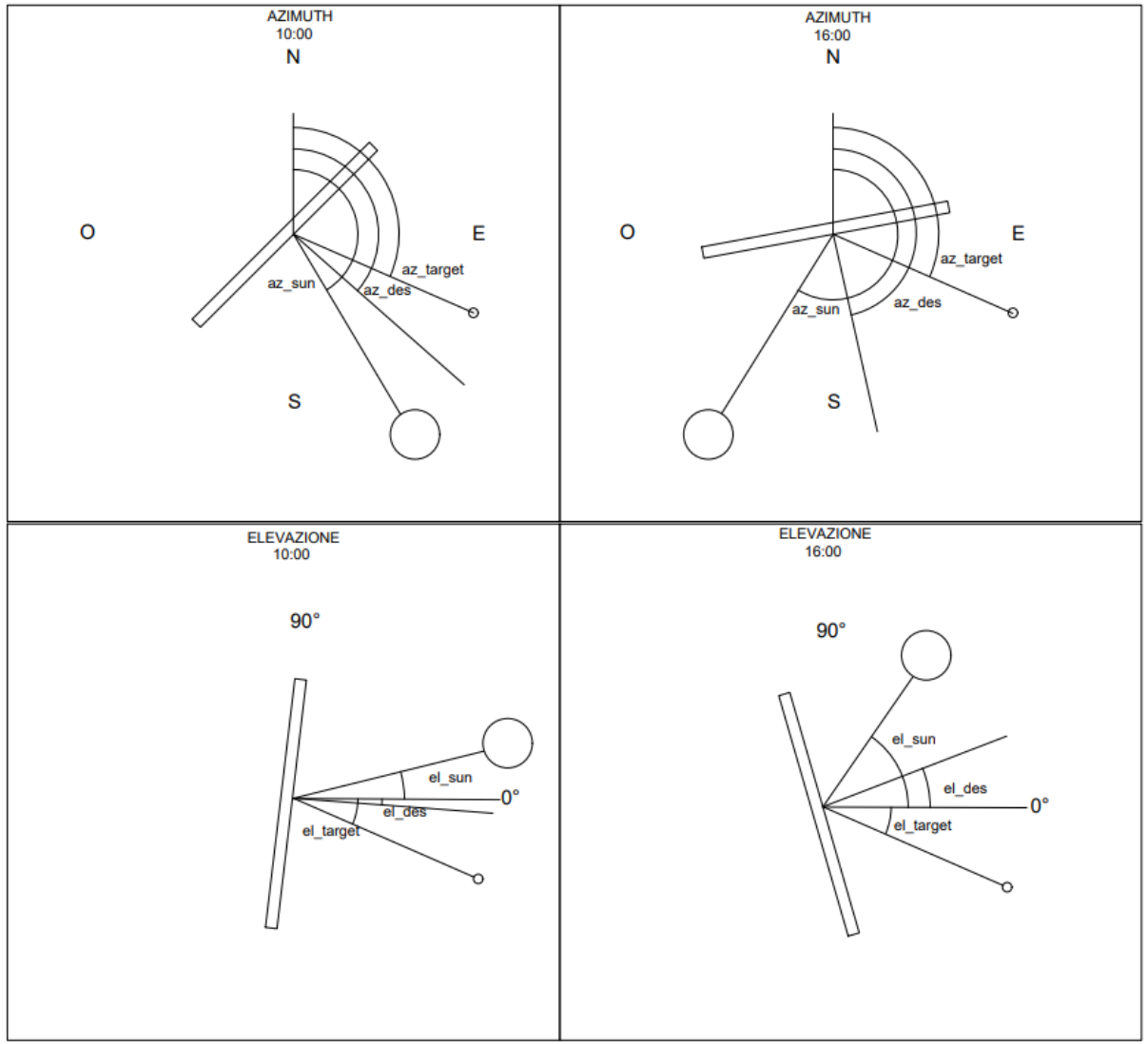
# Selezione coordinate del target

Per selezionare le coordinate del target, ho implementato una funzione all'interno del codice Arduino, selezionabile in qualsiasi momento tramite un interruttore, che permette di modificare l'orientamento dello specchio e salvare la nuova posizione all'interno della EEPROM di Arduino, permettendo così al sistema di avere memoria delle coordinate del target anche dopo lo spegnimento.

# Calcolo dell' orientamento desiderato

Il sistema doveva essere in grado di riflettere i raggi del sole su un target desiderato, attraverso uno specchio piano.

Conoscendo le coordinate del Sole e quelle del target nella sfera celeste, per garantire la riflessione lo specchio doveva essere orientato in modo che la normale coincidesse con l'angolo medio tra target e sole.




$$az_{des} = \frac{az_{target} + az_{sun}}{2}$$

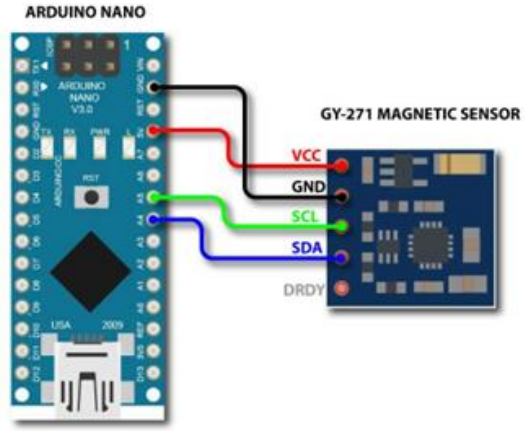
$$el_{des} = \frac{el_{target} + el_{sun}}{2}$$

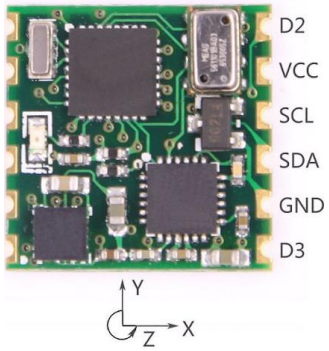
# Sensori e attuatori 1/2

Per permettere il controllo in posizione è necessario effettuare una retroazione tramite dei sensori che possano restituire informazioni utili rispetto all'orientamento dello specchio.

## Azimut

L' azimut è ricavato da un magnetometro GY-271 collegato tramite $I_2C$



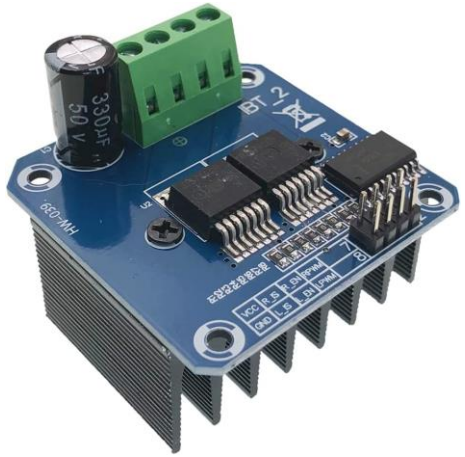




## Elevazione

L'elevazione è ottenuta tramite un WIT-IMU e la lettura dell'angolo di rotazione attorno all'asse x

# Sensori e attuatori 2/2

Gli attuatori dovevano essere in grado di far ruotare il sistema a velocità controllata. Si è optato per un motore con moto riduzione ed un driver per il controllo tramite PWM.



Driver ZK BTS7960B

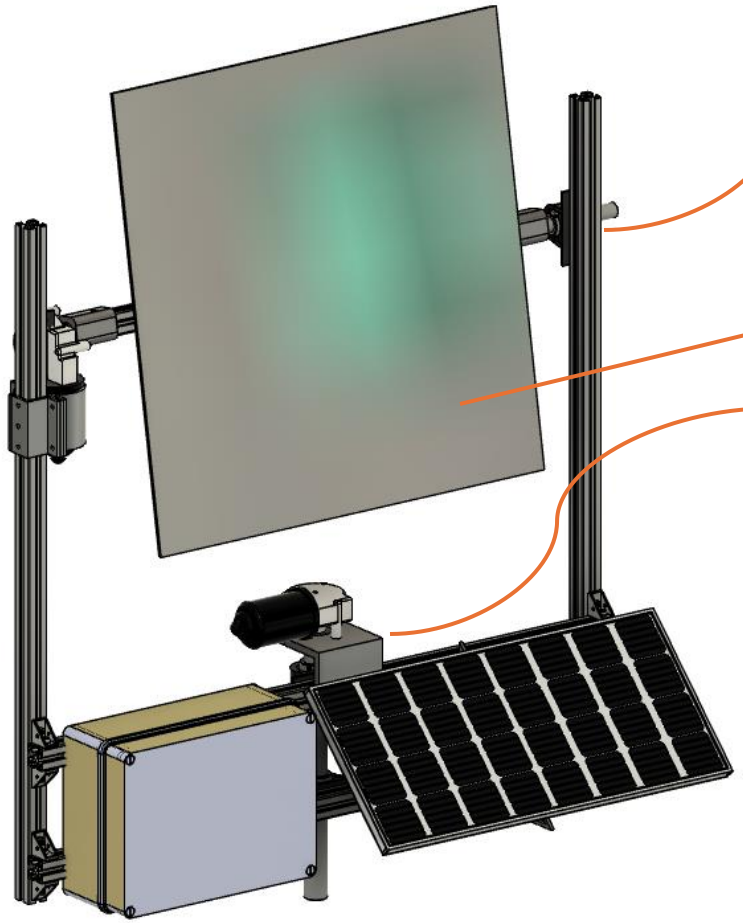
Attraverso segnali PWM permette di pilotare velocità e senso di rotazione dei motori

Motori 12V

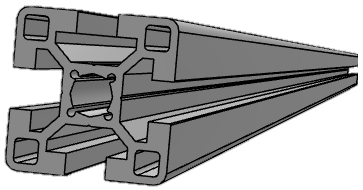
- velocità angolare massima di $10rpm$
- corrente nominale minore di $1.6A$
- coppia massima di 70 $Kg * cm$

# Struttura fisica del sistema 1/2

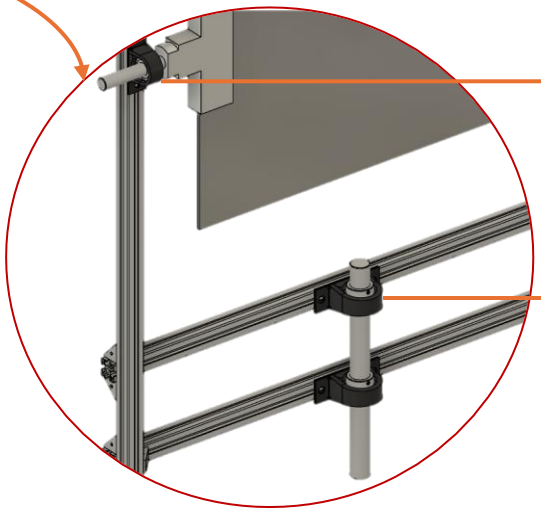
Ho progettato la struttura fisica del sistema attraverso il CAD3d Fusion 360 effettuando un pre-montaggio con gli elementi acquistati che compongono lo scheletro della stuttura.



4 Profilati di alluminio da $90\ cm$
+1 da $70\ cm$



Specchio $50 * 60\ cm$

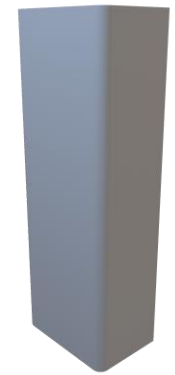


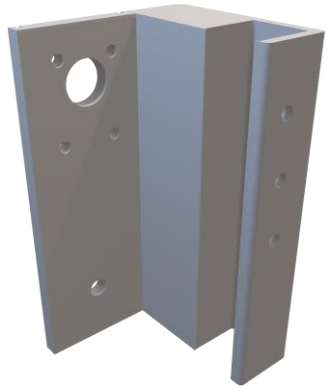
2 alberi in alluminio cavi $\phi 25$

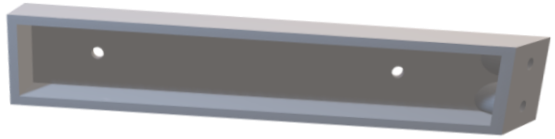
3 Supporti cuscinetto

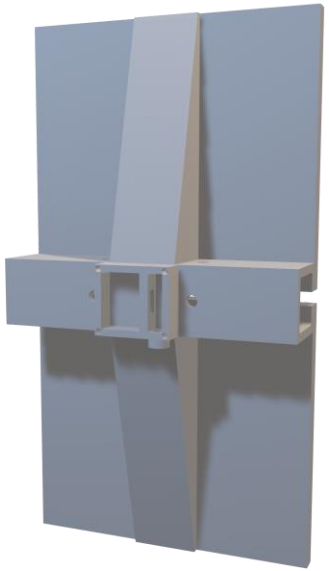
# Struttura fisica del sistema 2/2

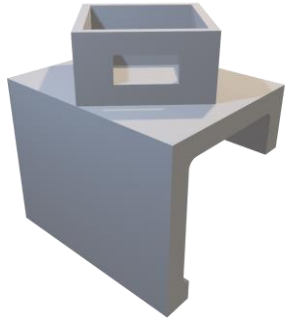
Gli elementi sviluppati in CAD e stampati hanno permesso il montaggio dei motori, dei sensori e dello specchio sulla struttura, in più hanno garantito gli accoppiamenti albero-motore.

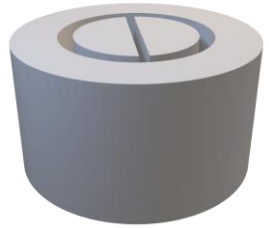












# Filtraggio del rumore 1/2

I dati grezzi ricavati dai sensori risultavano inquinati da interferenze, è stata dunque necessaria l'implementazione all'interno del codice di un filtro digitale passabasso (Digital Low Pass Filter). Ciò è stato possibile sfruttando le caratteristiche del filtro passa basso di ordine 1:

$$y_k = (1-\alpha)y_{k-1} + \alpha u_k$$

$$TC = \frac{1}{2\,\pi\,f_c}$$

$$\alpha = \frac{\Delta T_{camp}}{TC + \Delta T_{camp}}$$

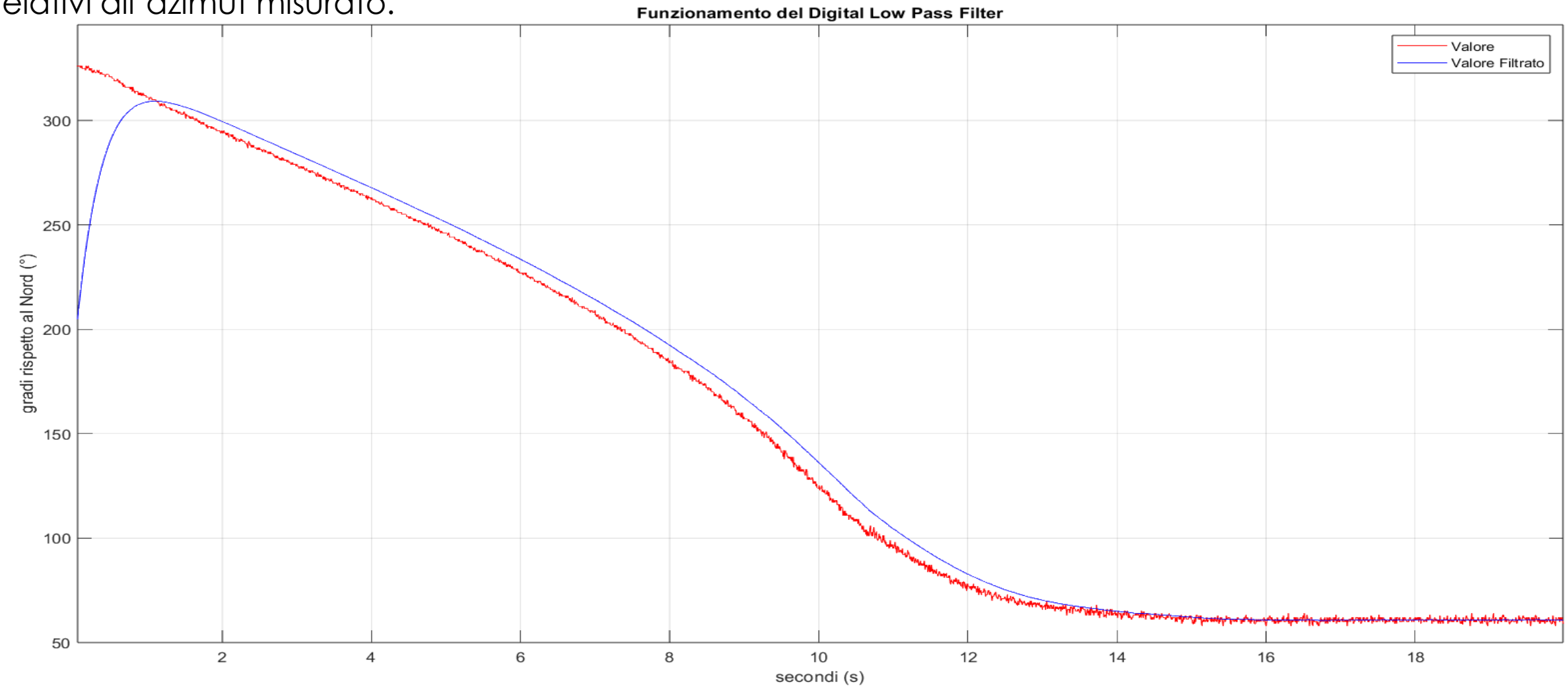
Implementate poi nel codice di Arduino come una classe definita DLPF.h

```h
#ifndef _FILTER_H_
#define _FILTER_H_
#include <math.h>
class LPF{
    private:
        double _y_old;
        double _y;
        double _alfa;
    public:
        LPF(double f_c, double T_s);//Definizione Costruttore
        double update(double x_raw);
        void set_initial_condition (double y_0);
};
LPF::LPF(double f_c, double T_s){//COSTRUTTORE implementazione
    double tau_c= 1.0/(2.0*PI*f_c); //const. di tempo (data frequenza di taglio(_c = cut))
    _alfa = (T_s ) / (tau_c + T_s );
    _y = 0.0;
    _y_old=_y;
}

void LPF::set_initial_condition(double y_0){
    _y=y_0;
    _y_old=_y;
}
double LPF::update(double x_raw) {
    _y = ((1 - _alfa) * _y_old) + ( _alfa * x_raw);
    _y_old = _y;
    return _y;
}
#endif // _FILTER_H_
```

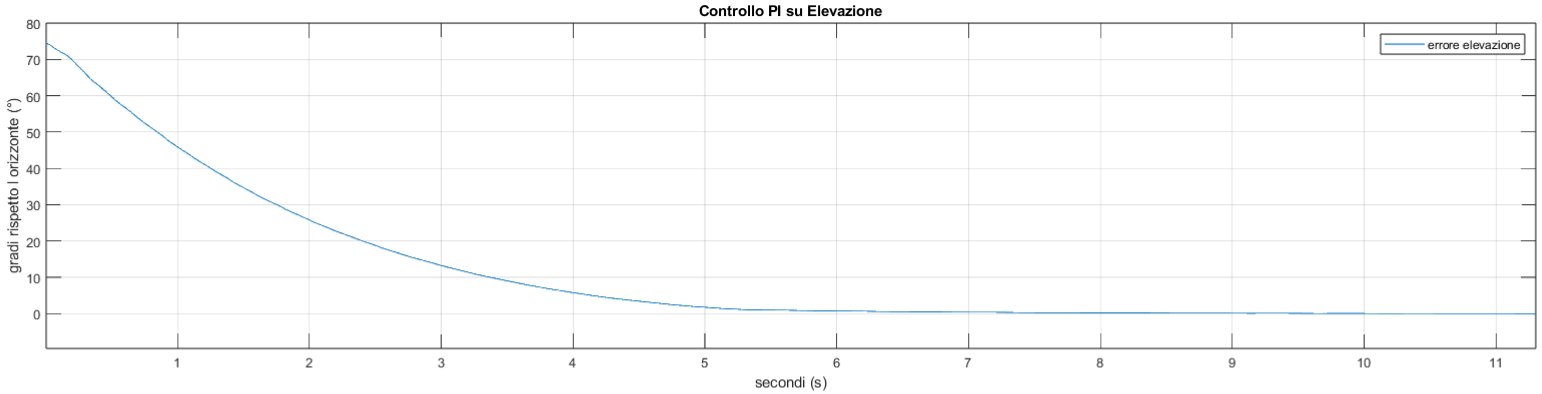
# Filtraggio del rumore 2/2

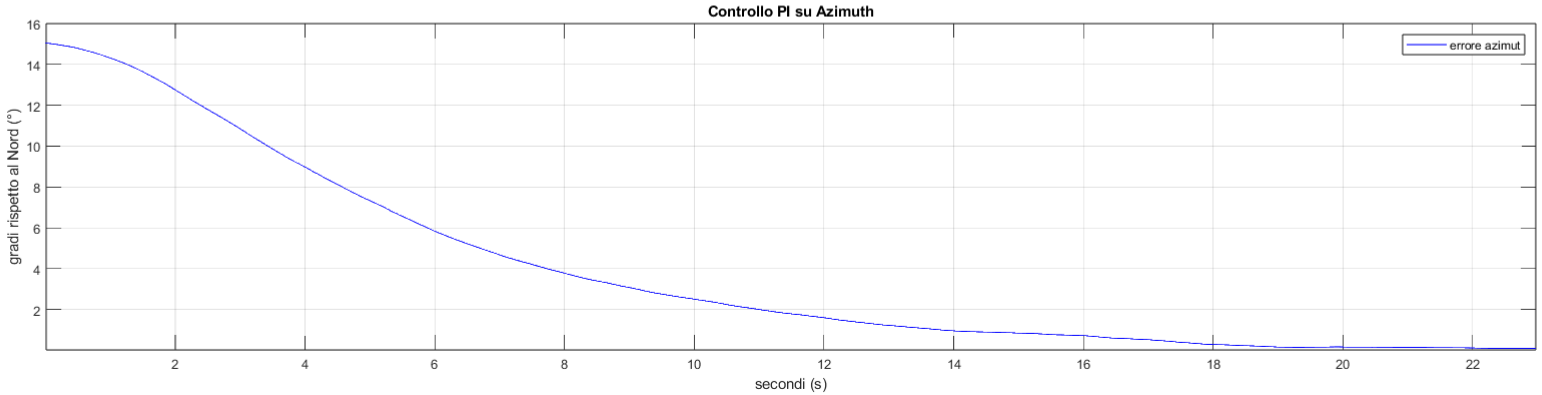
Il funzionamento del filtro può essere visualizzato nel grafico seguente, che mostra i dati grezzi e filtrati relativi all'azimut misurato.



# Controllo di posizione 1/2

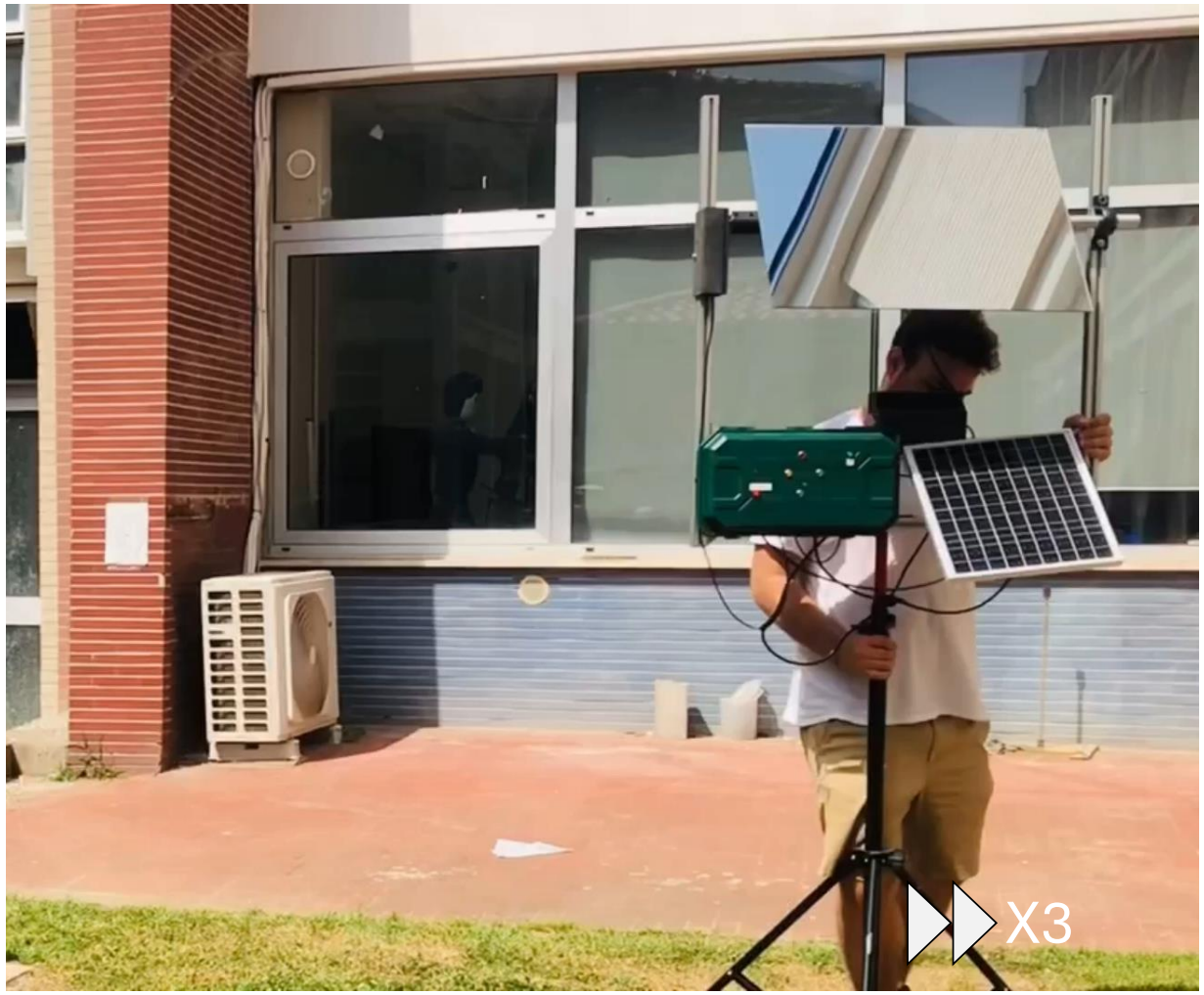
Il controllo in posizione è gestito da un controllore PI implementato all'interno del codice Arduino come classe (PI.h).
Essa prende in ingresso l'errore e i guadagni proporzionali ed integrato per generare un segnale che poi andrà a pilotare la potenza fornita ai motori.
I guadagni sono stati scelti attraverso un processo di trial and error e sono diversi per elevazione e azimut.



# Controllo di posizione 2/2

Il controllo risulta efficace ed il target viene correttamente illuminato dai raggi del sole riflessi dal sistema.

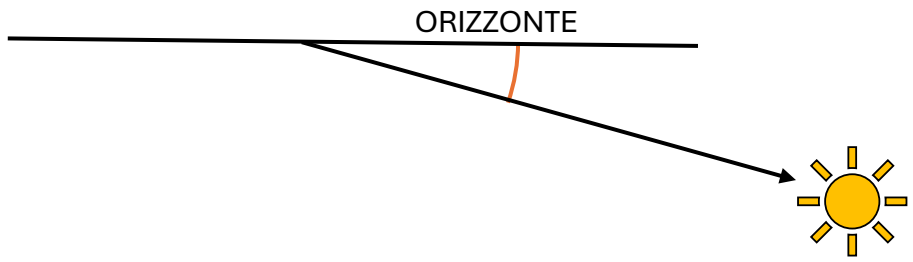


# Modalità Low-Power

Grazie alla libreria LowPower.h il sistema è in grado entrare in una modalità di basso consumo risparmiando energia. Essendo la variazione delle coordinate solari molto lenta, il codice prevede di richiamare questa funzione in maniera periodica durante l'arco della giornata, al fine di ridurre il consumo.

È stata sviluppata inoltre una modalità notturna, che permette al sistema di entrare automaticamente in modalità Low-Power quando ottiene una lettura negativa dell'elevazione solare, essendo quest'ultima possibile solo durante le ore notturne.

```
92      while (el_sun<0){
93        LOW_P.ACTIVE();
94        delay(1000);
95        calcHorizontalCoordinates(utc, latitude, longitude, az_sun, el_sun);
96      }
```



# Grazie per l'attenzione