

I'm not a Robotic Arm

Prof. : Alessandro Rizzo

TAs: David Pangcheng Cen Cheng
Andrea Usai

Muratore Luigi s333098

Gennero Giorgia s333099

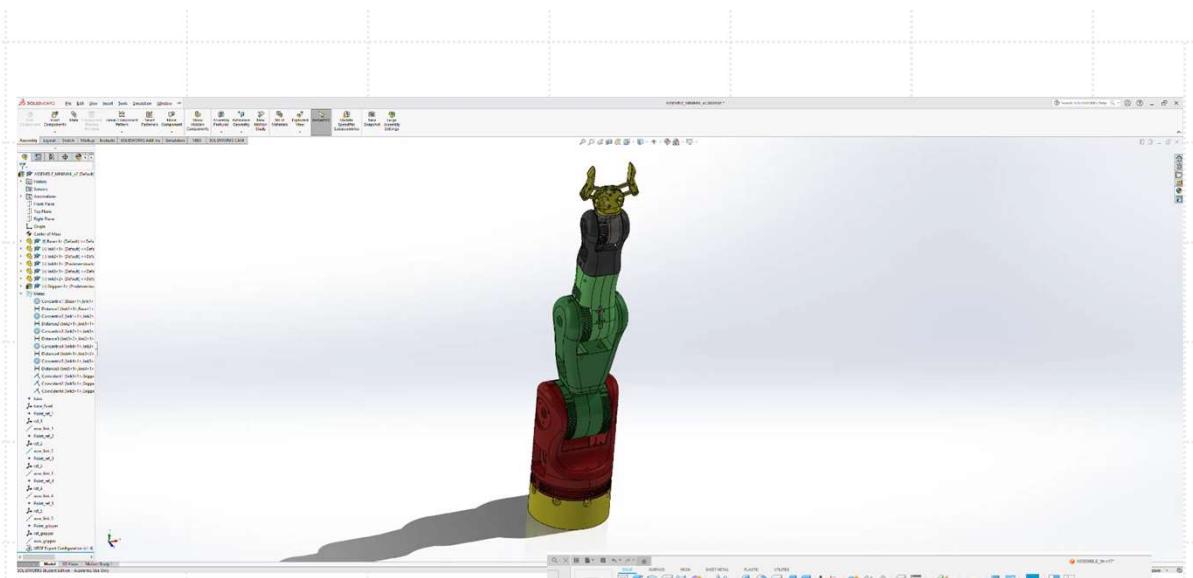
Akbarov Iskandar s329650

Muhammad Fatir Noshab s331898

Moussa Swaidan s334402

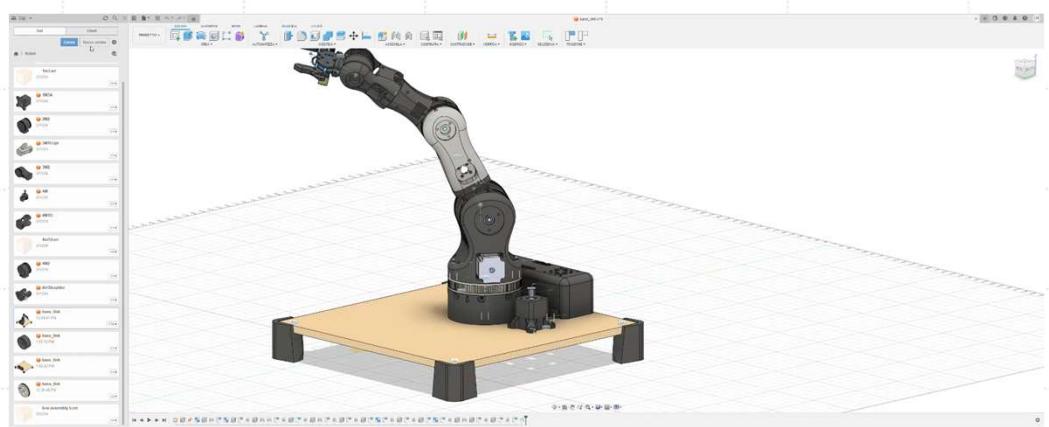
CAD

- SolidWorks
- Autodesk Fusion360

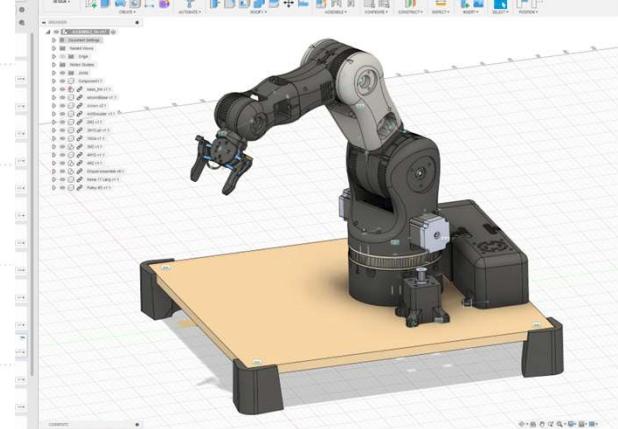


GOAL

- Pick and Place



- Requirements
- Limits



BOM

MOTORS:

- 2x Nema 17
- 1x Nema 14
- 1x Nema 17 1:5
- 2x Nema 23
- 1x Servo 20kg

DRIVERS:

- 6x TB6560

CONTROLLERS :

- Arduino MEGA
- Raspberry Pi 4b

POWER SUPPLY:

- 24V 13A

MOTORS



2x Nema 23



1x Nema 17 1:5



1x Nema 14



1x Servo 20kg

DRIVERS



6x TB6560

POWER SUPPLY



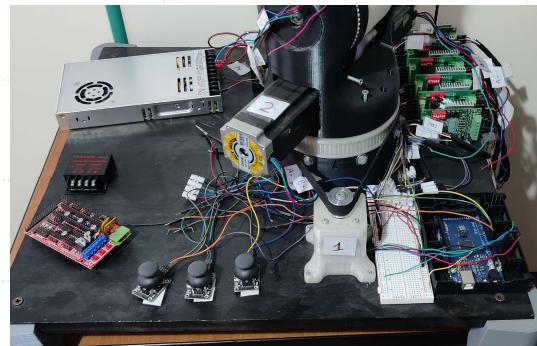
1x 24V 13A

PROGRESS

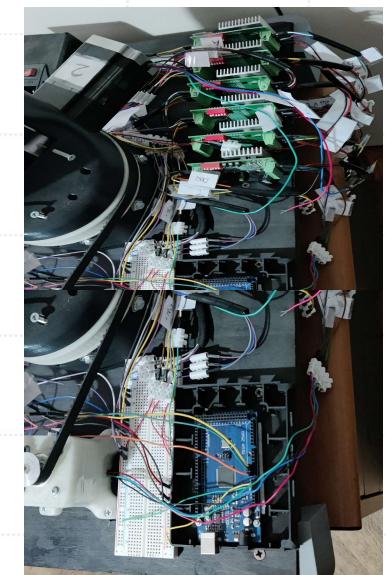
FINAL RESULT



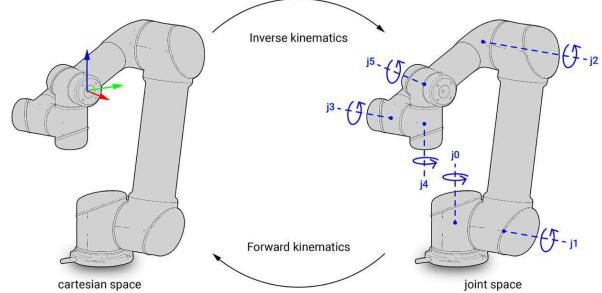
- 3D Printing
- PLA Filament



- Mechanical assemble
- Electronic assemble
- Electrical connections



FORWARD Kinematics



MATLAB computation

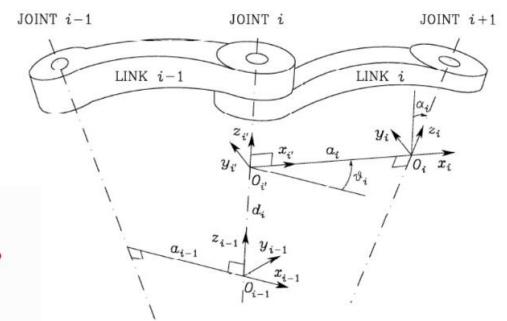
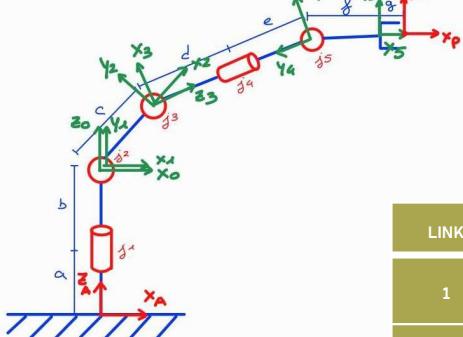
```

MATLAB ROBOT - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
File New Open Save Compare Go To Find Refactor Profiler Analyze Run Run & Advance Run to End Run Step Stop Search Documentation Georgia
Current Folder Editor C:\Users\admin\Dropbox\Georgia Gennaro\02 optional project\MATLAB\ROBOT.m
Name - RVC-MATLAB-main
1 close all
2 clear all
3 format compact
4 clc
5 % KINEMATICS
6 % links
7 % I
8 a = 8;
9 b = 16;
10 c = 22;
11 d = 13;
12 e = 9.5;
13 f = 4;
14 g = 10.5;
15
16 % 3D robot
17 e3D=Tz(q1)*Tc(q2)*ET53.Ry("q3")*ET53.Tz(b)*ET53.Ry("q2")*ET53.Tz(c)*ET53.Ry("q3")*ET53.Tz(d)*ET53.Rz("q4")...
18 *ET53.Tz(e)*Tc(q5)*ET53.Ry("q5")*ET53.Tz(f)*ET53.Tz(g); % 6 links + gripper
19 n = e3D.njoints; % 5 joints
20 e3D.fkine(zeros(1,n));
21 e3D.teach
22
23 % Rigid Body
24 myrobot=et53rb(e3D);
25 myrobot.showdetails;
26
27 return

```

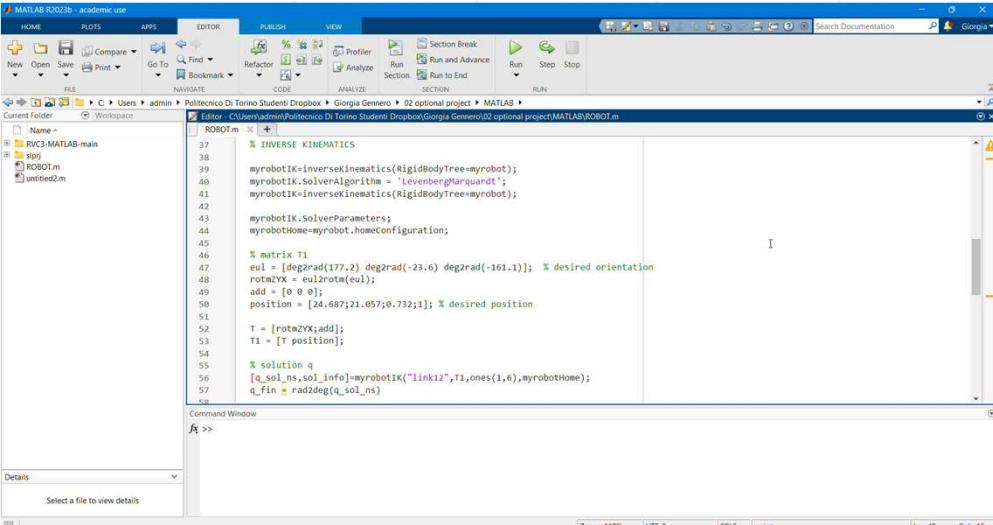
Denavit-Hartenberg convention

- a = link length
- α = link twist
- d = link offset
- θ = joint angle



LINK	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	0	θ_1
2	c	0	0	θ_2
3	0	$\frac{\pi}{2}$	0	θ_3
4	0	$-\frac{\pi}{2}$	$d + e$	θ_4
5	f	0	0	θ_5

INVERSE Kinematics in MATLAB



```
% INVERSE KINEMATICS
myrobotIK.inverseKinematics(RigidBodyTree=myrobot);
myrobotIK.SolverAlgorithm = 'LevenbergMarquardt';
myrobotIK.inverseKinematics(RigidBodyTree=myrobot);

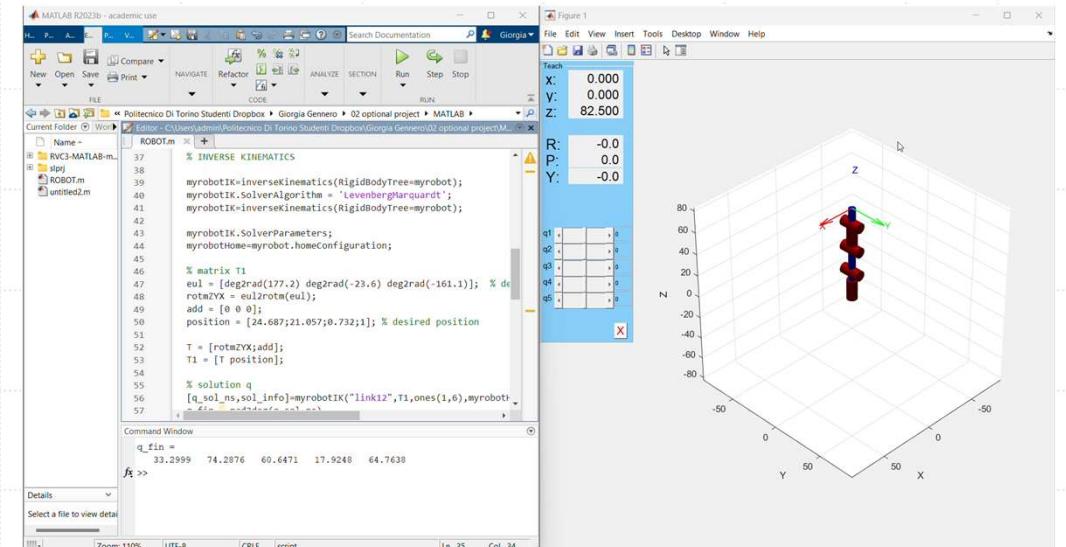
myrobotIK.SolverParameters;
myrobotHome=myrobot.homeConfiguration;

% matrix T1
eul = [deg2rad(177.2) deg2rad(-23.6) deg2rad(-161.1)]; % desired orientation
rotmXY = eul2rotm(eul);
add = [0 0 0];
position = [24.687;21.057;0.732;1]; % desired position

T = [rotmXY;add];
T1 = [T position];

% solution q
[q_sol_ns,sol_info]=myrobotIK("link12",T1,ones(1,6),myrobotHome);
```

- Decentralized control:
independent joint control



- Numerical solution instead of analytical one

URDF – Unified Robot Description File

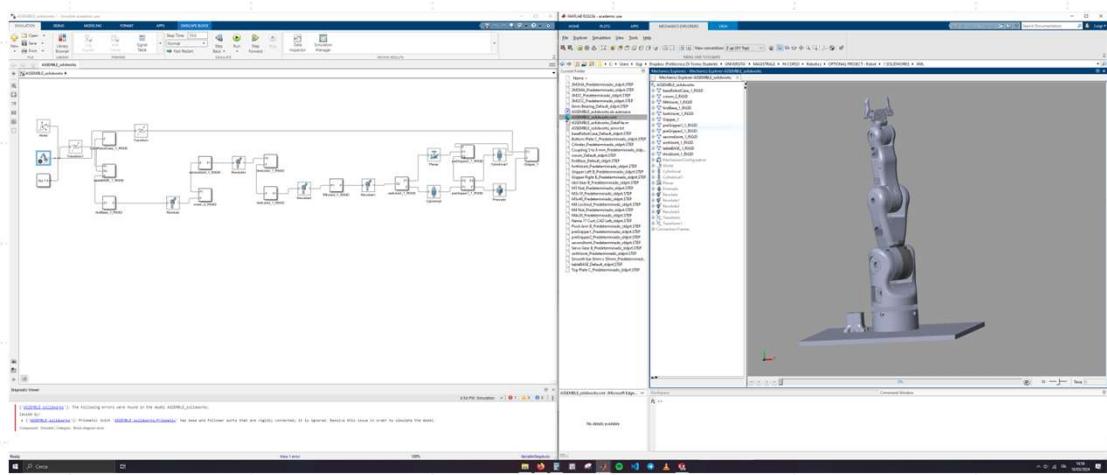
```
final_mod.urdf.x
0 URDF > ROS1 > Gigi > urdf > final_mod.urdf
  2   <robot>
  5     <link>
  6       <visual>
  7         <origin>
  8           <xyz>-0.1411E-05 1.439E-05 0.08307</xyz>
  9           <rpy>0 0 0</rpy>
 10         <mesh>
 11           <value>"1.5994"</value>
 12         <inertia>
 13           <ixx>0.006851</ixx>
 14           <ixy>-3.0652E-05</ixy>
 15           <ixz>4.1042E-07</ixz>
 16           <iyx>0.0083958</iyx>
 17           <iyz>3.7857E-07</iyz>
 18           <izx>0.0063085</izx>
 19         </inertia>
 20       <visual>
 21         <origin>
 22           <xyz>0 0 0</xyz>
 23           <rpy>0 0 0</rpy>
 24         <geometry>
 25           <mesh>
 26             <filename>package://final/meshes/link_1.STL</filename>
 27         </geometry>
 28       <material>
 29         <name></name>
 30         <color>
 31           <rgba>0.79216 0.81961 0.93333 1</rgba>
 32         </color>
 33       </material>
 34     </collision>
 35   </link>
 36
 37 <joint>
 38   <name>"joint_0"</name>
 39   <type>"revolute"</type>
 40   <origin>
 41     <xyz>0 0 0.866</xyz>
 42     <rpy>0 0 0</rpy>
 43   <parent>
 44     <link>"base"</link>
 45   <child>
 46     <link>"link_1"</link>
 47   <axis>
 48     <xyz>0 0 1</xyz>
 49   <limit>
 50     <lower>-1.6</lower>
 51     <upper>1.6</upper>
 52     <effort>100</effort>
 53     <velocity>100</velocity>
 54   </limit>
 55 </joint>
```

Links

- Origin
- Mass
- Geometry
- Inertia
- Material
- Collision

- Extensible Markup Language (XML)
- Robot description
- Visual representation
- Collision model
- Extensible

MATLAB/Simulink Simscape Multibody Link

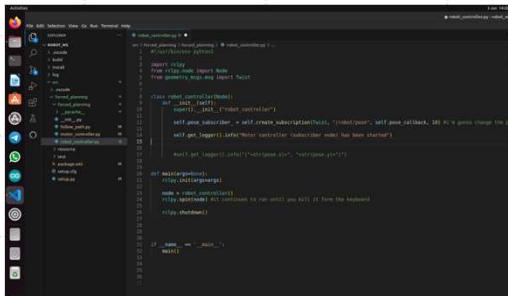


Joints

- Type
- Origin
- Axis
- Parent
- Child
- Limits

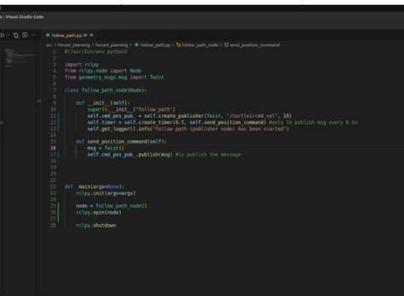
ROS2 - Humble

Subscriber



```
#!/usr/bin/env python3
# coding: utf-8
# subscriber.py
# Author: Lijun Wang
# Date: 2023-06-05
# Description: This script is a ROS2 subscriber that receives joint position commands from a publisher and publishes sensor data back to the publisher.
```

Publisher



```
#!/usr/bin/env python3
# coding: utf-8
# publisher.py
# Author: Lijun Wang
# Date: 2023-06-05
# Description: This script is a ROS2 publisher that sends joint position commands to a subscriber and receives sensor data from the subscriber.
```

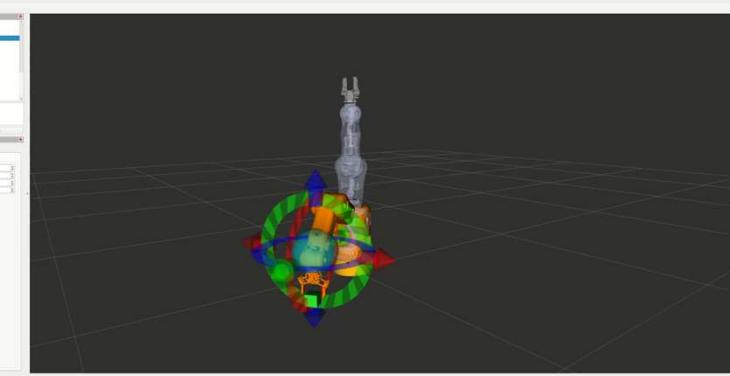
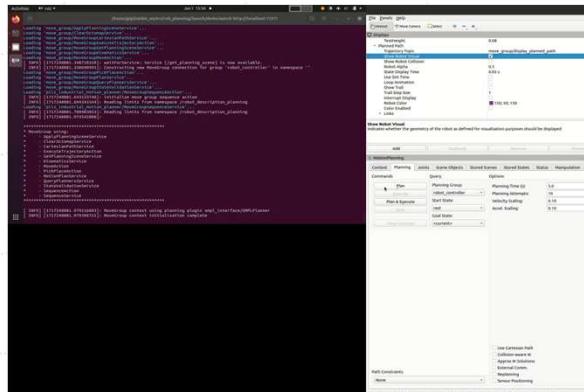
Python

```
# Read pinmap.c
1 import ev3py
2 import GPID as GPID
3 import time
4
5 # Pin configuration for each motor
6 MOTORS = [
7     { 'STEP': 19, 'DIR': 26, 'IN1': 22, 'IN2': 23 },
8     { 'STEP': 20, 'DIR': 26, 'IN1': 24, 'IN2': 20 },
9     { 'STEP': 21, 'DIR': 26, 'IN1': 25, 'IN2': 20 },
10    { 'STEP': 22, 'DIR': 26, 'IN1': 24, 'IN2': 20 },
11    { 'STEP': 23, 'DIR': 26, 'IN1': 21, 'IN2': 19 }
12 ]
13
14 # Pin PINS
15 DIO_PIN_1 = 13
16 DIO_PIN_2 = 14
17 DIO_PIN_3 = 15
18 DIO_PIN_4 = 16
19
20 # GPID pins
21 GPID.setmode(GPID.BCM)
22 for i in range(1, 5):
23     GPID.setup(motor=STEP[i], motor_dir=DIR[i], GPID.DIR)
24
25 def move_motor(STEP_pin, GPID_pin, max_pos, direction, steps, delay=0.001):
26     GPID.setpwm(GPID_pin, 0.5)
27     GPID.setmotor(STEP_pin, motor_dir, motor=STEP[i], GPID.DRV)
28
29     for step in range(steps):
30         GPID.setpwm(GPID_pin, 0.5)
31         GPID.setmotor(STEP_pin, GPID.LWV)
32         time.sleep(delay)
33         GPID.setpwm(GPID_pin, 0.5)
34         GPID.setmotor(STEP_pin, GPID.HDV)
35         time.sleep(delay)
36
37     GPID.setpwm(GPID_pin, 0.5)
38
39     while True:
40         # Double the steps
41         GPID.setmotor(DIO_PIN_1, GPID.LWD) # LOW to enable (comes for many drivers)
42         GPID.setmotor(DIO_PIN_2, GPID.HWD) # HIGH to enable (comes for many drivers)
43         GPID.setmotor(DIO_PIN_3, GPID.LWD) # LOW to enable (comes for many drivers)
44         GPID.setmotor(DIO_PIN_4, GPID.HWD) # HIGH to enable (comes for many drivers)
45
46         user_num = input("Select motor (1 to 5): ")
47         if user_num not in range(1, 6):
48             print("Invalid motor number")
49         else:
50             continue
51
52         direction = input("Enter direction (forward or backwards): ").lower()
53         if direction != "forward" and direction != "backward":
54             print("Invalid direction")
55             continue
56
57         steps = int(input("Enter number of steps: "))
58         if steps < 0:
59             print("Invalid number of steps")
60             continue
61
62         max_pos = GPID.getpos(STEP_pin)
63         direction = GPID.DRV if direction == "forward" else GPID.LWD
64         move_motor(STEP[i], motor_dir=DIR[i], motor=STEP[i], direction=direction, steps=steps)
65
66 except KeyboardInterrupt:
67     print("Program terminated")
```

- Converter angle-steps
- Motors script

Movelt

- OMPL – RRT planning library
- Plan trajectories
- Inverse Kinematics
- Angles



TESTS

Pick and Place with a cube



I'm not a Robot

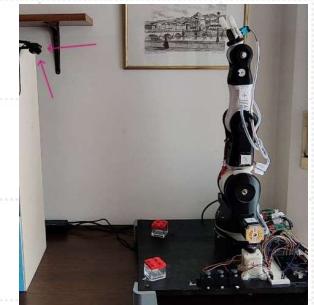
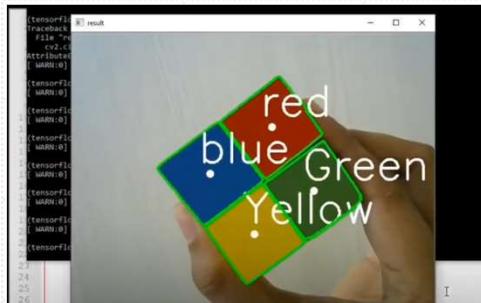


Future Upgrades and Implementations

- Vacuum gripper

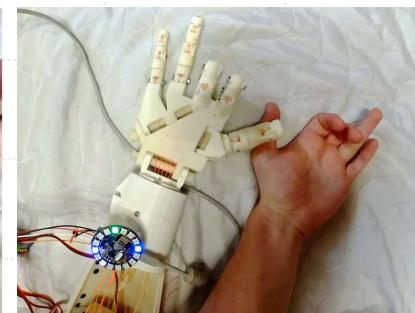


- Stereo camera



- Artificial Intelligence:

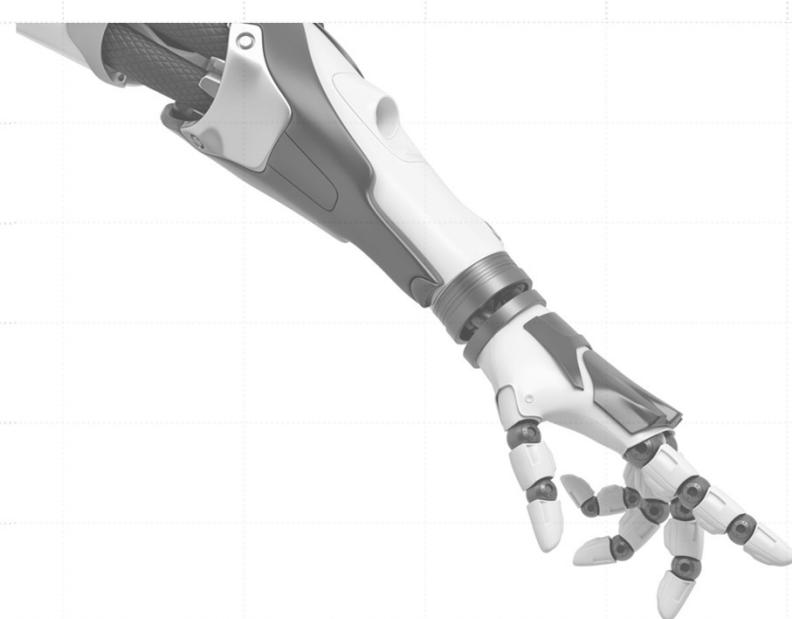
 - Voice control



 - Code generator

 - EMG control

**THANK YOU FOR
YOUR ATTENTION**



I'm not a Robotic Arm