Overview

The Robotic Arm was conceived to be an easy-to-build and inexpensive platform for research and education in STEM fields. Currently available 6 degrees of freedom manipulators are either exorbitantly priced or are too rudimentary to be reliable teaching and learning assets. The fundamental goal of this project is to create an affordable, open-source manipulator without compromising on capability. The six degrees of freedom support a vast variety of functionalities and ensure rugged stability while maintaining a considerable level of dexterity.

The robot is actuated by six NEMA stepper motors with varying mechanical reductions in the form of 3D printed gearboxes. Accurate movements and end effector forces are achieved with feedback from AS5048B hall effect encoders and ACS723 current sensors. A STM32F103C8 microcontroller serves as the primary low-level control device.

The Arm has been extensively simulated in gazebo using ROS framework as well as in MATLAB. A number of mathematical control laws and trajectory planners were implemented in these simulations.

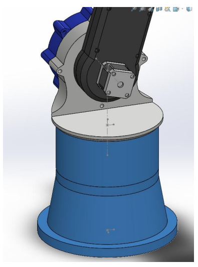
The robot is general purpose by design and hence has application in various tasks such as basic pick and place operations and path tracing. It is intended for the manipulator to be used in conjunction with a computer for computationally intensive tasks.

Technical

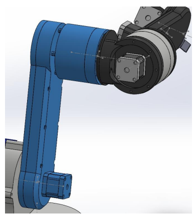
Mechanical

Two versions of the robot's mechanical structure exist with the electronic hardware being identical for both. The current design version-version 2 saw major mechanical upgrades and is detailed as follows.

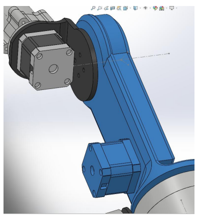
The mechanical structure of the manipulator is entirely 3D printable. Mechanical reduction for the joint motors is achieved with 3D printed gearboxes.



Both Joints 1 and 2 have a cycloidal gearbox with a 28:1 reduction ratio. Cycloidal gearboxes are the ideal choices due to the high reduction ratios and negligible backlash. The actuators of both joints are coupled with additional gear trains for further reduction, giving a total combined reduction ratio of 50:1.



Joint 3 has a planetary gearbox, composed of helical gears. This leverages the relatively smoother performance characteristics of helical gears as compared to spur gears. The input for the gearbox is supplied by a stepper motor located at the base of the parent link through a timing belt.



Planetary gearboxes are also employed by joints 4 and 5, each having a reduction ratio of 4.5:1. Joint 4 is directly coupled to the stepper motor whereas joint 5 is supplied input power through a timing belt.

Stepper motors (Nema 23 and Nema 17) are the actuators of choice. These actuators, when combined with a suitable driver circuit, provide accurate position control and large stall torques to maintain robot configurations under external loads.

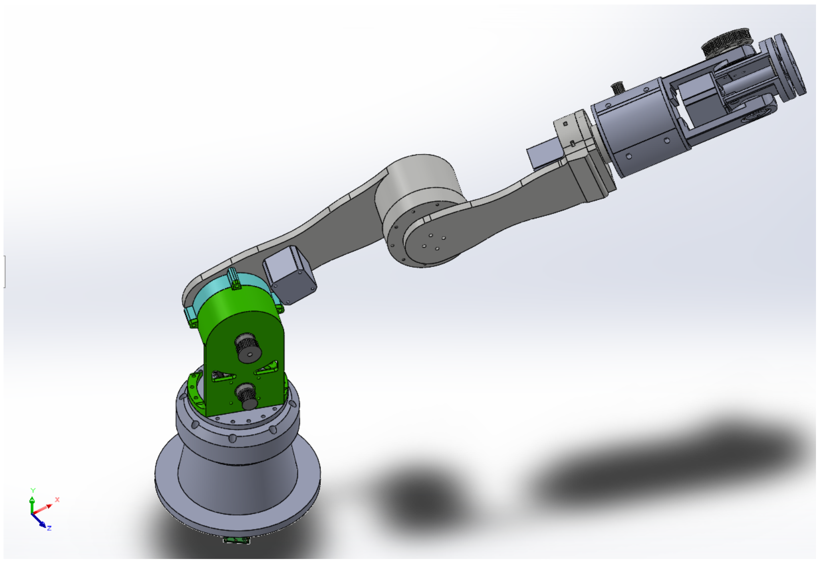


Fig. 1. Version 1 prototype

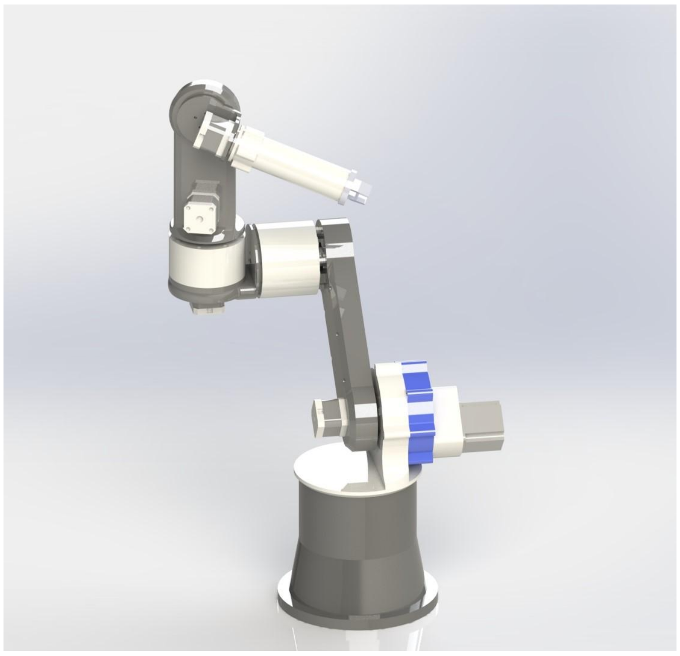


Fig. 1. Render of the latest version

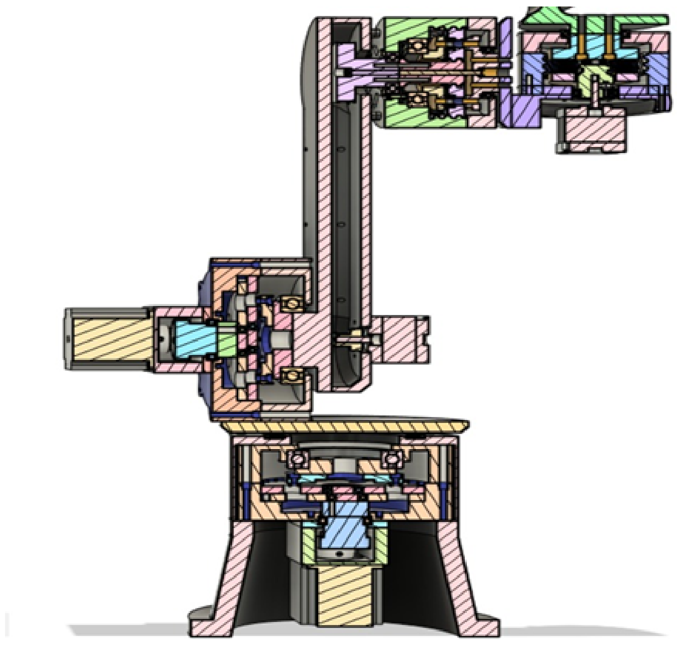


Fig. 1. Sectional view of robot base and first link

Electronics

The onboard low-level control device is a STM32F407 discovery board microcontroller. This particular microcontroller was selected for its 168MHz clock frequency,

Each stepper motor has a ACS723 current sensor connected in series to provide an indication of individual joint torques. This is essential for torque control of the joints as well as for safety purposes.

AS5048B hall effect encoders provide joint position feedback. Stepper motors, when combined with additional feedback sensors are capable of high accuracy movements under varying load torques. The encoders are connected in a daisy chain configuration to reduce the amount of wiring required.

The stepper motors are driven by six TB6600 stepper motor drivers.
All components will be connected with a custom designed printed circuit board.

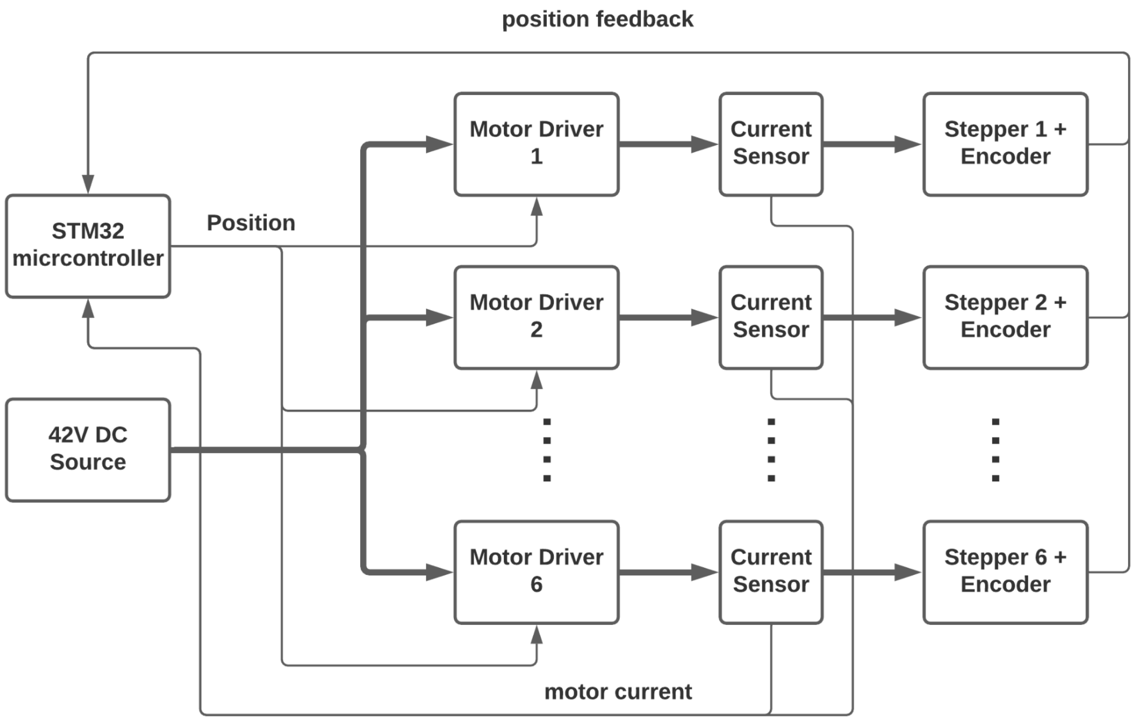


Fig. 1. Block diagram of the manipulator electronics.

Control

This section serves to describe the mathematical equations responsible for trajectory planning and actuator control. The entire functional algorithm will consist of two parts, the trajectory planner and the control law. The trajectory planner can be further subdivided into two parts, a path in the end-effector task space and a time scaling. The planner will output a list of end-effector velocities at regular timesteps when supplied with a start and goal state. The velocity of the end-effector can be expressed as a spatial twist that will take the end-effector reference frame from initial to goal configuration in unit time. Adding a quintic time scaling factor will result in smoother joint acceleration.

$$X(s) = X_{initial} e^{log(X_{initial}^{-1} X_{end})s}, \; s\epsilon[0,1]$$

Eq. 1. Vectorized taskspace trajectory equation

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, \; t\epsilon[0,T]$$

Eq. 1. Quintic time scaling factor. Where polynomial coefficients are evaluated from desired boundary conditions.

The second part of the algorithm consists of the control law. The robot arm is expected to move through free space and hence a simple motion controller will be sufficient for reliable functioning. The controller commands joint velocities to match the desired velocity trajectory generated by the taskspace trajectory equation.

$$U = J_b^{-1}(\theta)([Ad_{X_{bd}}]\upsilon_d(t) + K_p X_e(t) + K_i \int_0^t X_e(t)dt)$$

Eq. 1. Vectorized Equation for PID control of joint velocities.

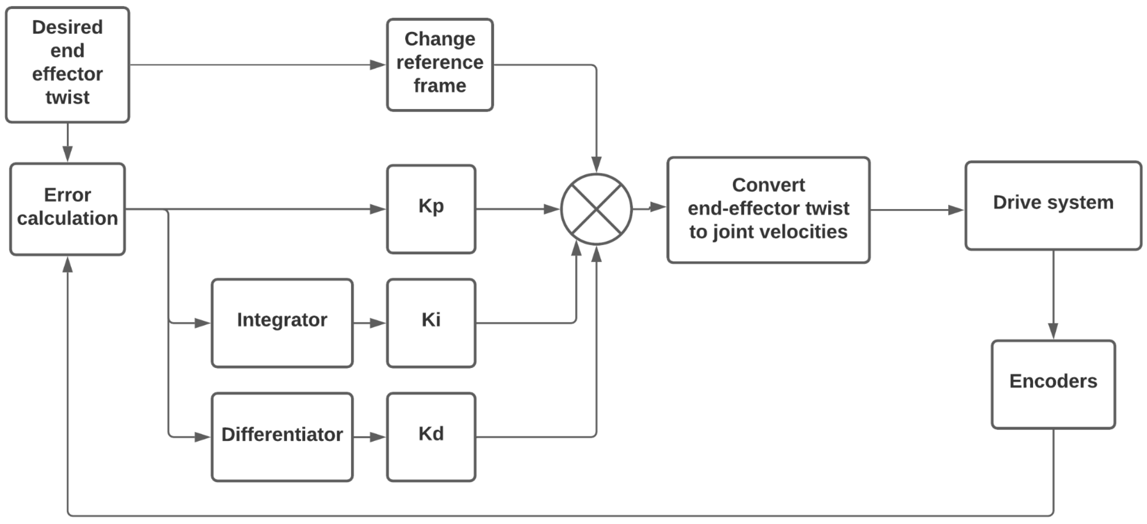


Fig. 1. Motion Controller Block Diagram.

Simulations

Simulations for both versions were performed and evaluated. Manipulator actuations were simulated in MATLAB. The robot’s URDF file is used to construct a rigid body tree for this purpose. Goal states were set by manually rotating the joints using the visualizer interface. The motion of the robot was guided by straight line joint space trajectories scaled by a quintic time scaling factor.

$$\theta(s) = \theta_{start} + (\theta_{end} - \theta_{start})s,\ s\epsilon[0, 1]$$

Eq. 1. Vectorized expression for straight line joint space trajectory.

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5,\ t\epsilon[0, T]$$

Eq. 1. Quintic time scaling

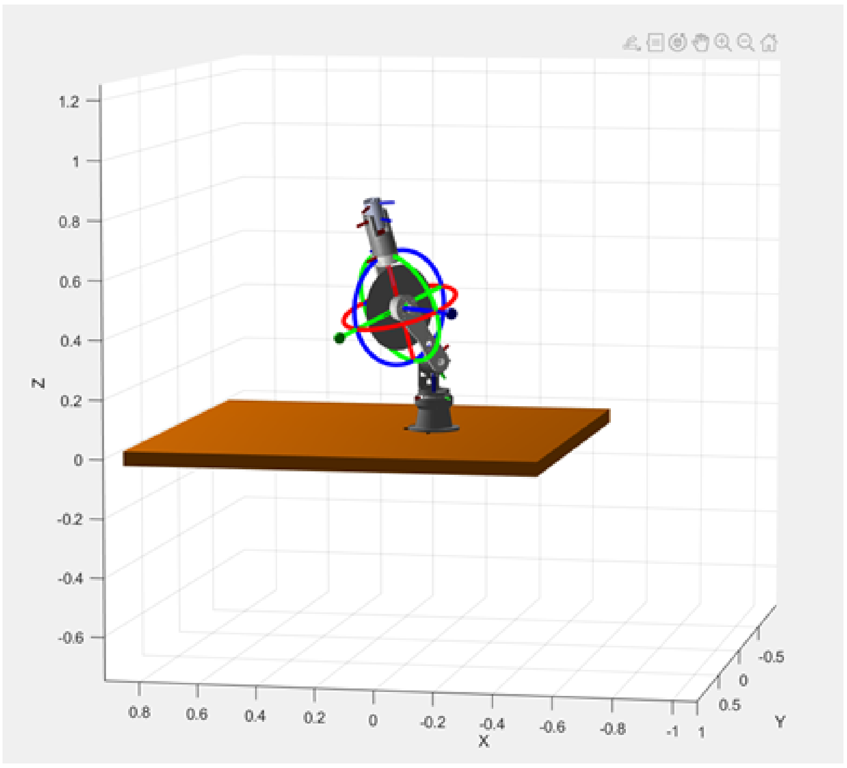


Fig. 1. The goal joint configurations are set using the visualization interface.

The simulation also involved pick and place operations. The goal configurations were set by manually positioning the end effector using the visualizer interface. An inverse kinematic solver was used to calculate joint values for each goal state and the previously mentioned joint space trajectory planner generated the timestamped joint values to be followed.

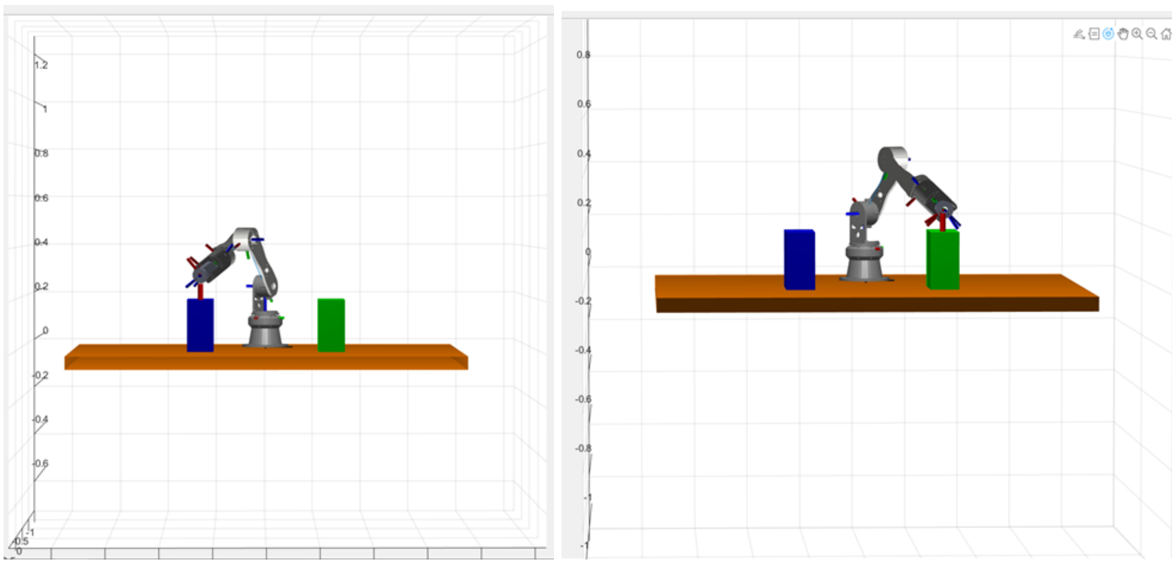


Fig. 1. Simulation of pick and place operations in MATLAB

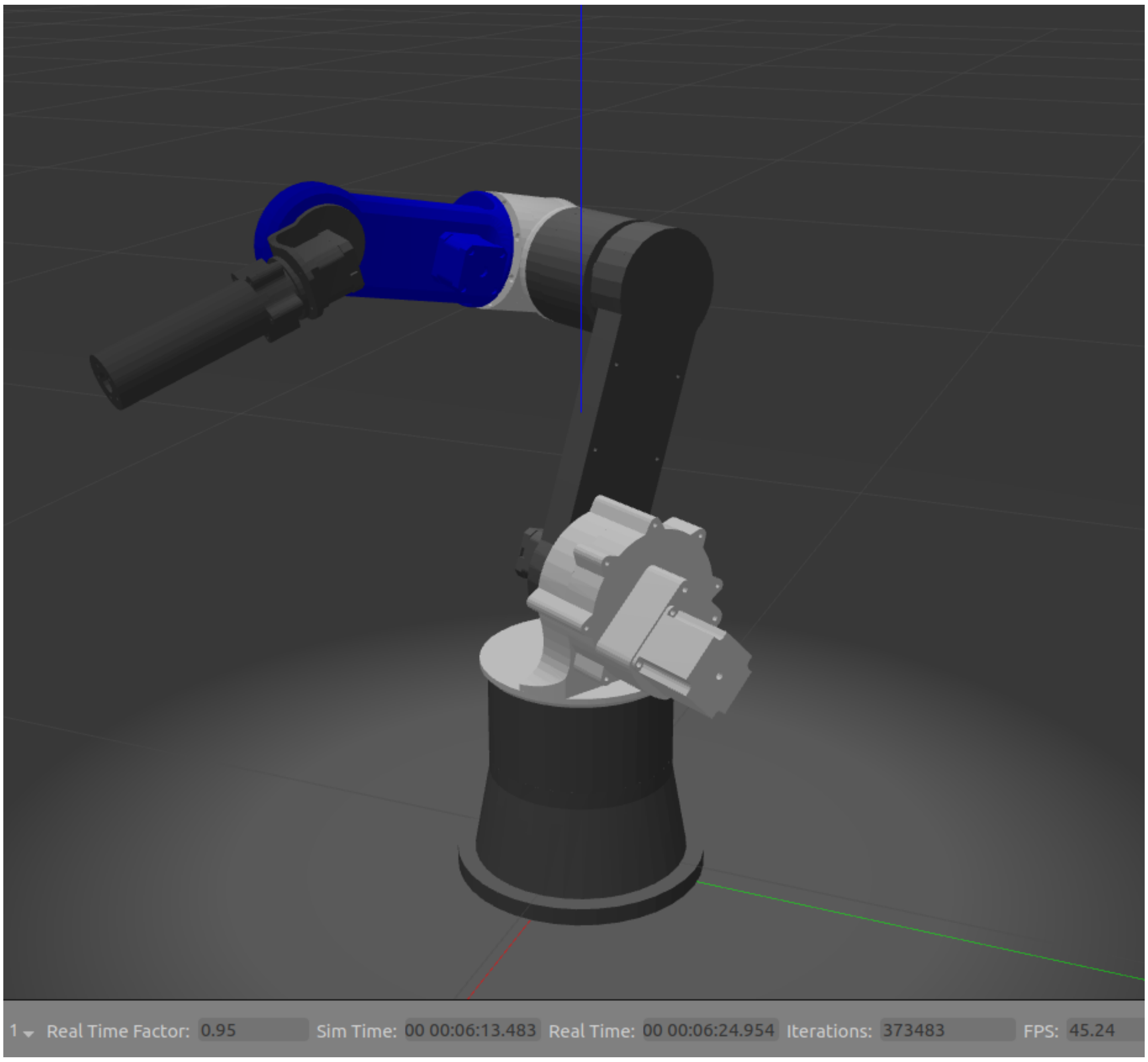

1 Real Time Factor: 0.95 Sim Time: 00 00:06:13.483 Real Time: 00 00:06:24.954 Iterations: 373483 FPS: 45.24

Fig. 1. Joint position control simulation in gazebo

Future Prospects

As it stands, the manipulator is reliant on an external computer system for performing computationally intensive calculations in real-time, such as actively generating joint space trajectories from task space configurations. Additional computing power will be required for evaluating the robots dynamic model for torque control applications. These functions can only be reliably executed by a powerful single board computer if the robot is to become a stand-alone system. Hence a Nvidia Jetson Nano Development Board will be required for the same.

To enable environment perception and other such autonomous functions an Intel RealSense stereo camera must be interfaced with the main computing system. This will greatly expand the capabilities of the robot and make it a testbench for future research in the fields of computer vision and machine learning.