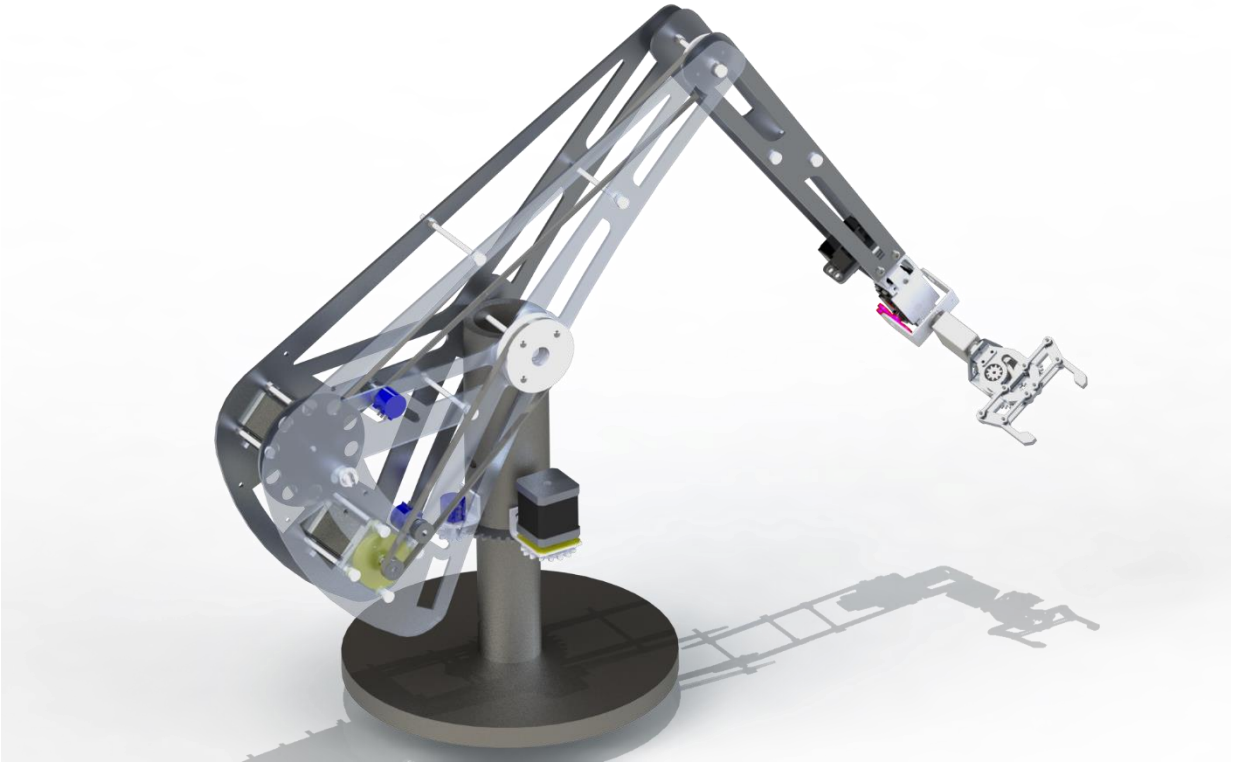
STRONG HOLD REPORT

6-DOF Robotic Arm



SPC535

Ahmed Nashat 201303805 - Ahmad Radwan 201305232 - Amr Mousa 201305701
Ahmad El-Ghanam 201303525 - Ahmed Hashem 201301700 - Gehad Essam 201302548
Mahmoud El-Deep 201300527 - Mahmoud Ayman 201300579 - Essraa Hosny 201306046
Ahmed El-Moslemany 201300465 - Mohamed Ahmed 201304242 - Randa Negm 201305531
Mohamed El-Mallah 201301661 - Mohamed Hussein 201300116

Contents

Abstract

The aim of our project is to design, simulate, control, manufacture and test a 6-DOF (degree of freedom) Robotic Arm manipulator to perform a pick and place mission with highest efficiency in terms of reachability, maneuverability, and controllability as well as the quality of fabrication and design. The given a range is 35 Cm to 65 Cm.

In this project, several design iteration were conducted starting with preliminary design and ending with the final design for manufacturing. Robust control algorithms are implemented using Robotics Operating System (ROS) software. Finally, an elegant design with adequate response rate and control error is introduced with low cost and high efficiency.

Introduction

Mechatronics is synergistic integration of mechanical engineering, electronics engineering, control engineering, and computer science. Serial robotic manipulators are considered of the most famous and most beneficial manifestations of mechatronics. Being used in many applications from industry to home appliances, they helped promote production and welfare. They replace human agents in places of danger and in dull, dangerous and dump tasks. These manipulators are also a great manifestation of control engineering and computer science.

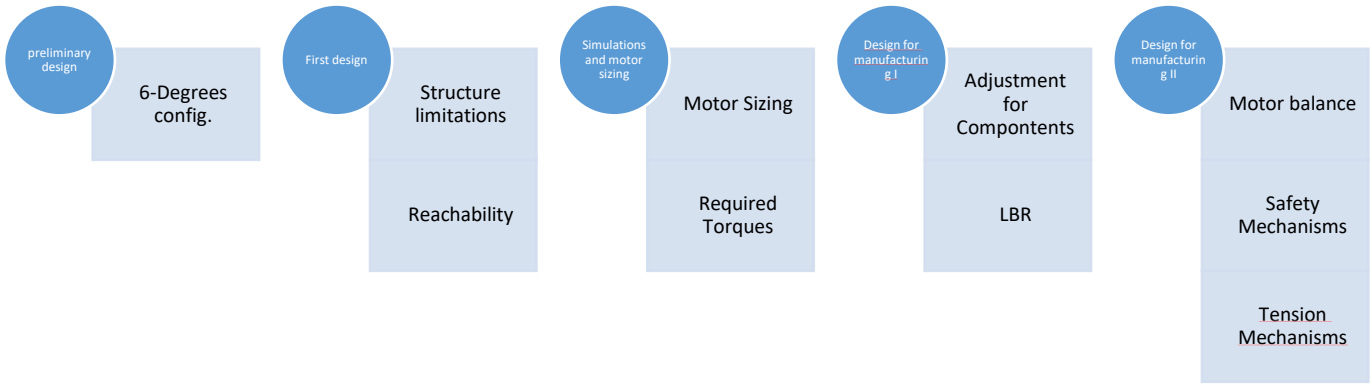
In this project, several design iteration were conducted starting with preliminary design and ending with the final design for manufacturing. Robust control algorithms are implemented using Robotics Operating System (ROS) software. Finally, an elegant design with adequate response rate and control error is introduced with low cost and high efficiency.

Mechanical design and manufacturing

The mechanical design is the first step in the robotic arm design process. The main concept of mechanical design in this project was to try as much as we can to balance the loads on the joints so that the motors have only the dynamical inertial loads to deal with. This will allow for longer lifetime for the motors as well as higher response speed.

At least two design iterations were performed before starting another couple of design for manufacturing iterations. In each of these iterations, the defies of the previous iterations are considered and new concepts and considerations are implemented and tested.

The manufacturing process started after the third iterations, beginning with purchasing and manufacturing the parts that we were sure will not change again in future modifications.



DESIGN I

Even before the start of the preliminary design, allocated members of the team were set to learn mechanical simulation tools such as MSC Adams®. A wide literature review of different kinds and concepts of robotic arms was conducted by all the members of the team.

The preliminary design was not much more than a visualization and proof of concept step for us. The 6-degrees configuration is considered the output of this iteration. It was also used for a preliminary mechanical simulation for the motor sizing. At this stage, note points were taken regarding the structure and support.

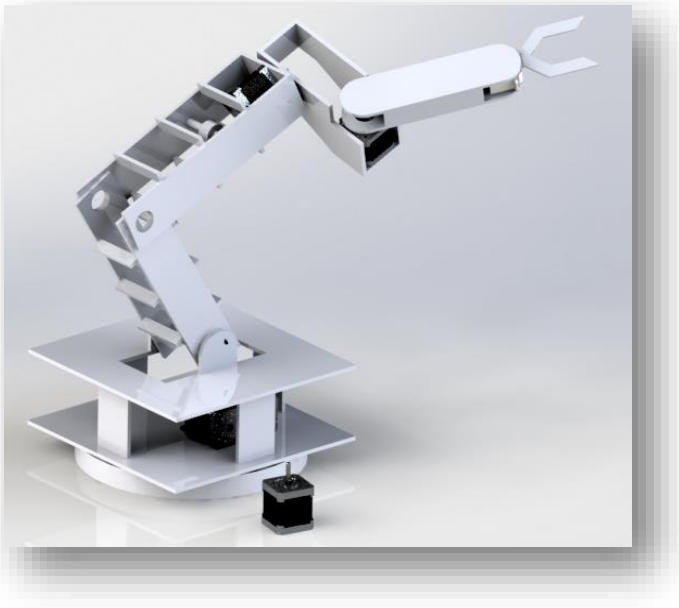


Figure 1: **Plerimilinary design**

DESIGN II

From the preliminary mechanical simulation in the previous iteration, it was found that the motors would require huge torque that is not present in the market. A new idea was needed to solve this problem. We introduced a new design where the motors are placed on the other side of the joint of their respective links as shown in the design image. This reduced the required torque at each motor significantly, yet not enough to meet the market motors.

Figure 2: **Second Iteration Design**

DESIGN III

At this stage, we have started the design for manufacturing. Aluminum sheet metal is chosen as a material for the structure for its stiffness, accuracy and availability in market. Moreover, aluminum metal was too strong that we were able to cut a lot of spaces from it and still achieve a sufficient safety factor.

At the same time, members were creating exact CAD designs for the motors, bullies, belts, electronic components and everything we would buy of-the-shelf including the gripper which we figured would be much better bought than manufactured.

Figure 3: **First Design for Manufacturing Iteration**

DESIGN IV

The final design for manufacturing iteration was a rather radical one. Starting again from scratch yet avoiding all the mistakes and taking all the noted considerations in the previous designs, we were able to introduce an elegant design that requires motor torques small enough to meet the market motors at last. The idea was to put the motor of the third joint before the second joint and connecting it with a belt.

A limit switch was designed and integrated.

To have a detailed explanation of the design, we can break it down to four parts; the base, the first link, the second link and the gripper complex.

The Base

- The base structure is composed of a heavy circular steel flange for supporting the arm and preventing it for tipping over.
- Then there are two concentric steel pipes with a bearing between them allowing for the first degree of freedom, which is a 360 free rotation about a vertical axis.
- The base motor weighs 400 grams and provides 4.5 Kg.CM of torque. The motor is hanged outside of the upper pipe and its gear is attached to the base gear in a sun gear configuration as in figure. The gear ratio is 1:2.5 magnifying the torque of the motor.

Figure 4: **The Base complex; Flange, Pipes, Motor and Potentiometer**

- A similar configuration is attached for the feedback mechanism, using a multi-turn potentiometer for linearity as illustrated in the image.

The First Link

- The structure of the link is composed of triangular double sheets of aluminum laser cut and laser hollowed for weight reduction. The triangular shape is to allow space for the motor of the third joint as shown in the figure.
- A multi-turn potentiometer is used as well and installed along the belt.
- The motor weighs 675 grams and provides 10.4 Kg.CM torque with 1:4 gear ratio.
- A lock mechanism for sudden power cut-off was implemented using a solenoid device depicted in the figure. The function of the solenoid is to push a small rod forward once the power stops. The solenoid acts as a weight balance as well.
- Finally, a tension mechanism is implemented to account for any slight elongation in the belts.

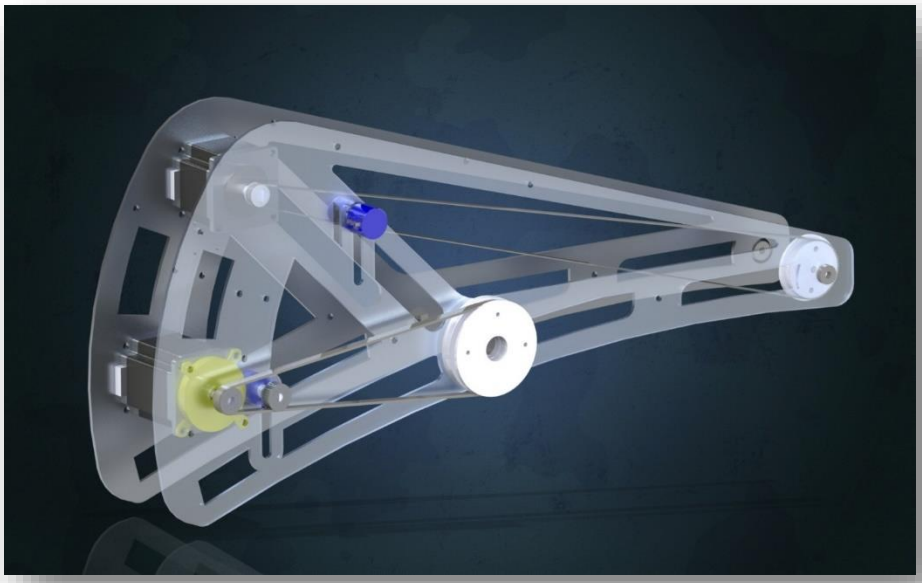


Figure 5: **First Link**

The Second Link

- A light link, since its motor is mounted on the first link for extra weight balance, and connected to the joint via a belt with ratio 1:5 as shown.

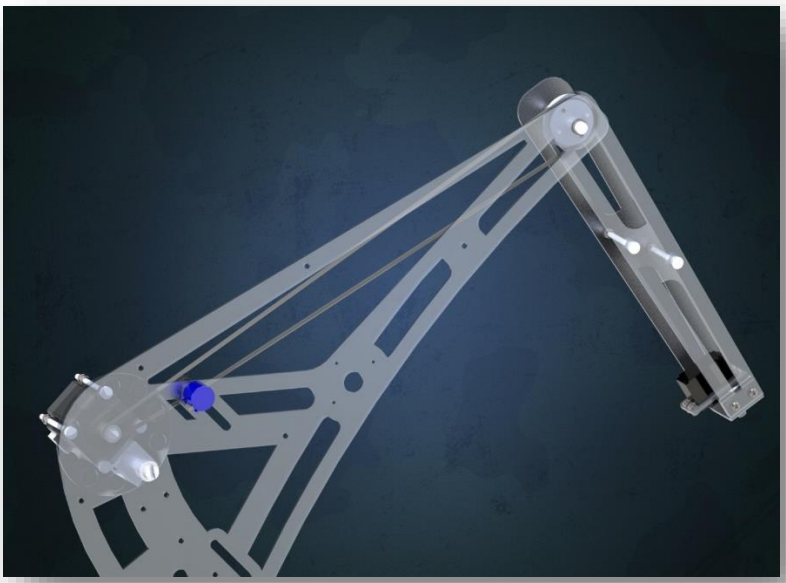


Figure 6: **Second Link Motor, belt, tension mechanism and Feedback potentiometer**

- The motor weighs 1025 gram and provides 12 Kg.cm torque.

- Similar to the previous link; tension-feedback mechanism is installed on the belt using multi-turn potentiometer.
- A solenoid device for safety is installed as well

Figure 7: **Second Link and Joint**

The Gripper complex

- The gripper complex was one of the most challenging parts of the mechanical design. Although we concluded we would buy an off-the-shelf gripper, its installation, control and alignment was quite a challenge.

Figure 8: Gripper Complex auxiliary DOF

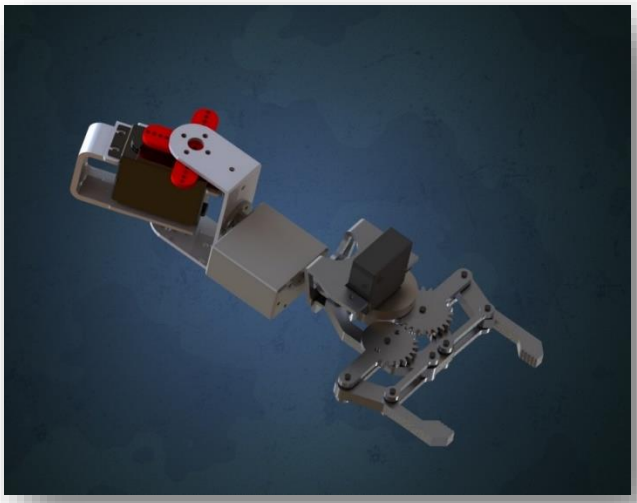


Figure 9: **Gripper Spherical joint assembly**

Full Design

- Different parts and designs were assembled and the full model was ready. Final analyses were performed and the manufacturing process started.

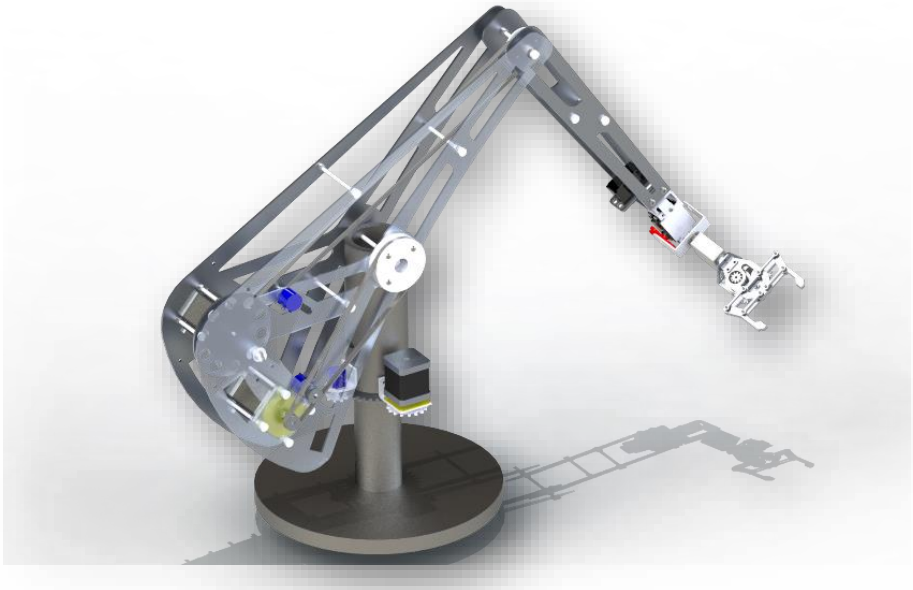


Figure 10: **Full Ready-for-Manufacturing Design**

- Forward and inverse kinematics calculations were conducted to the robot. The robot reachability limits and configuration space was drawn. These analyses were done using MATLab Robotics Toolbox.

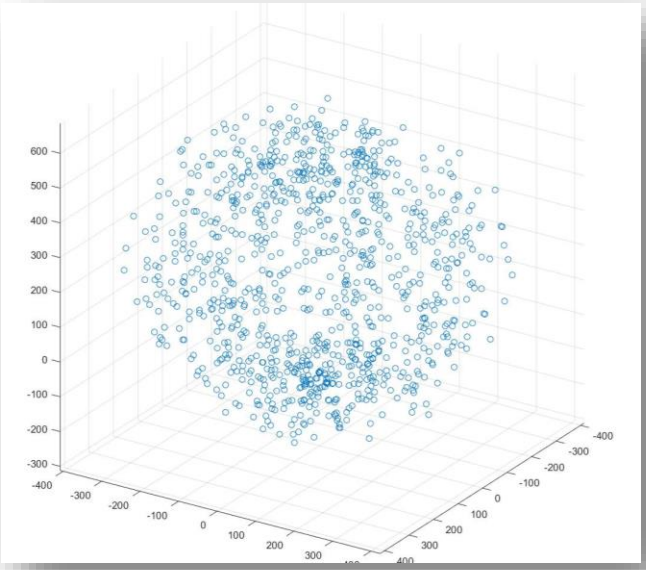


Figure 11: **Robot Configuration Space**

Manufacturing and assembly

- As mentioned before, the gripper was decided to be off-the-shelf, so it was one of the first things to buy. A detailed CAD for it was then created.

Figure 12: **Gripper end effector**

- The limiting motors were bought to fix some of the variables of the design process. This was a market constraint on us justified by the mechanical simulation and motor sizing analysis.

Figure 13: **Purchased Motors**

- The solenoid was also bought as it constituted a market limit as well.

Figure 14: **Safety break; the solenoid**

- The base was the first part to be manufactured. It was done using milling.

Figure 15: **Robot Base**

- The links sheet metal parts were then laser cut.

Figure 16: **First Link Sheet metal laser cut parts**

- The rest of the parts were laser cut sheet aluminum as well. The complex parts were folded.

Figure 17: **End effector support parts, laser cutting and folding were applied**

- The robot was assembled then the motors were installed. A special passage for the electrical wires was taken into consideration in the design and in assembly.

Figure 18: **Robot assembly, motor and belts installation, and wiring**

- Finally, the power supply and control box were installed. A cooling fan was attached for each system to prevent overheating.

Figure 19: **Power supply installation and Control box augmentation**

Electrical and control

The main task of the electrical and control team is to solve the inverse kinematics problem for the robotic arm. ROS (Robotic Operating system) have been used for his mission as it is very common platform for robotics industry nowadays. ROS is an open source license and has a very good community all over the world helping to develop it and solve the problems. ROS provides libraries so users do not start the robot programming from scratch and that is the main advantage of it. ROS, also, provides hardware abstraction, visualization and good package management.

Motion planning of the robotic arm

The motion planning package that we have used is Moveit Package. Moveit is a highly adaptive ROS package that uses plugins for motion planning algorithms. We can use any motion planner by simply changing the plugin. This method is highly extensible so we can try our own custom motion planners if needed. Moveit default uses RVIZ for visualizing the arm trajectory and it provides many inverse kinematics solvers. We have used IK solver which shows some errors that will be discussed later. We have tried many design iterations before reaching the final one. Figure 1 shows the first proof of concept iteration and the final design visualized by RVIZ.

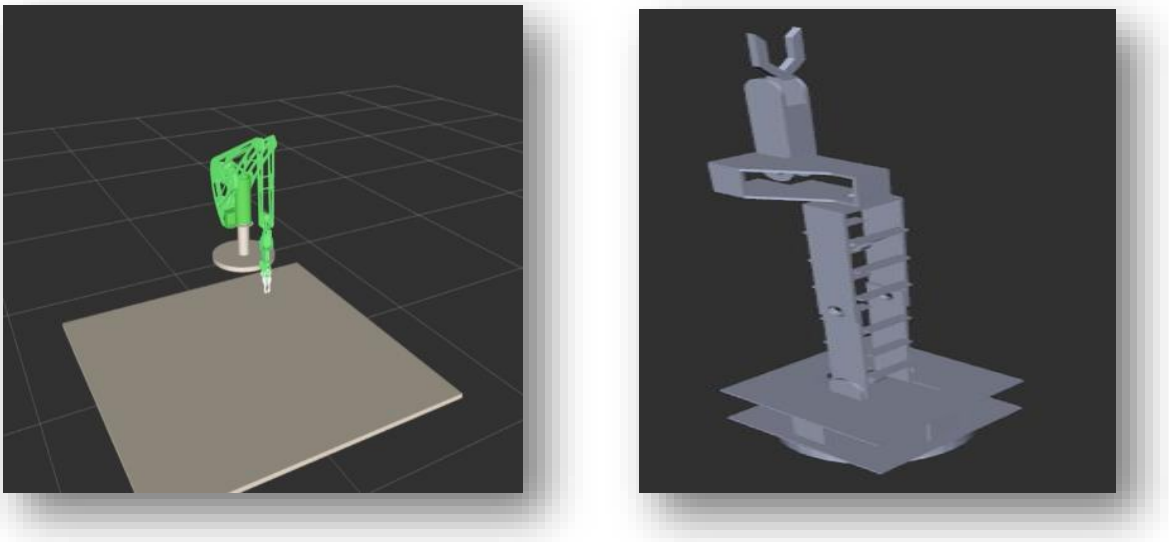


Figure 20: **Control simulation designs**

In the next section, we will go through the steps beginning from the CAD model to fully simulating the trajectory and later apply it on the hardware.

Simulation phase

URDF Model

SolidWorks parts are all we get from the mechanical team. To visualize the robot on ROS, The CAD model must be converted to URDF (Unified Robot Description Format) file. This file describes the robot links, joints, actuators and if there any extra plugins like gazebo control plugin. For the link description; mass, inertia, geometry, origin, collision and color should be specified. For the joint description; the parent and child links must be specifying, the type of the joint; prismatic or revolute and the limits of the motion, velocity and effort. By the end of this step, we can visualize the arm using RVIZ and test the joint poisoning and limits.

Motion planning task

ROS has made it very easy to do the motion planning task by Moveit package with its built-in solvers and plugins. Launching moveit_setup_assistant helps a lot introduce the arm to Moveit package and do some great functionalities like:

- It makes a Self-Collision matrix. This matrix is internally saved for the arm and help greatly reducing the time calculating the collision every time.
- We can add passive joints to be not taken in consideration when planning the trajectory.
- Adding planning groups; in our case, two groups have been added. The arm group to move as a chain from the base till the grasping frame and the gripper group to simplify the pick and pace task as possible.

The product of the setup assistant is a configuration folder that has the information about the arm kinematics, joint limits, optimization technique and inverse kinematics solver. We have use a short path optimization method and goal tolerance of 0.01.

Testing Motion Planning Using RVIZ

Before conducting real physical simulation, we tested that the planner is running using rviz visualization tool as shown in figure.

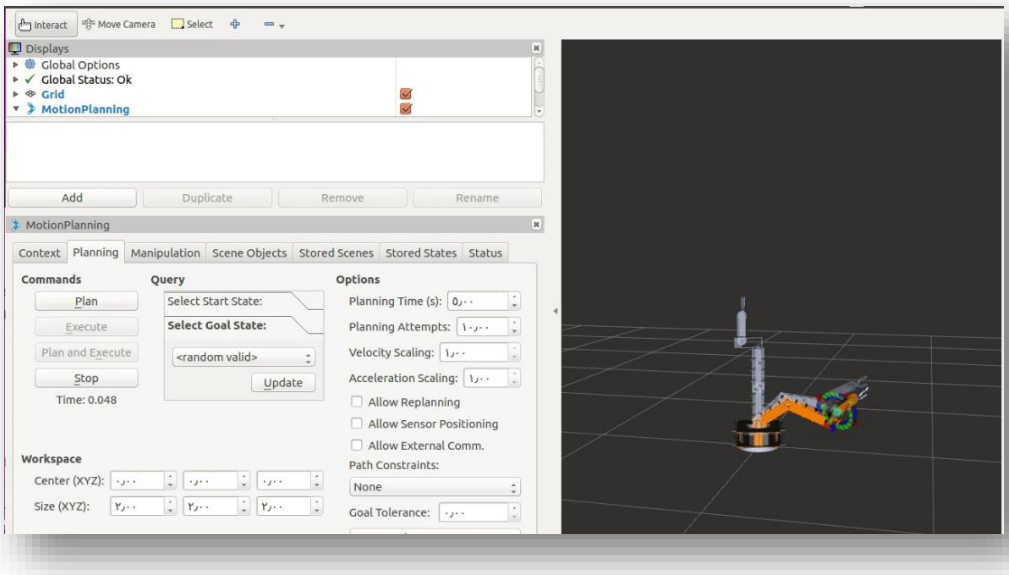


Figure 21: **RVIZ visualization of robot**

Simulation using GAZEBO physics simulator

Gazebo simulation is useful to test the real world actions and responses. There are standard steps to make insert actuators and controllers on the joint instead of the fake controllers used in RVIZ. The controller configuration file task is to define the control groups, the PID values and the controller type. We have used the default PID expecting that the values will be changed when implementing on the hardware but the default PID values give a very reasonable accuracy so we did not change it. A very important task is to specify a certain controller as ROS has many types of controllers. We have used joint state controller so every joint gives a feedback state. We also have used trajectory controllers and position controllers.

Pick and place

Moveit provides an interface for applying pick and place function. The interface consists of

- Pick class
- Place class

The pick class takes an input a grasping frame for the object and produce a suitable trajectory message for the actuators to grab the object without colliding with it. To provide a grasping frame we use the user defined package (simple_grasps) written by Davetcoleman. The figure below shows Pick applied on a simple design of arm.

The package isn't modular so it must be modified to be used with custom manufactured robotic arms. The place function isn't mature enough to make it modular and apply it on our robot.

The place function was applied by using direct move group interface by:

- Determining the final position of the object
- Determining the constrains of the bath
- Determining a Cartesian points for the final position
- Controlling the gripper manually

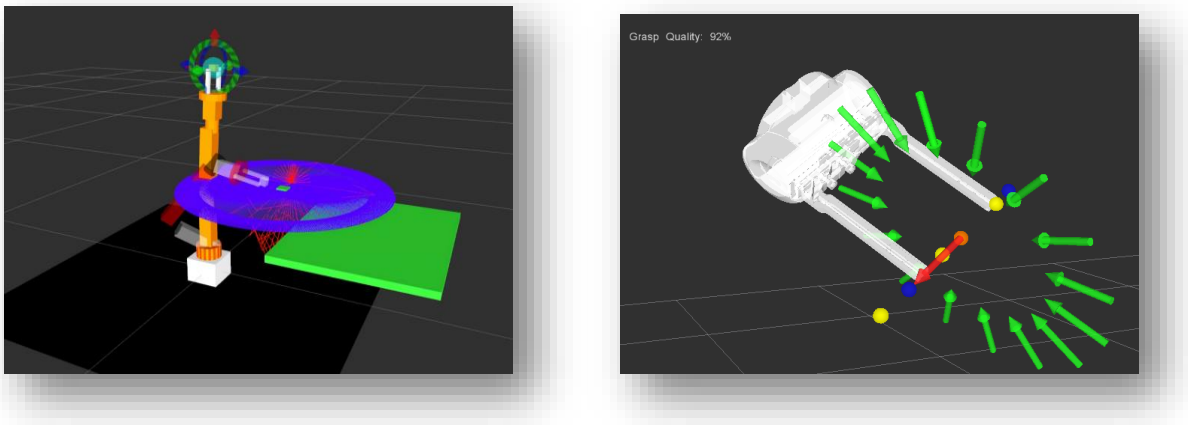


Figure 22: **Pick and Place Package**

Hardware Implementation phase

In this section, we will go through the connection between mechanical and simulation work. The phase that makes the arm alive.

Motors & Microcontroller

The requirements was to command angles from ROS and the angles should be executed with a microcontroller. So, we've used an Arduino microcontroller to interface with servo and stepper motors. The servo motors weren't a big issue as they have their own internal feedback mechanism which allows for direct angle commands to be sent directly from the microcontroller and executed quickly without much lag. The real problem was to map stepper movements and how many pulses it take to move for one revolution given that we don't use micro-stepping as we need the motor to have its full torque. So, we estimated how many pulses given the actual stepper motor pulses per rev and gear ratio between the motor gear and the driving gear then we had to tune this estimation to get accurate angles.

Interfacing Arduino with Moveit

Now we have done the motors interface, it is time to get it to work with ROS moveit, listen to the angles published and execute them. So, we have created a subscriber on the Arduino using ros_lib library. This library allows us to use the Arduino as a node that publishes and subscribes to topics easily using another package in ROS called rosserial_arduino. What the Arduino code does exactly is to subscribe on a topic named "joint_states" with a high frequency (around 50 Hz) to get the angles published by moveit package on this topic and move each joint to its desired angle. Also we have used a library for stepper motors called Accelstepper which move each joint to a desired location with max speed and max acceleration specified in the code. This guarantees that there will be deceleration and acceleration in each movement which results in more rigid motions avoiding much vibrations and avoiding missing steps by the steppers.

To make the process reliable and simple so we made the robot starts from its zero position every time we reset the Arduino. Figure shows a schematic for the hardware:

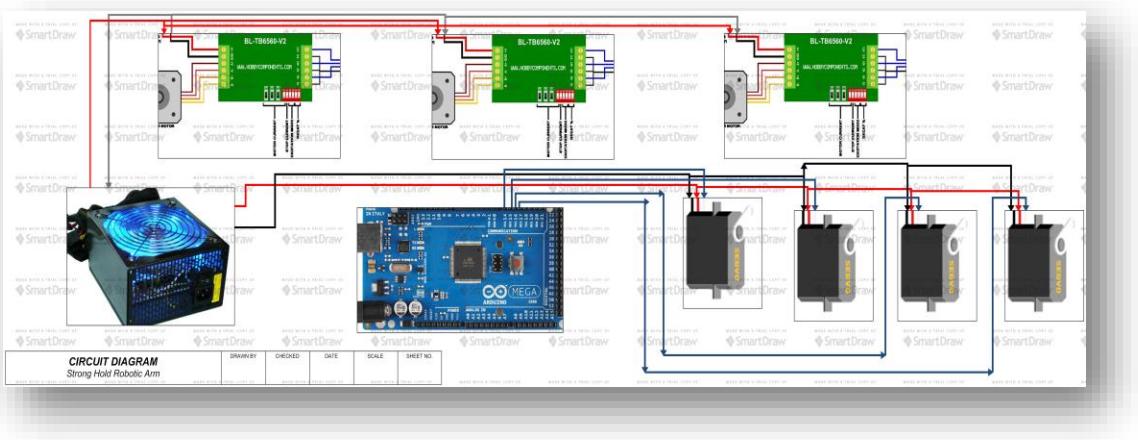


Figure 23: **Hardware schematic**

Robotic Vision

Object recognition is a process for identifying a specific **object** in a digital image or video. **Object recognition** algorithms rely on matching, learning, or pattern **recognition** algorithms using appearance-based or feature-based techniques.

In this approach, DevIL image library along with opencv & open GL & GL shading language is used to implement this, so the plane is to finish the part of the robotic vision on several steps :
- Object Recognition
- Object detection
- Object tracking
- Multiple object tracking

Object recognition

The vision and robotics communities have developed a large number of increasingly successful methods for tracking, recognizing and online learning of objects, all of which have their particular strengths and weaknesses. A researcher aiming to provide a robot with the ability to handle objects will typically have to pick amongst these and engineer a system that works for her particular setting.

How is this done?

The system works with a CAD model of the object provided. The current implementation of the detector module uses SIFT feature descriptors to provide an approximate estimation of the object's pose for the tracker module which will track the object using edge-based methods. The algorithm used here depends on basically writing fragment & vertex shader files through opengl shading language.

Demo example

In this demo, it's presented how a detector can recognize a previously known object such as Pringles box & detect its position in 3D

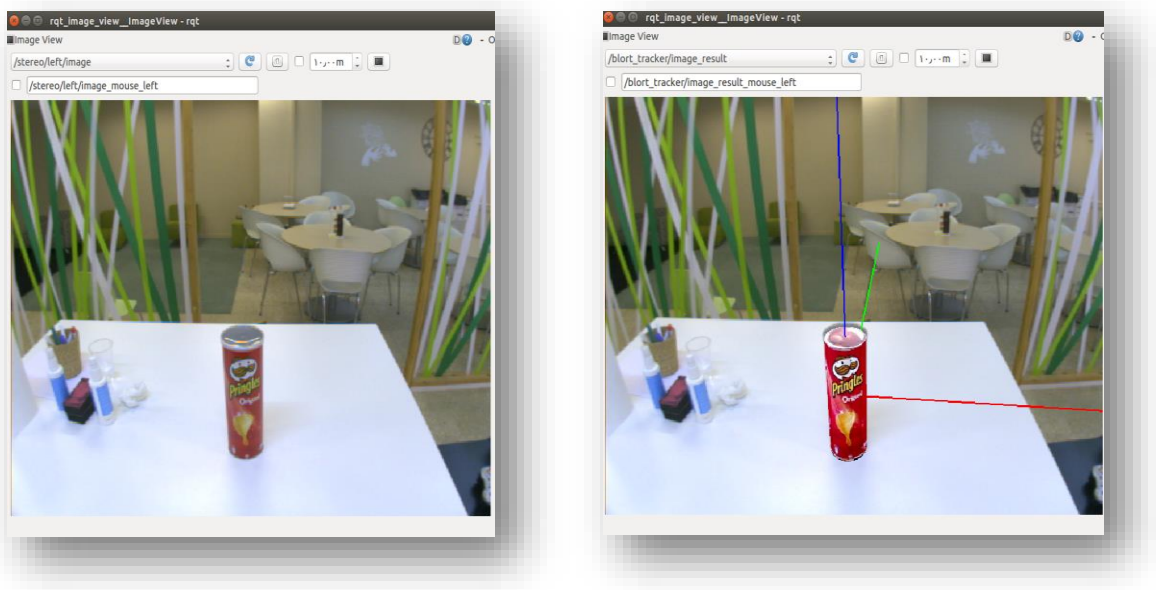


Figure 24: **The original video frame Vs the tracker output**

```
pose:
  position:
    x: -0.0281778407724
    y: 0.155703602358
    z: 0.70790146651
  orientation:
    x: 0.839918992522
    y: 0.0403102525056
    z: -0.0345629904414
    w: 0.540107923693
```

Figure 25: **Tracker results**

In the previous example, a trained model is used but in case we need to train a new model we have to create a sift file, the training needs to be done manually.
To begin the training stage a **CAD** model of the object is needed. Blender software is recommended.

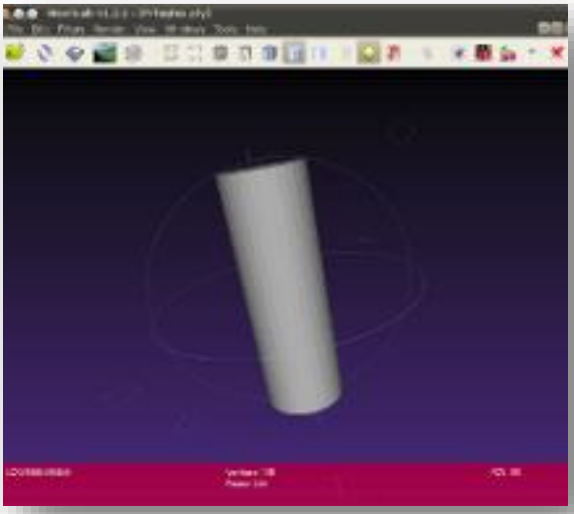


Figure 26: **Simple CAD model for training**

By orienting the object to match its cad file in all faces, the training is done & the object is ready for detection. The object must be added to the detector list in tracking.ini file & it's all done.

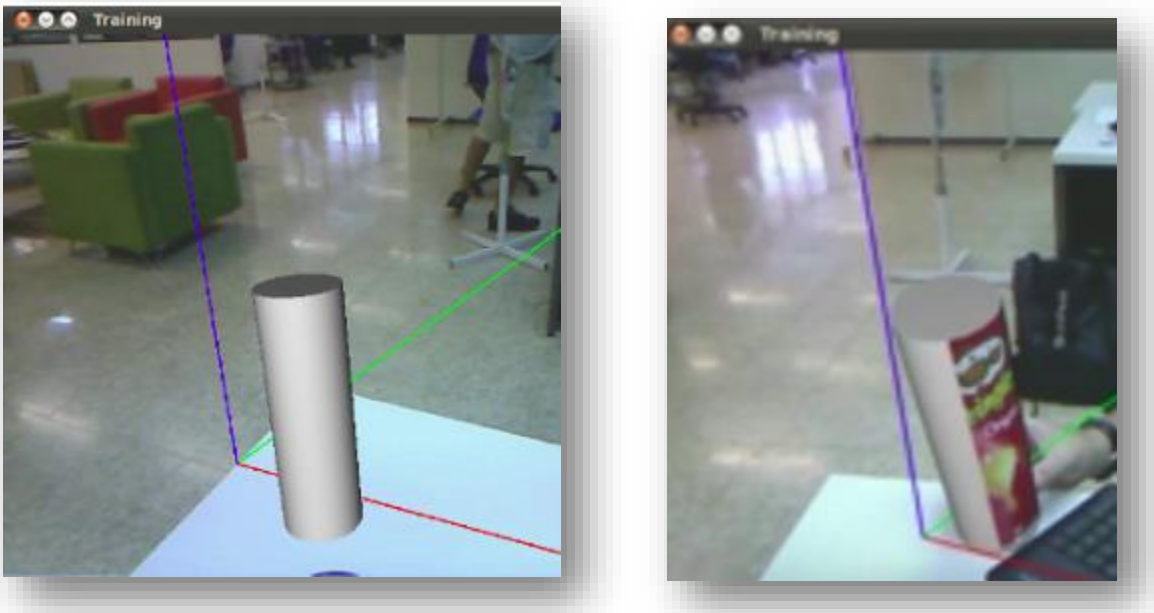


Figure 27: **Tracker training**

Conclusion

A 6-DOF serial manipulator robotic arm with range of 90 cm reachability and 65 cm with any orientation. Mechanical design iterations were done implementing non-conventional ideas; namely weight balance for static load elimination.

Moveit package from ROS was used for path planning and control. Robot vision was tested.

Market and technology limitations were faced, yet the robot was successfully manufactured and tested. For further work, limitations of accuracy, efficiency and speed is to be optimized.

A video of the testing was produced.