

# I'm not a Robotic Arm

Prof. : Alessandro Rizzo

TAs: David Pangcheng Cen Cheng  
Andrea Usai

Muratore Luigi s333098

Gennero Giorgia s333099

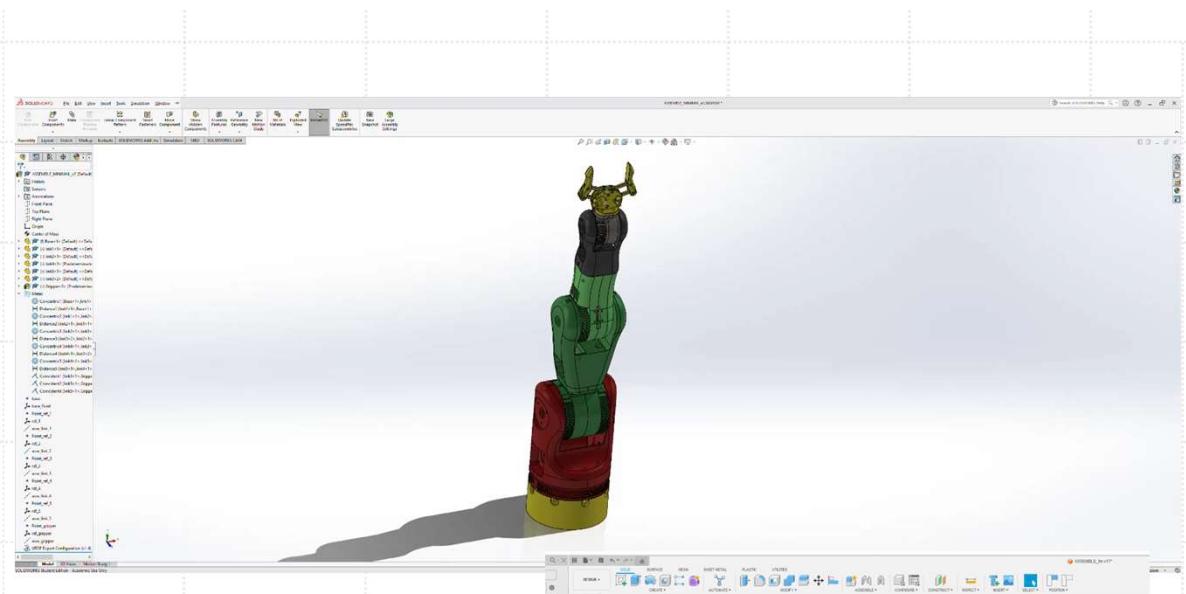
Akbarov Iskandar s329650

Muhammad Fatir Noshab s331898

Moussa Swaidan s334402

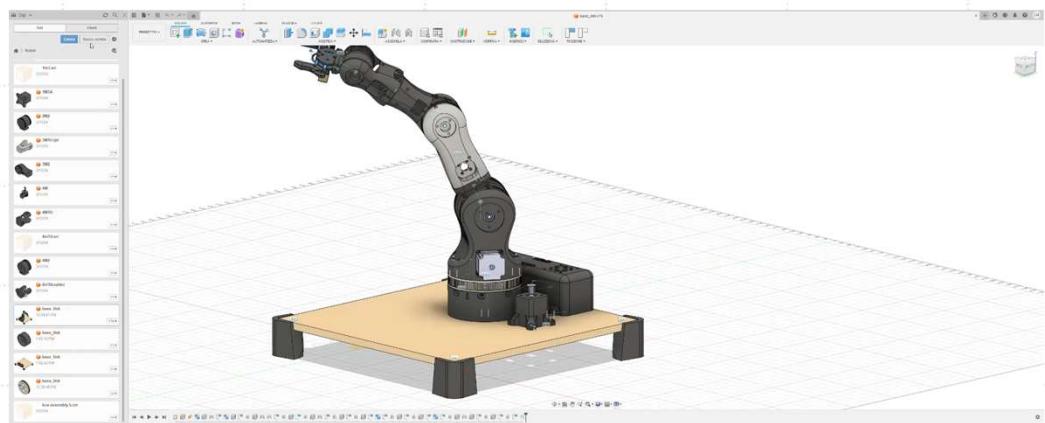
# CAD

- SolidWorks
- Autodesk Fusion360

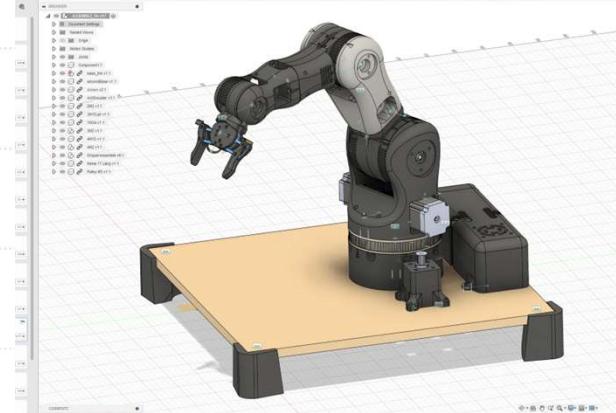


# GOAL

- 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

## POWER SUPPLY:

- 24V 13A

## CONTROLLERS :

- Arduino MEGA
- Raspberry Pi 4b

# MOTORS



2x Nema 23



1x Nema 17 1:5



1x Servo 20kg



2x Nema 17



1x Nema 14

# DRIVERS



6x TB6560

# POWER SUPPLY



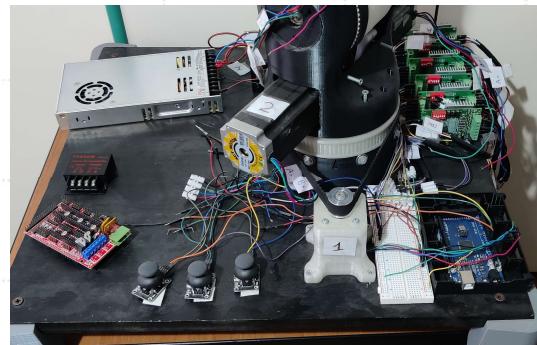
1x 24V 13A

# PROGRESS

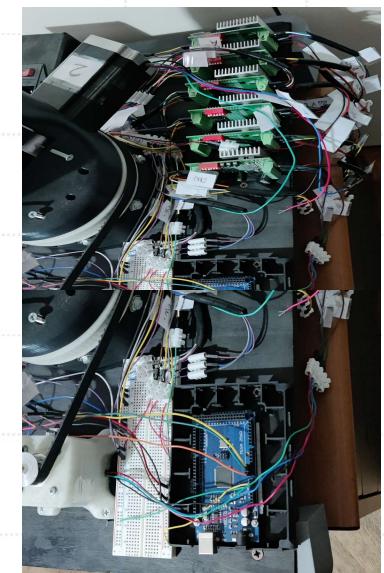
## FINAL RESULT



- 3D Printing
- PLA Filament



- Mechanical assemble
- Electronic assemble
- Electrical connections



# CONTROLLERS



- Single task
- Joysticks -> 6 independent movements
- C++ scripts



# Arduino MEGA 2560



# Raspberry Pi 4b

- Linux Ubuntu 22.04 -> ROS2 Humble
  - MoveIt
  - Python scripts

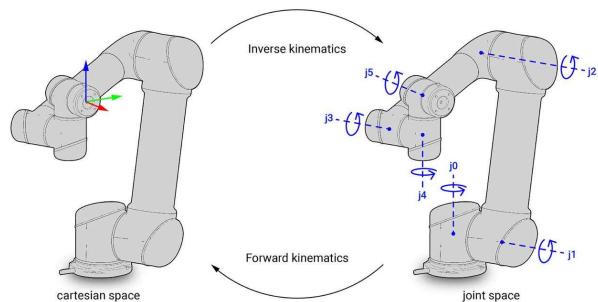


```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

```
import java.util.logging.*;
import java.util.logging.Level;
import java.util.logging.Logger;

class Hello {
    public static void main(String[] args) {
        Logger logger = Logger.getLogger("Hello");
        logger.log(Level.INFO, "Hello world!");
    }
}
```

# KINEMATICS



- Workspace volume
- Accuracy
- Repeatability

## MATLAB computation

```

% MATLAB R2020b - academic use
% File: ROBOT.m
% Author: Giorgia Gennaro
% Description: Forward Kinematics of a 5-link robot using DH convention.

% Link parameters
a = 82;
b = 165;
c = 23.5;
d = 75;
e = 9.5;
f = 4;
g = 10.5;

% DH parameters
theta_i = [pi/2; 0; pi/2; -pi/2; 0];
alpha_i = [0; 0; pi/2; -pi/2; 0];
d_i = [0; 0; 0; d + e; 0];
a_i = [0; c; 0; 0; f];

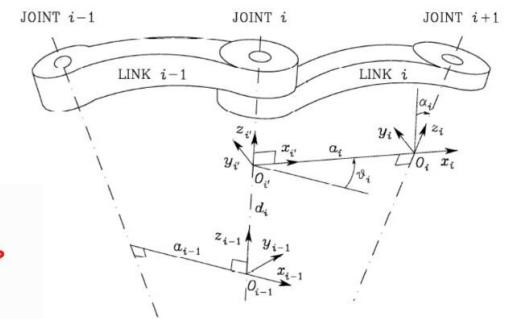
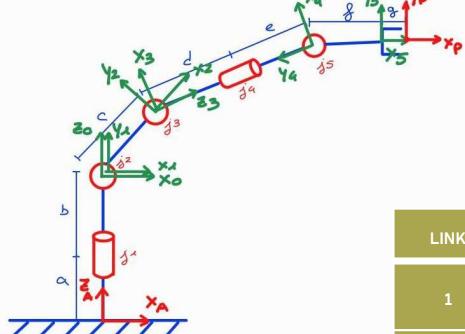
% DH matrix
DH = [theta_i; alpha_i; d_i; a_i];

```

# FORWARD Kinematics

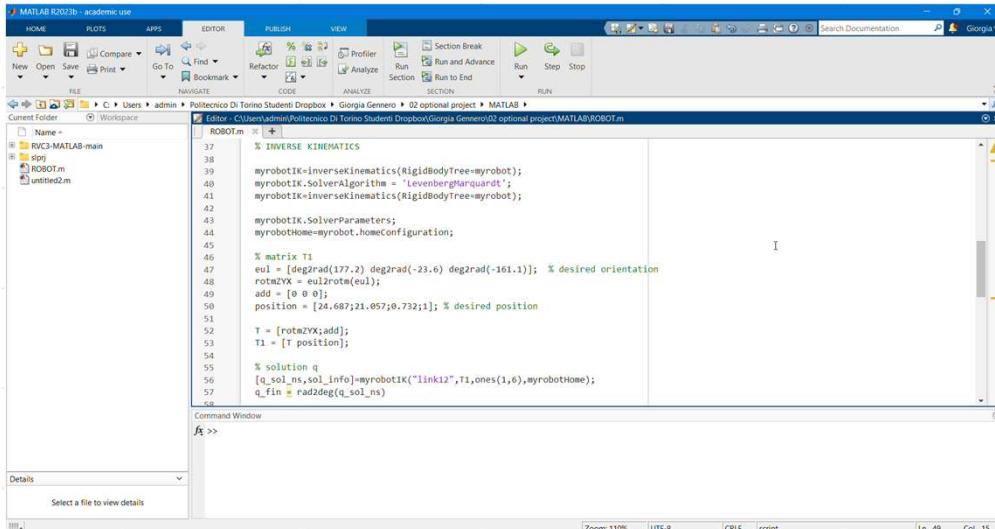
## Denavit–Hartenberg convention

- $a$  = link length
- $\alpha$  = link twist
- $d$  = link offset
- $\theta$  = joint angle



LINK	$a_i$	$\alpha_i$	$d_i$	$\theta_i$
1	0	$\frac{\pi}{2}$	0	$\theta_1$
2	c	0	0	$\theta_2$
3	0	$\frac{\pi}{2}$	0	$\theta_3$
4	0	$-\frac{\pi}{2}$	$d + e$	$\theta_4$
5	f	0	0	$\theta_5$

# INVERSE Kinematics in MATLAB



MATLAB R2023b - academic use

Editor - C:\Users\admin\Politecnico Di Torino Studenti Dropbox\Giorgia Gennaro\02 optional project\MATLAB\ROBOT.m

```
% 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);
q_fin = rad2deg(q_sol_ns)
```

Command Window

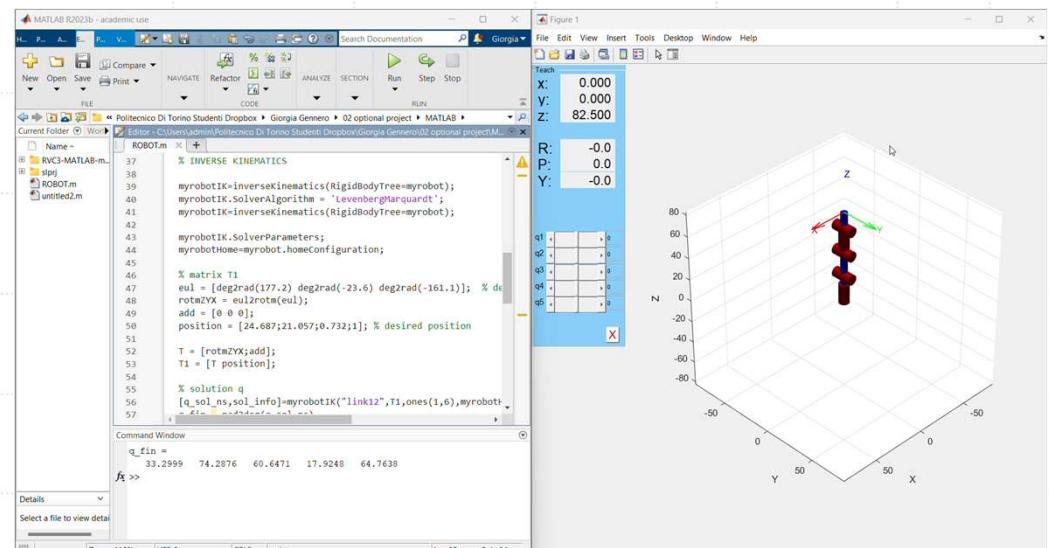
```
A >>
```

Details

Select a file to view details

Zoom: 110% UTF-8 CRLF script Ln 49 Col 15

- Numerical solution instead of analytical one



- Best solver approximation solution

# URDF – Unified Robot Description File

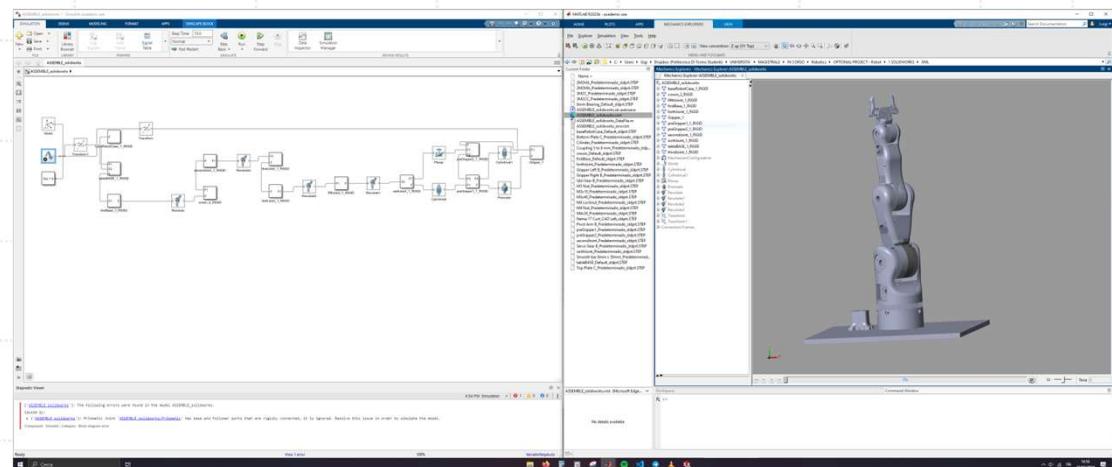
```
final_mod.urdf.x
0 URDF > ROS1 > Gigi > final > urdf > final_mod.urdf
  2 <robot>
  5   <link>
  6     <name>base</name>
  7     <origin>
  8       <xyz>0 0 0</xyz>
  9       <rpy>0 0 0</rpy>
 10     </origin>
 11     <mass>
 12       <value>1.5994</value>
 13     </mass>
 14     <inertia>
 15       <ixx>0.006851</ixx>
 16       <ixy>-3.0652E-05</ixy>
 17       <ixz>4.1042E-07</ixz>
 18       <iyx>-0.0083958</iyx>
 19       <iyz>3.7857E-07</iyz>
 20       <izx>0.0063085</izx>
 21     </inertia>
 22     <visual>
 23       <origin>
 24         <xyz>0 0 0</xyz>
 25         <rpy>0 0 0</rpy>
 26       </origin>
 27       <geometry>
 28         <mesh>
 29           <filename>package://final/meshes/base.STL</filename>
 30         </mesh>
 31       </geometry>
 32     </visual>
 33   </link>
 34   <link>
 35     <name>link_1</name>
 36     <origin>
 37       <xyz>0.1411E-05 1.439E-05 0.08307</xyz>
 38       <rpy>0 0 0</rpy>
 39     </origin>
 40     <mass>
 41       <value>1.5994</value>
 42     </mass>
 43     <inertia>
 44       <ixx>0.006851</ixx>
 45       <ixy>-3.0652E-05</ixy>
 46       <ixz>4.1042E-07</ixz>
 47       <iyx>-0.0083958</iyx>
 48       <iyz>3.7857E-07</iyz>
 49       <izx>0.0063085</izx>
 50     </inertia>
 51     <visual>
 52       <origin>
 53         <xyz>0 0 0</xyz>
 54         <rpy>0 0 0</rpy>
 55       </origin>
 56       <geometry>
 57         <mesh>
 58           <filename>package://final/meshes/link_1.STL</filename>
 59         </mesh>
 60       </geometry>
 61     </visual>
 62   </link>
 63   <joint>
 64     <name>joint_0</name>
 65     <type>revolute</type>
 66     <origin>
 67       <xyz>0 0 0.866</xyz>
 68       <rpy>0 0 0</rpy>
 69     </origin>
 70     <parent>
 71       <link>base</link>
 72     </parent>
 73     <child>
 74       <link>link_1</link>
 75     </child>
 76     <axis>
 77       <xyz>0 0 1</xyz>
 78     </axis>
 79     <limit>
 80       <lower>-1.6</lower>
 81       <upper>1.6</upper>
 82       <effort>100</effort>
 83       <velocity>100</velocity>
 84     </limit>
 85   </joint>
 86 </robot>
```

## 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

```

class Subscription(Node):
    def __init__(self):
        super().__init__('subscription')
        self.create_subscription(FollowPath, 'follow_path', self.follow_path_callback, 10)
        self.get_logger().info('Node has been started')

    def follow_path_callback(self, msg):
        self.get_logger().info(f'Received path: {msg.path}')

```

### Publisher

```

class Publisher(Node):
    def __init__(self):
        super().__init__('publisher')
        self.publisher_ = self.create_publisher(FollowPath, 'follow_path', 10)
        timer = self.create_timer(0.5, self.publish_path)

    def publish_path(self):
        msg = FollowPath()
        msg.path = ...
        self.publisher_.publish(msg)

```

## Python

- Converter angle-steps
- Motors script

```

# Main planner.py
1: import ev3py
2: from ev3py import GPOD as GPOD
3: import time
4:
5: # Pin configuration for each motor
6: MOTORS = [
7:     1: (STEP: 18, OEM: 26, INA: 1),
8:     2: (STEP: 23, OEM: 29, INA: 1),
9:     3: (STEP: 26, OEM: 24, INA: 1),
10:    4: (STEP: 23, OEM: 24, INA: 1),
11:    5: (STEP: 9, OEM: 21, INA: 1)
]

```

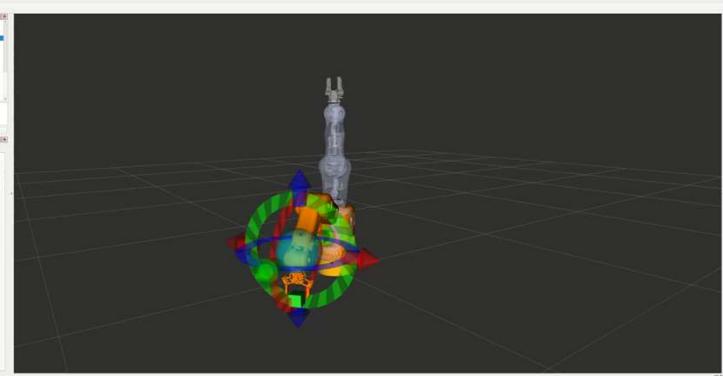
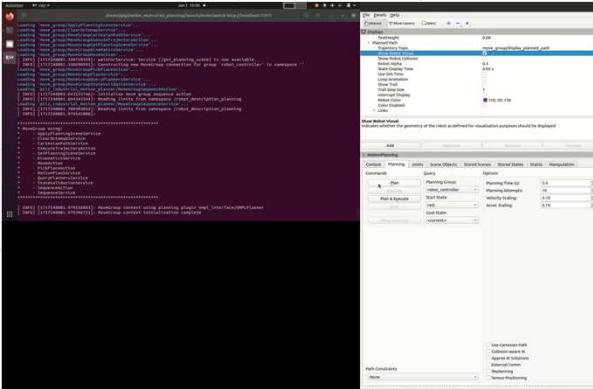
```

1: def angle_steps(angle, transmission, excitation):
2:     """ Converts angles to steps using two coefficients.
3:
4:     Args:
5:         angle: The number to be converted (float).
6:         transmission_ratio: The first coefficient (float).
7:         excitation_ratio: The second coefficient (float).
8:
9:     Returns:
10:         STEPS (float).
11:     """
12:     if not isinstance(angle, float):
13:         raise TypeError("Input number must be a float")
14:     if not isinstance(transmission, float):
15:         raise TypeError("Transmission ratio must be a float")
16:     if not isinstance(excitation, float):
17:         raise TypeError("Excitation ratio must be a float")
18:
19:     angle = angle * transmission * excitation / 1.8
20:
21:     return steps
22:
23: # Example usage
24: angle = 30.15
25: transmission = 5
26: excitation = 32
27:
28: try:
29:     steps = angle_steps(angle, transmission, excitation)
30:     print("STEPS: ({steps})".format(steps))
31: except TypeError as e:
32:     print("Error: ({e})".format(e))

```

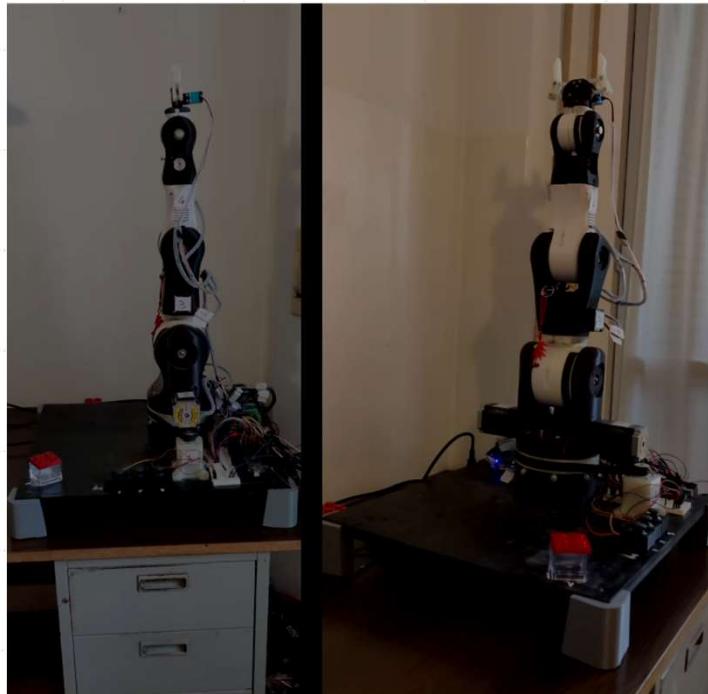
## Movelt

- 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

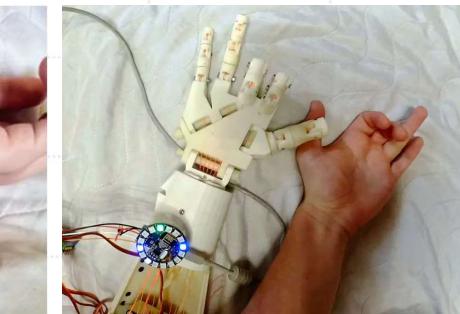


- Stereo camera



- Artificial Intelligence:

  - Voice control

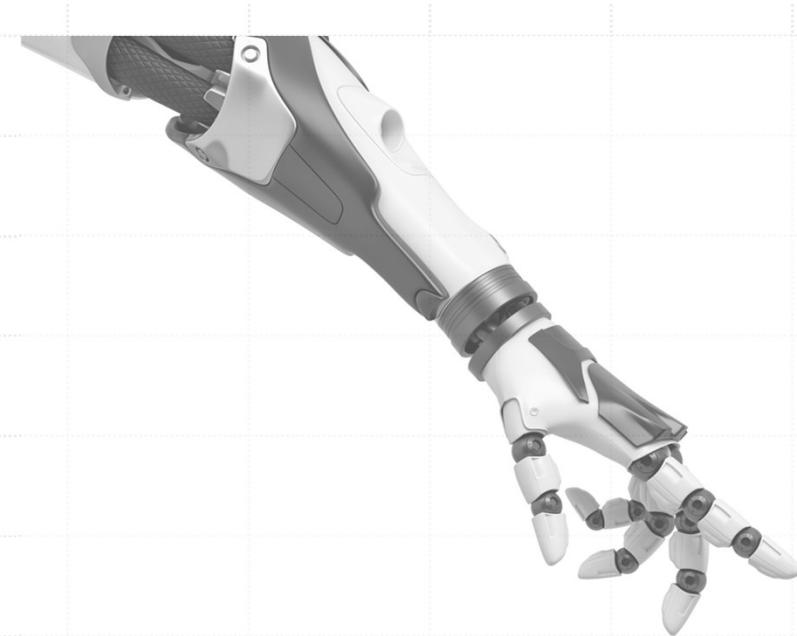




  - Code generator

  - EMG control

**THANK YOU FOR  
YOUR ATTENTION**



**I'm not a Robotic Arm**