Emanuele Altamura - 144776

Relazione progetto Internet of Things A.A. 2020/2021

Introduzione

Nell'ultimo decennio grazie allo sviluppo e diffusione di tecnologie open source è aumentato in maniera esponenziale il numero e la qualità dei progetti realizzabili in casa con il contributo di sviluppatori, ingegneri e appassionati. InMoov è uno di questi. Il primo robot umanoide a grandezza umana realizzabile con una stampante 3d.

Essendo open source permette di essere modificato per migliorie e per questo si è deciso di sviluppare il software in ROS, in maniera da poter in seguito usare l'edge-cloud computing per ottenere una interazione con il robot efficiente e il più completa possibile, senza appesantire l'hardware presente.

In attesa di trovare un single board computer ottimale si è scelto di mantenere un computer attaccato al robot.

La funzione implementata in questo caso è il gioco di sasso carta forbice, il codice è all'interno della repository :
https://github.com/PluggaIndustries/Inmoov-playing_rock_scissor_paper .

SCHEMA DI FUNZIONAMENTO

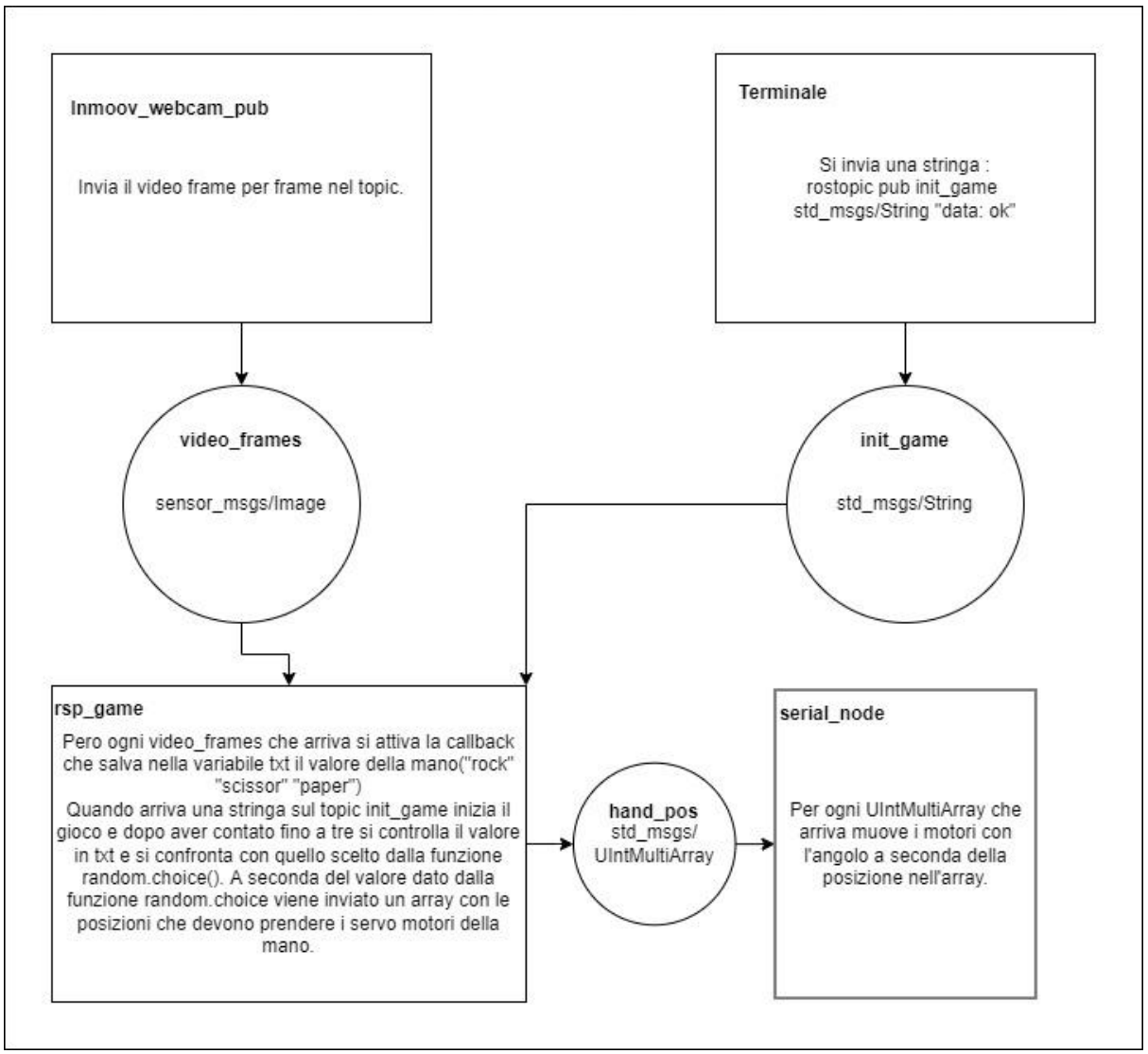
I requisiti

- Pc con Ubuntu 20 e Ros noetic
- Esemplare di InMoov con telecamera, arduino e 5 servomotori

Il funzionamento è il risultato di soli tre nodi per minimizzare il codice e il numero di nodi.

Si attiva il gioco con il comando "roslaunch cv_base RSP_easy.launch" da terminale che attiva i tre nodi: Inmoov_webcam_pub , RSP_easy e serial_ros che permette la comunicazione con arduino con baudrate di 57600 sulla porta ACM0tty.

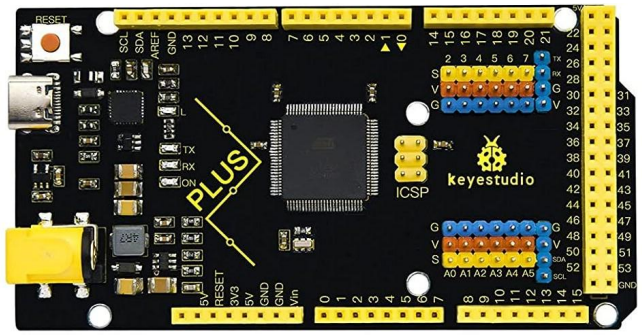
Si invia un comando da terminale e ascoltando il robot si può seguire il gioco nelle sue fasi. Importante è mantenere la mano nel campo visivo del robot in quanto il tracking della mano avrebbe rallentato le prestazioni del gioco. Basta controllare il video sullo schermo che mostrerà anche se la mano sta in posizione di sasso o carta o forbice grazie ad una scritta in sovrimpressione.



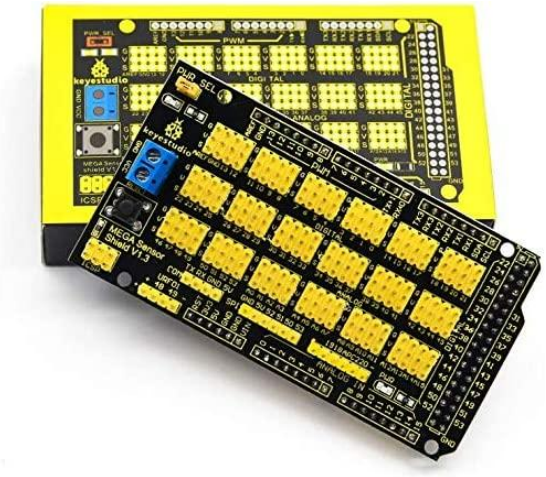
SCELTE IMPLEMENTATIVE

Il robot è umanoide e possiede una LifeCam 3000 nell'occhio collegata al pc. Il microcontrollore è un mega r3 2560 plus della Keyestudio, ed è collegato al pc e con una sensor board 5 servomotori che muovono le dita posizionati nell'avambraccio. Alimentato esternamente con un alimentatore 50W. Uno speaker inserito nella mandibola permette di sentire la voce del robot direttamente da lui dando un senso di realismo.

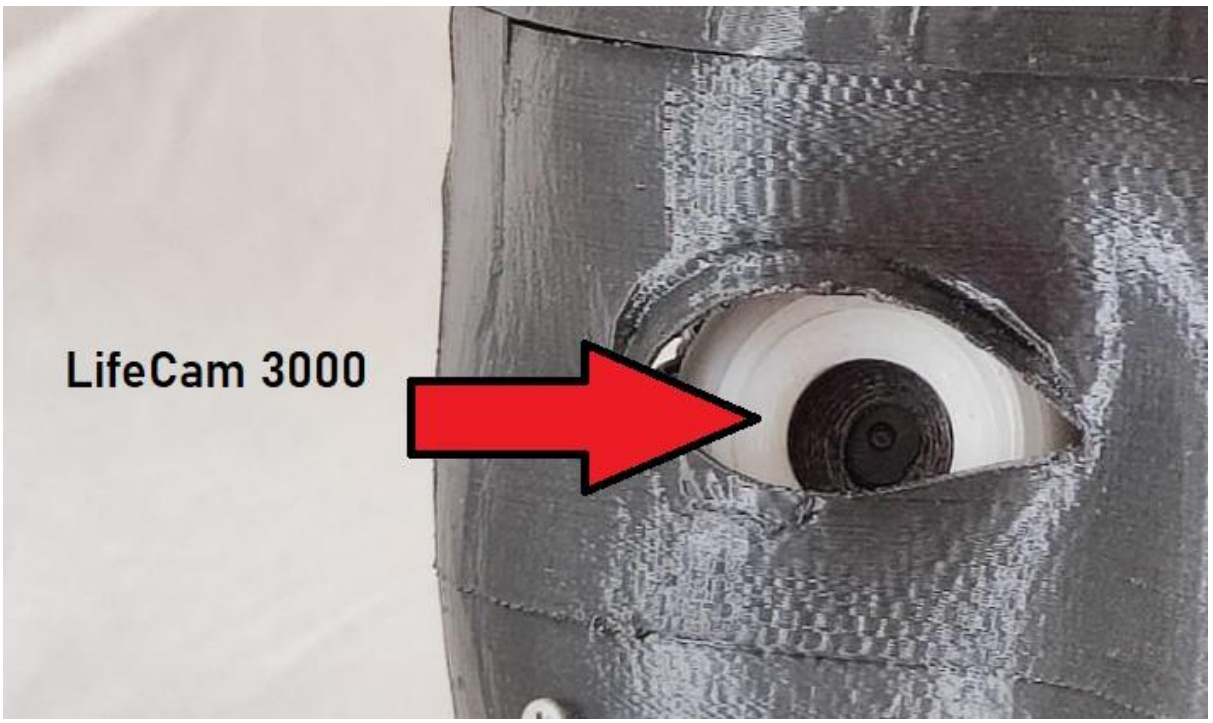
Alimentatore.



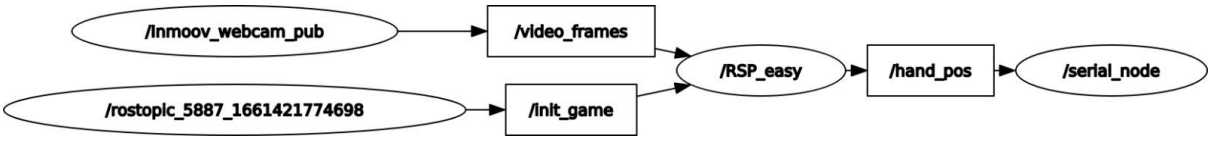
Mega R3



Sensor board.



Per l’implementazione software è utilizzato il sistema operativo ROS Noetic e come linguaggio di programmazione è stato scelto Python.



Inmoov_webcam_pub

Si è scelto di usare il nodo precedentemente realizzato in un'ottica modulare del sistema che prende il video dalla webcam con la funzione di opencv `cv2.VideoCapture(0)` e lo inoltra nel topic `/video_frames` con una frequenza di 10 Hz.
I frame prima di essere inviati vengono convertite dalla funzione `pub.publish(br.cv2_to_imgmsg(frame))` in un messaggio standard di ROS.

RSP_easy

Il nodo è composto da due listener che attivano due differenti callback.
Il primo che è iscritto al topic `/video_frames` e per ogni frames che arriva richiama `callback_hand(data)` che lo converte da Image a immagine utilizzabile per la libreria opencv.
A questo punto il detector di mani della libreria cvzone analizza l'immagine e se presente una mano controlla quali dita sono alzate e se corrispondono ad una mossa di sasso carta forbice e in questo caso viene convertita nella corrispettiva stringa nella variabile fingers_txt.
Il secondo listener, iscritto al topic /`init_game` attiva la funzione `callback_init_game(stringa)` che inizializza il publisher sul topic `hand_pos` dove in base al valore scelto nella funzione `random.choice(game_list)` invia il corrispettivo array di uint16 al microcontrollore che muoverà i servo.
Si continua poi con il conto da uno a tre e dopo il tre si muoverà la mano del robot nella mossa scelta e si attendono tre secondi annunciati dalla frase `"Let me check"` perchè si è notato un ritardo di quasi tre secondi per riconoscere la posizione della mano del giocatore. Dopo questa pausa viene valutata la combinazione delle mosse per determinare il risultato che viene detto dal robot.

Si è scelto di utilizzare la libreria pyttsx3, che sfrutta lo il sintetizzatore vocale espeak nel caso di Ubuntu, nella funzione say() nel nodo rsp_game per comunicare a voce per rendere più autentica l'interazione.

```
110  def say(command):
111
112      engine = pyttsx3.init()
113      engine.say(command)
114      engine.runAndWait()
115      print(command)
```

Serial_node

Il nodo serial_node è un'interfaccia per i dispositivi abilitati alla comunicazione con ROS. Questo nodo automaticamente attiva le iscrizioni e i publisher scritti nel codice del dispositivo.

RSP_easy.ino

Nel microcontrollore, che grazie alla libreria ros.h può interfacciarsi con il serial_node, è presente un listener al topic /`hand_pos` dove vengono inviati array contenenti gli angoli dei servo. E' istanziato un servo per dito e in base al valore nella posizione corrispondere nel array viene mosso.

Future implementazioni:

- Un nodo che gestisce in maniera coerente tutti i movimenti dei motori.
- Un nodo text-to-speech che gestisce in maniera coerente il testo inviatogli.
- Un nodo speech-to-text per gestire l'interazione (es. l'utente dice "voglio giocare" e il robot inizia la sequenza)
- Un nodo che gestisce la face detection e recognition per conoscere le persone e salvare il loro nome e nel caso già conosciute le saluta.