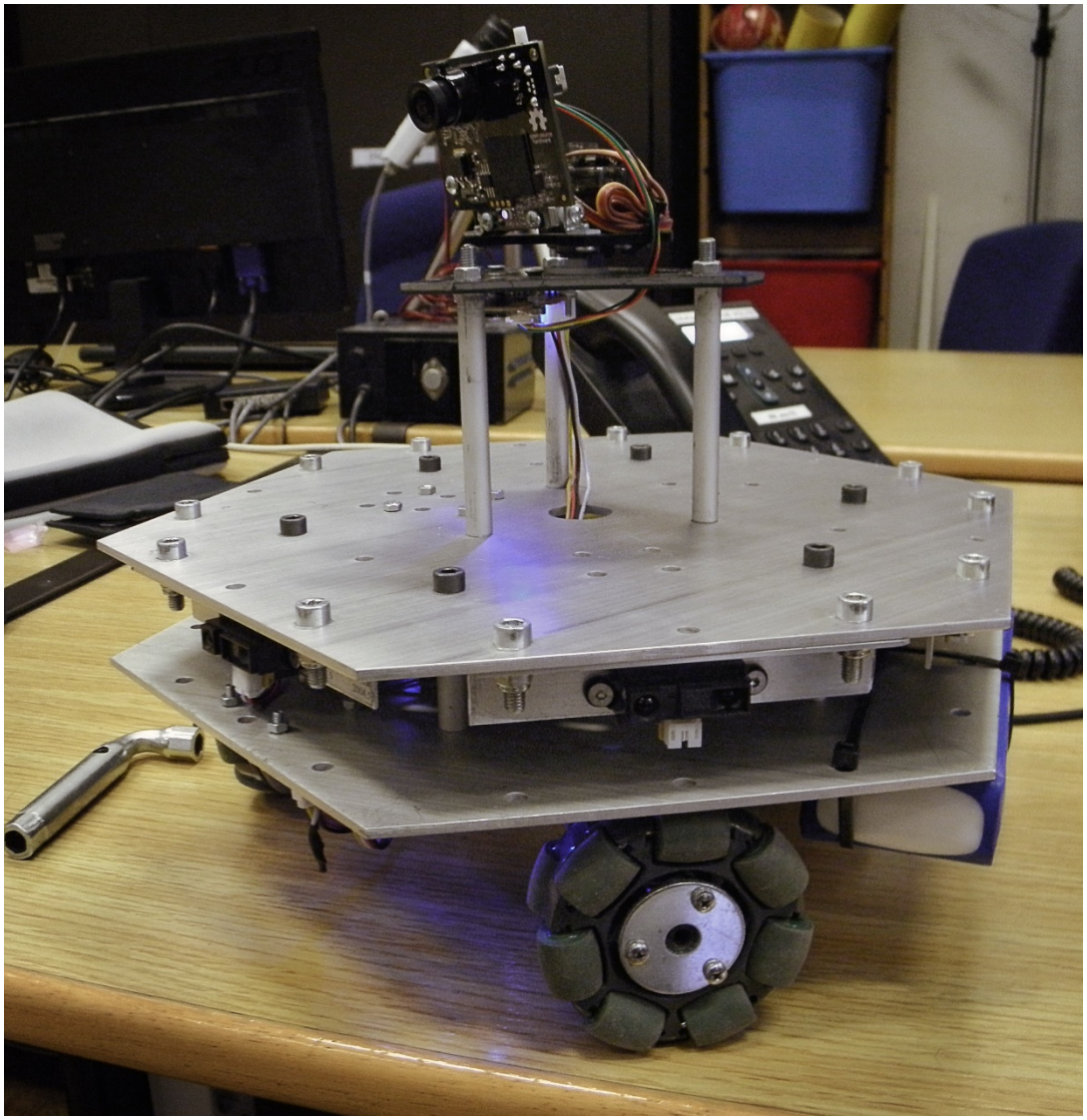
POLITECNICO DI MILANO

ARTIFICIAL INTELLIGENCE and ROBOTICS LABORATORY

# Progetto TriskarPixy



*Dipartimento di Elettronica, Informazione e Bioingegneria*

*Master degree in Computer Engineering*

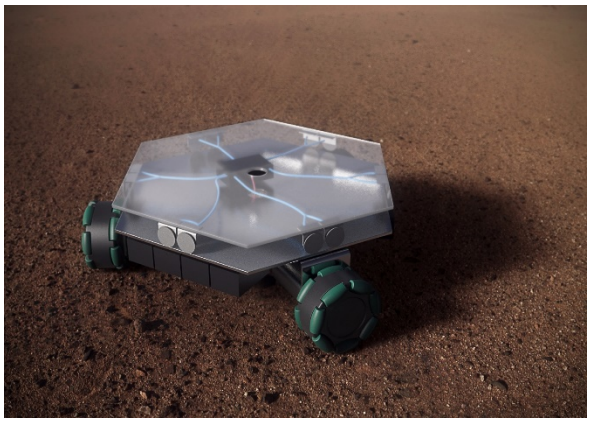
*Alessandro De Angelis* *Andrea Sorbelli*

# Sommario

# 1. Introduzione

**TriskarPixy è un progetto di integrazione di una telecamera Pixy , capace di riconoscere oggetti di vari colori e di inviare i dati su diversi canali di trasmissione, sul progetto Triskar , un robot omnidirezionale basato sul sistema r2p , creato nel laboratorio AirLab del Politecnico di Milano. Il robot è capace di cercare, riconoscere e di inseguire oggetti, oltre ad avere un sistema di collision-avoidance che utilizza sensori ad infrarossi.**

# Fase Preliminare: Progettazione

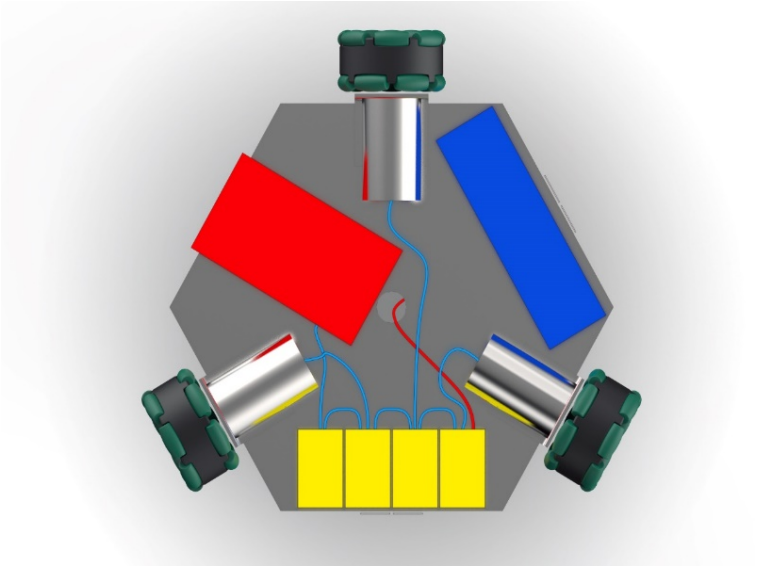


**Nella fase preliminare abbiamo progettato al computer la disposizione delle componenti sulla base in metallo. Inizialmente il progetto doveva utilizzare ROS e una scheda Raspberry PI che è stata successivamente eliminata.**

**Abbiamo optato per un sistema a livelli, in modo da poter rendere il robot facilmente modificabile con l'aggiunta di componenti con caratteristiche e funzioni diverse. Per quanto questa scelta sia utile a progetto ultimato, si è dimostrata problematica durante la fase implementativa. Essendo questo un prototipo e data la nostra scarsa esperienza con il sistema operativo siamo stati obbligati più volte a smontare e ricostruire il robot per modificare i componenti interni.**

**Sfruttando la praticità delle schede r2p è stato possibile collegare facilmente ogni livello in serie con il successivo.**

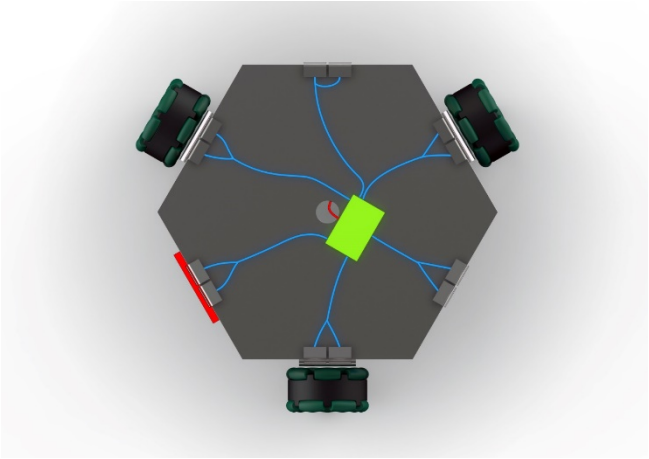
## 2.1 1° Livello: Movimento



**La piastra metallica ha un lato di 15cm e forma esagonale. A 3 lati dell'esagono sono fissate le ruote di 7cm di diametro con attacchi ad L di 4 cm. I motori cilindrici occupano circa 6,5 cm della base. Le 3 zone create dai motori sono occupate da:**

- *Una batteria (di colore blu) di dimensioni 154 x 27 x 44mm (Turnigy nano-tech 5000mah 3S 40~80C).*
- *Un blocco di 3 Schede per i motori (di colore giallo). Dal blocco un filo collegherà il 1° livello con quelli successivi (il filo è evidenziato in rosso).*

### 2.2 2° Livello: Anti-Collisione



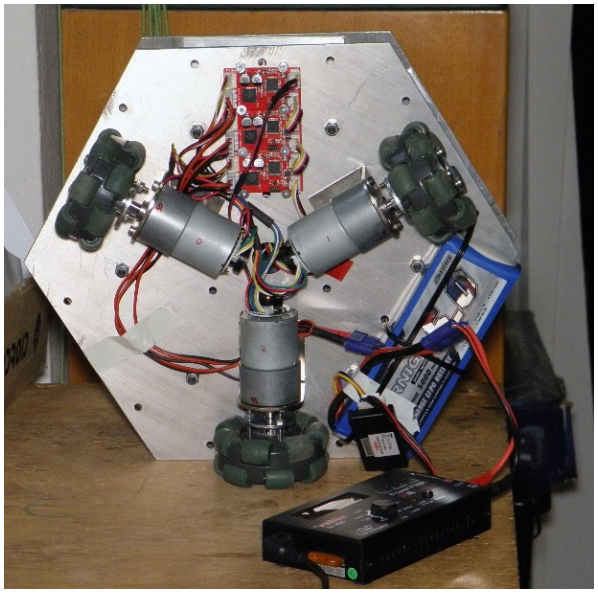
Alla base è fissata centralmente la scheda (di colore verde chiaro) adibita alla gestione dei sensori a infrarossi che sarà collegata al modulo sottostante.

Dei ganci ad L fissano i sensori a infrarossi su ogni lato della base. In fase preliminare abbiamo pensato di montare i ganci il più internamente possibile permettendo una copertura maggiore.

### 2.3 3° Livello: Telecamera

Questo Livello, che deve essere per ovvi motivi il livello più alto del robot, sostiene la telecamera e la relativa scheda. La telecamera è alloggiata su una base sopraelevata per rendere più facile l'individuazione degli oggetti.

## Fase 2: Realizzazione

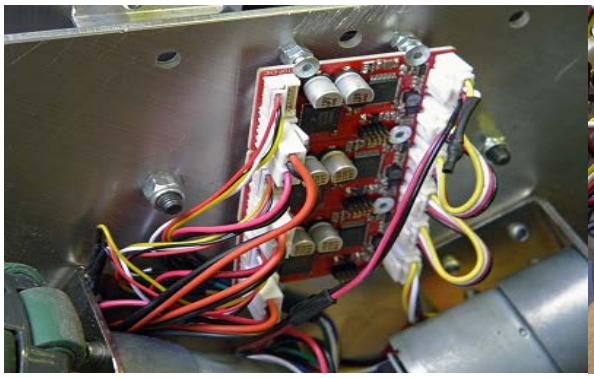


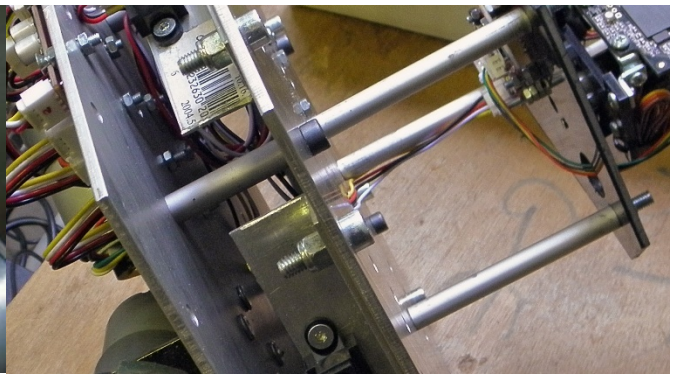
In fase realizzativa sono state apportate alcune modifiche al progetto iniziale.

La scheda Raspberry è stata eliminata, sostituita da un sistema hardware modulare: r2p.

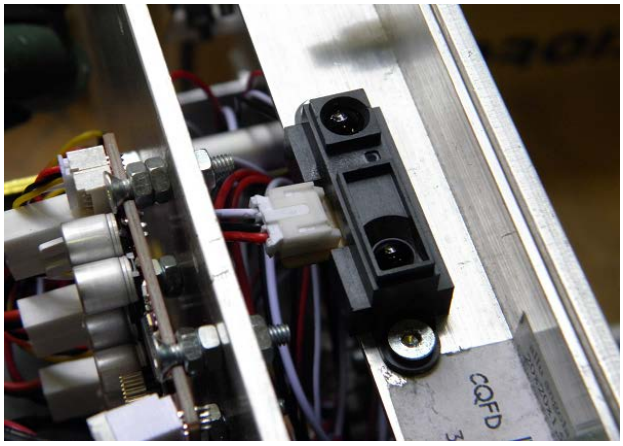
I motori sono stati alloggiati più internamente permettendo un minore ingombro e una più facile gestione delle collisioni, a scapito degli spazi sulla base del robot con il necessario spostamento della batteria che è stata alloggiata leggermente esterna, senza però provocare problemi nel movimento del robot.

Eliminata una scheda è stato possibile alloggiare le schede ruotate rispetto al progetto iniziale, rendendo più facili i collegamenti.



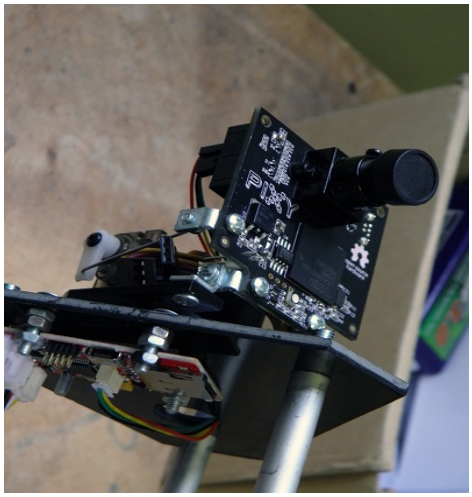


Per rendere più facile la realizzazione e la modifica del robot i livelli sono stati distanziati con un sistema a vite e bullone autobloccante e dei distanziali in alluminio. In maniera similare le schede sono state fissate alle basi con dei distanziali e fissate con dei bulloni, sfruttando le guide per le viti presenti sulle schede stesse.



I sensori a infrarossi sono stati fissati alla seconda piastra con dei ganci ad L ma abbiamo abbandonato l'idea di inserirli più internamente per non compromettere le prestazioni dei sensori.

A causa di una incompatibilità tra i sensori a infrarossi e le scheda che li gestisce non è stato possibile collegare più di due sonar contemporaneamente al sistema. Abbiamo quindi optato per l'utilizzo del solo sensore a infrarossi corrispondente alla parte anteriore del robot, essendo nel nostro caso gli altri superflui: il robot omnidirezionale può ruotare su se stesso durante la ricerca degli oggetti e procede solamente in avanti durante l'inseguimento. Di conseguenza il solo sonar anteriore è sufficiente.



La telecamera è fissata ad una base in plexiglas nero che è collegato alla piastra sottostante con 3 distanziali. Al di sotto della base è presente la scheda che riceve i dati dalla telecamera e che permette la comunicazione del sistema con il computer tramite USB. La telecamera Pixy possiede un sistema di movimento che però non è stato possibile utilizzare a causa di incompatibilità con il sistema. Abbiamo sostituito il movimento della telecamera con la rotazione sul proprio asse del robot, ottenendo il medesimo risultato.

# Fase 3: Programmazione

**Abbiamo utilizzato del tempo per comprendere il funzionamento del sistema studiando le librerie già sviluppate e forniteci. In particolare i nodi dei motori e la Shell di controllo USB erano già presenti e sono state trasferite sulle schede.**
**Una volta compresa la gestione dei nodi delle schede e i relativi publisher e subscriber nella gestione dei messaggi, abbiamo iniziato la realizzazione del codice mancante.**

## 3.1 Scheda Sensori a infrarossi

**Abbiamo deciso di predisporre il codice per gestire tutti i sensori di collisione nonostante non siano utilizzati nel nostro caso. Il messaggio "*ProximityMsg*" contiene un buffer di 8 interi contenenti il valore che proviene dai sensori a infrarossi.**

**Il nodo converte i valori utilizzando il comando adcConvert e ne pubblica il risultato utilizzando il topic *proximity*.**

## 3.2 Telecamera Pixy

**La realizzazione del nodo che gestisce la telecamera è quello a cui abbiamo dedicato più tempo. La Telecamera trasmette dati al sistema attraverso una comunicazione seriale. Per sapere quando effettivamente la trasmissione ha inizio il sistema riceve due volte consecutivamente la parola da 32bit "*0xaa55*". Avvenendo la trasmissione in caratteri esadecimali da 16 bit il nodo Pixy esegue la funzione GetStart( ) che ritornerà il valore 1 quando la sincronizzazione sarà ultimata.**

```
//return 1 when pixy is in syncro
int getStart(void)
{
        uint16_t w, lastw;

        lastw = 0xffff; // some inconsequential initial value

        while(1){
                w = getWord();
                if (w==0 && lastw==0)
                        return 0; // in I2C and SPI modes this means no data, so return immediately
                else if (w==PIXY_START_WORD && lastw==PIXY_START_WORD)
                {
                        g_blockType = NORMAL_BLOCK; // remember block type
                        return 1; // code found!
                }
                else if (w==PIXY_START_WORD_CC && lastw==PIXY_START_WORD)
                {
                        g_blockType = CC_BLOCK; // found color code block
                        return 1;
                }
                else if (w==PIXY_START_WORDX) // this is important, we might be juxtaposed
                        sdGetTimeout(&SD3, MS2ST(100)); // we're out of sync! (backwards)
                lastw = w; // save
        }
}
```

Dopo i caratteri di sincronizzazione i blocchi seguenti contengono i dati relativi alle rilevazioni della telecamera nell'ordine: checksum, signature, x (centro dell'oggetto sull'asse x), y (centro dell'oggetto sull'asse y), height, width.

```
uint16_t tempValues[6],w,sum=0;
tempValues[0]=checksum;
for(int i=1;i<6;i++){
        tempValues[i]=getWord();
        sum = sum + tempValues[i];
}
if(checksum==sum)
{
        if (pixy_pub.alloc(msgp)) {
                //order : checksum,signature,x,y,height,width
                setPixyMsg(msgp,tempValues[0],tempValues[1],tempValues[2],
                                tempValues[3],tempValues[4],tempValues[5]);
                pixy_pub.publish(*msgp);
        }
}
```

Il valore di checksum permette di controllare che i valori siano corretti, in quanto deve essere uguale alla somma dei valori restanti.

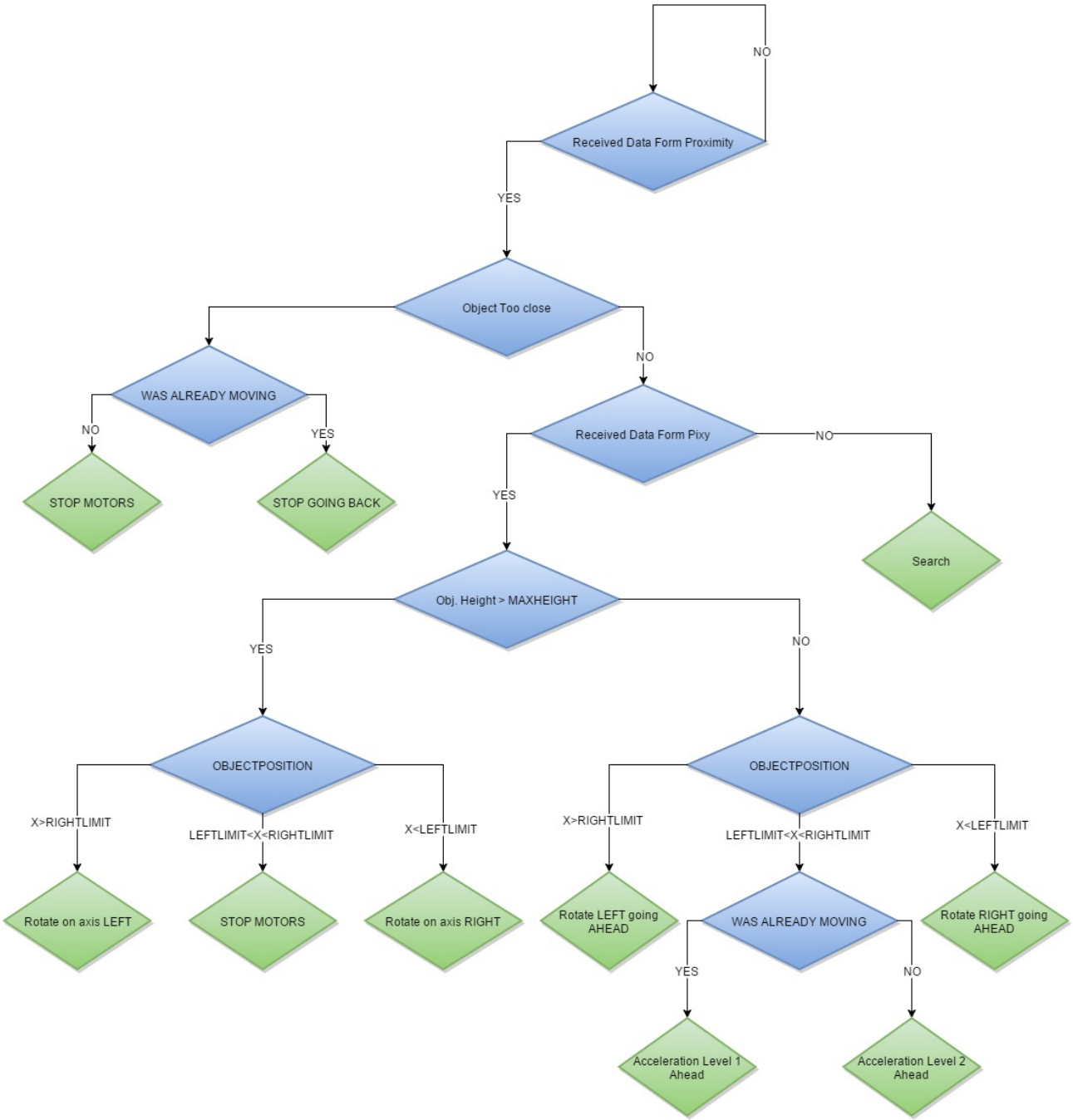
Se il publisher riesce ad allocare il messaggio di tipo "*PixyMsg*" ne salva i parametri con la seguente funzione e i nodi che ne hanno bisogno potranno leggere le informazioni contenute nel messaggio.

```
//set a message of type PixyMsg
void setPixyMsg(r2p::PixyMsg * msgp,int checksum,int signature,int x,int y,int width,int height){
        msgp->checksum =checksum;
        msgp->signature=signature;
        msgp->x = x;
        msgp->y = y;
        msgp->width = width;
        msgp->height = height;
}
```

È possibile visualizzare i messaggi inviati tramite il comando "*pixy*" sulla Shell.

## 3.3 Sistema Di Inseguimento

***Il sistema di inseguimento implementato sul robot è descritto dal diagramma sottostante.***




***Il robot controlla se i sensori (nel nostro caso solo il sensore anteriore) rilevano un ostacolo e se è questo il caso i motori vengono fermati. Per diminuire il tempo di frenata del robot è stato implementato un controllo che verifica se il robot si sta muovendo quando l'ostacolo viene rilevato e nel caso imposta un set-point negativo per i motori.***

Se nessun ostacolo viene rilevato il programma prosegue con un controllo sui dati ricevuti dalla Pixy. La Pixy pubblica un nuovo messaggio solo nel caso in cui un oggetto venga riconosciuto, per questo è stato implementato un time-out che dopo alcuni secondi senza la ricezione di un messaggio fa ruotare il robot sul proprio asse per cercare l'oggetto.

Se è presente un messaggio dalla Pixy il controllo successivo è fatto sull'altezza dell'oggetto. Inizialmente questo controllo era usato al posto dei sensori a infrarossi per evitare una collisione, ma è stato modificato per poter centrare il robot sull'oggetto. Se l'altezza limite è stata raggiunta (il robot è abbastanza vicino all'oggetto) viene verificato il centramento dell'oggetto e nel caso questo non sia all'interno della finestra prevista il robot viene fatto ruotare leggermente per correggere la posizione.

Se l'altezza limite non è stata raggiunta viene sfruttato il parametro x del messaggio della Pixy per verificare in quale posizione si trova l'oggetto rispetto al robot e il comando appropriato viene inviato ai motori per raggiungerlo.

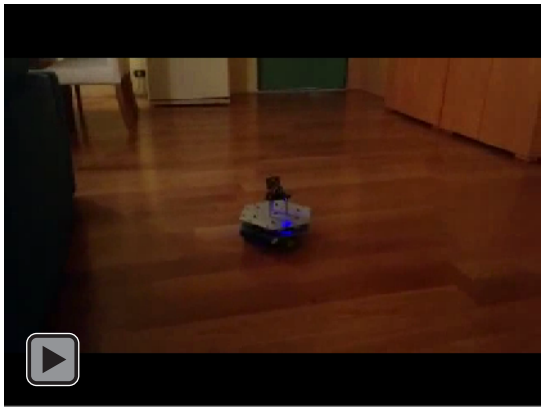
Un'ultima accortezza che è stata aggiunta è stata la distinzione all'interno del movimento in avanti del robot tra robot già in movimento e robot fermo.

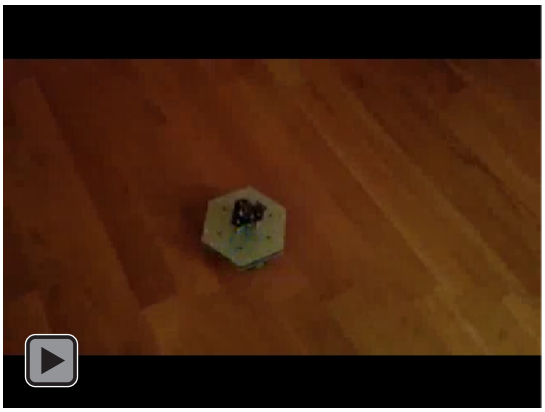
## Fase 4: Test

Per quanto riguarda i sonar valori ottenuti sono meno precisi e variabili di quanto ci aspettassimo ma dopo alcuni test abbiamo trovato un valore corrispondente a una distanza sufficiente per permettere al robot di evitare una collisione.

Ci siamo accorti da diversi test che impostando un set-point troppo alto per il movimento il robot era troppo veloce e questo non permetteva uno stop efficace nel caso di presenza di un ostacolo, ma allo stesso tempo un set-point troppo basso portava ad avere una partenza lenta e poco fluida soprattutto nel caso di piccole distanze da percorrere. Abbiamo risolto il problema imponendo due diversi set-point nelle diverse situazioni. L'utilizzo di sonar più precisi permetterebbe di utilizzare velocità molto più elevate (sfruttiamo molto meno della velocità massima raggiungibile dal robot).

Purtroppo uno dei motori anteriori (rispetto alla telecamera) si è leggermente rovinato e una ruota è più lenta rispetto alle altre. Per questo motivo il robot tende, mentre si sposta in avanti, a ruotare leggermente e questo lo obbliga a continui aggiustamenti della traiettoria, come è possibile vedere dai video.





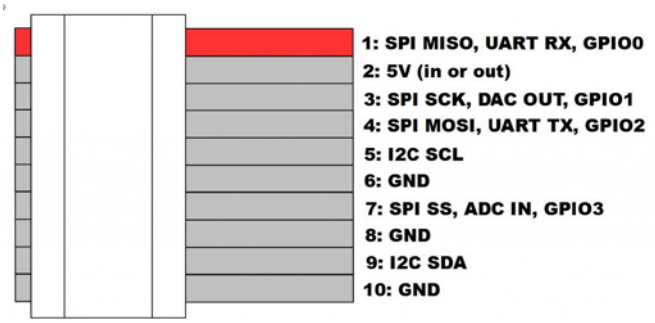
# Guida

## Collegamenti Elettrici

Le schede r2p che costituiscono la base del sistema del robot sono collegate in serie con un terminatore finale collegato all'alimentazione data da una batteria da 11,1 Volt.

Ogni motore è collegato ad una scheda r2p dedicata che funziona sia da controller che da sorgente energetica (fa da tramite tra la batteria e il motore.

La Pixy è collegata ad una scheda r2p attraverso un collegamento seriale che sfrutta il protocollo UART. I collegamenti sono fatti seguendo la figura a fianco.




I sensori a infrarossi sono collegati a una scheda r2p dedicata che supporta fino a 8 sensori.

La stessa scheda utilizzata per il collegamento della Pixy fornisce anche il supporto USB per il collegamento ad un computer.

## Insegnare un oggetto alla Pixy

I metodi possibili sono due:

Utilizzare il software Pixymon fornito dagli sviluppatori Pixy al link: http://cmucam.org/projects/cmucam5/wiki/Installing_PixyMon_on_Linux

Utilizzare il metodo rapido: tenere premuto il pulsante della Pixy fino a fermarsi sul led del colore più simile a quello dell'oggetto da riconoscere, rilasciare il pulsante e posizionare l'oggetto di fronte alla pixy in modo che il led assuma un colore abbastanza intenso della tonalità dell'oggetto, premere una volta il pulsante della pixy mantenendo l'oggetto fermo davanti ad essa.

## Connessione USB

Il robot può essere collegato ad un computer tramite una porta micro-usb presente sulla scheda utilizzata anche dalla Pixy che è comodamente raggiungibile, essendo posizionata nella parte inferiore della piastra il plexiglass rialzata.

Una volta collegato il robot ad un computer, da un terminale attraverso il comando "dmesg" è possibile conoscere il nome che è stato assegnato al robot (solitamente /dev/ttyACM0).

Utilizzando CuteCom (un terminale per la comunicazione seriale) è possibile inizializzare una comunicazione con il robot ed avere a disposizione la shell per impartire i comandi che abbiamo implementato.

Le impostazioni da utilizzare con CuteCom sono le seguenti:

Device: il nome ottenuto tramite il comando "dmesg" su un terminale;

Baud rate: 19200; Data bits: 8; Stop bits: 1;

Parity: none; Handshake: nessuno;

Open for: Reading e Writing;

Hex output: no; Line end: CR, LF;

## Comandi da Shell

I comandi che possono essere usati dalla shell attraverso CuteCom sono:

*Autofollow,* Mette Triskar in modalità inseguimento automatico (il computer può essere tranquillamente scollegato ed il robot continuerà l'inseguimento dell'oggetto).

*Unfollow,* Ferma la modalità inseguimento automatica se è stata precedentemente attivata.

*Pixy,* Mostra l'ultimo messaggio inviato al sistema dalla Pixy.

*Proximity,* Mostra l'ultimo messaggio inviato al sistema dai sensori di prossimità.

*Follow,* Attiva la modalità inseguimento con feedback immediato sulla shell per ogni azione eseguita dal robot.

*Run x y z,* Esegue un movimento con i parametri indicati.

*S,* Ferma i motori.

*E,* Mostra i valori istantanei degli encoder.

*Pidcfg,* Configura i pid.

# Conclusione

Alla fine di questo progetto abbiamo ottenuto un robot che soddisfa i requisiti richiesti e che può essere controllato comodamente dalla shell di comando collegandolo tramite USB ad un computer.

Dopo diversi suggerimenti da parte dei nostri colleghi in laboratorio abbiamo pensato di chiamarlo "The Bull" dato che i test sono stati fatti utilizzando come oggetto da inseguire una scatola rossa e le prime volte il robot la caricava come un toro arrabbiato.

Per qualsiasi domanda è possibile rivolgersi direttamente a noi scrivendo a:

Alessandro De Angelis: alessandro1.deangelis@mail.polimi.it

Andrea Sorbelli: andrea.sorbelli@mail.polimi.it