CAL POLY

CALIFORNIA POLYTECHNIC STATE UNIVERSITY
CENG - DEPARTMENT OF ELECTRICAL ENGINEERING

EE 471- Vision Based Robotic manipulation

Lab #1: Joint Space Control Interface, Data Logging, and Visualization

Presented to:
Siavash Farzan

Presented By:

Rodrigo Menchaca
Giovanni Dal Lago

OCT 3, 2025

INTRODUCTION

During Lab 1, we continued from the learnings of Lab 0 and started to implement joint-control space and data logging for our Robot arm. In order to implement this we used the code from lab 0 as our basis and added additional parameters with the using of pytool extensions. We created a code that would start logging the angle position of each joint as it moved to a given position as shown in Figure 0, joint angle data was saved in a python array for later use. Along with saving different joint angle positions, we also saved timestamps of each time the angles were measured. We first set the robot for a trajectory time of 10s from home position to the desired position and later made tests at a 2s trajectory time.

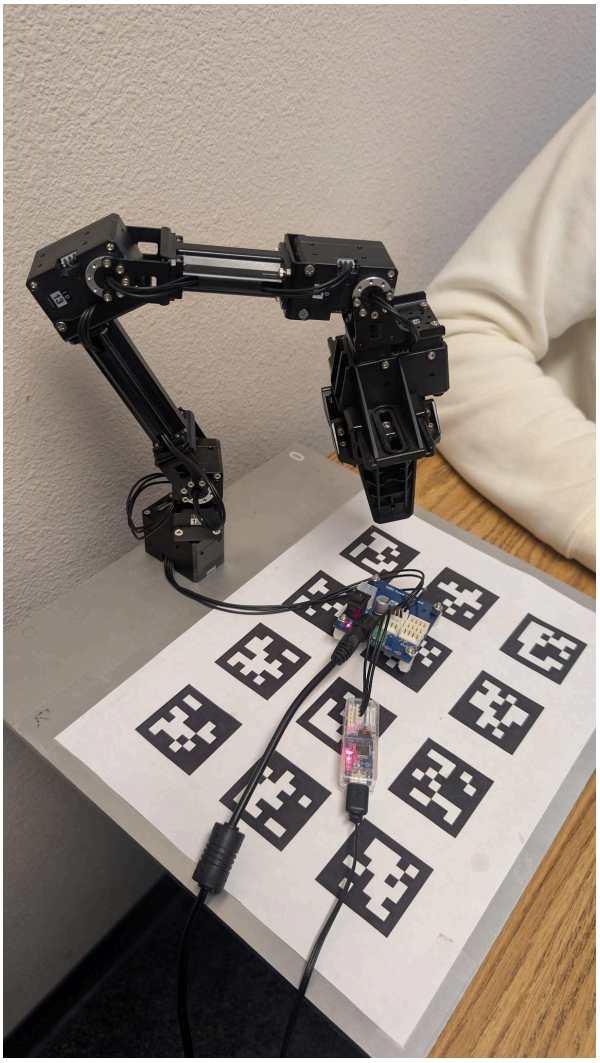

Figure 0: Final Robot Arm configuration

1) Joint Profile Plots (10 s)

The figure shows the time evolution of the four joint angles during the motion from the zero pose to the target pose [45°,−30°,30°,75°].
All joints start from 0° and reach their target positions in approximately 10 s, following smooth sigmoid-like trajectories.
Joint 2 moves in the opposite direction, consistent with its negative target value.
No steady-state offsets are observed; all trajectories converge accurately to their desired angles without noticeable overshoot.
Joints 1, 3, and 4 exhibit similar slopes, indicating coordinated kinematic coupling among the joints.

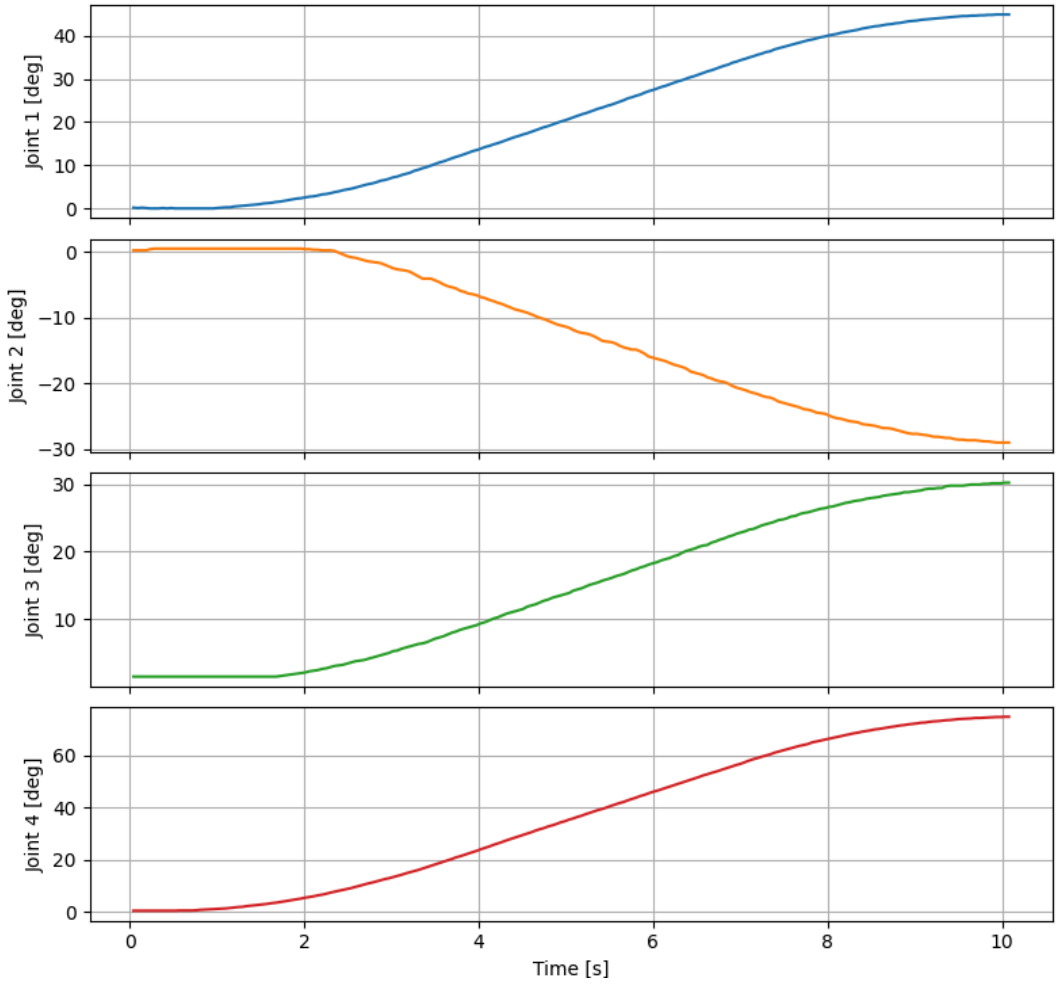

Figure 1: Joint Profile Plots (10 s trajectory time)

2) Joint Profile Plots (2 s)

Compared to the previous 10 s motion, the 2 s trajectory is much faster, with steeper slopes and higher angular velocities.
Despite the reduced duration, the profiles remain smooth and monotonic, showing that the controller preserves trajectory shape and stability.
No overshoot or oscillations appear, though transient acceleration is visibly higher due to the shorter motion time.

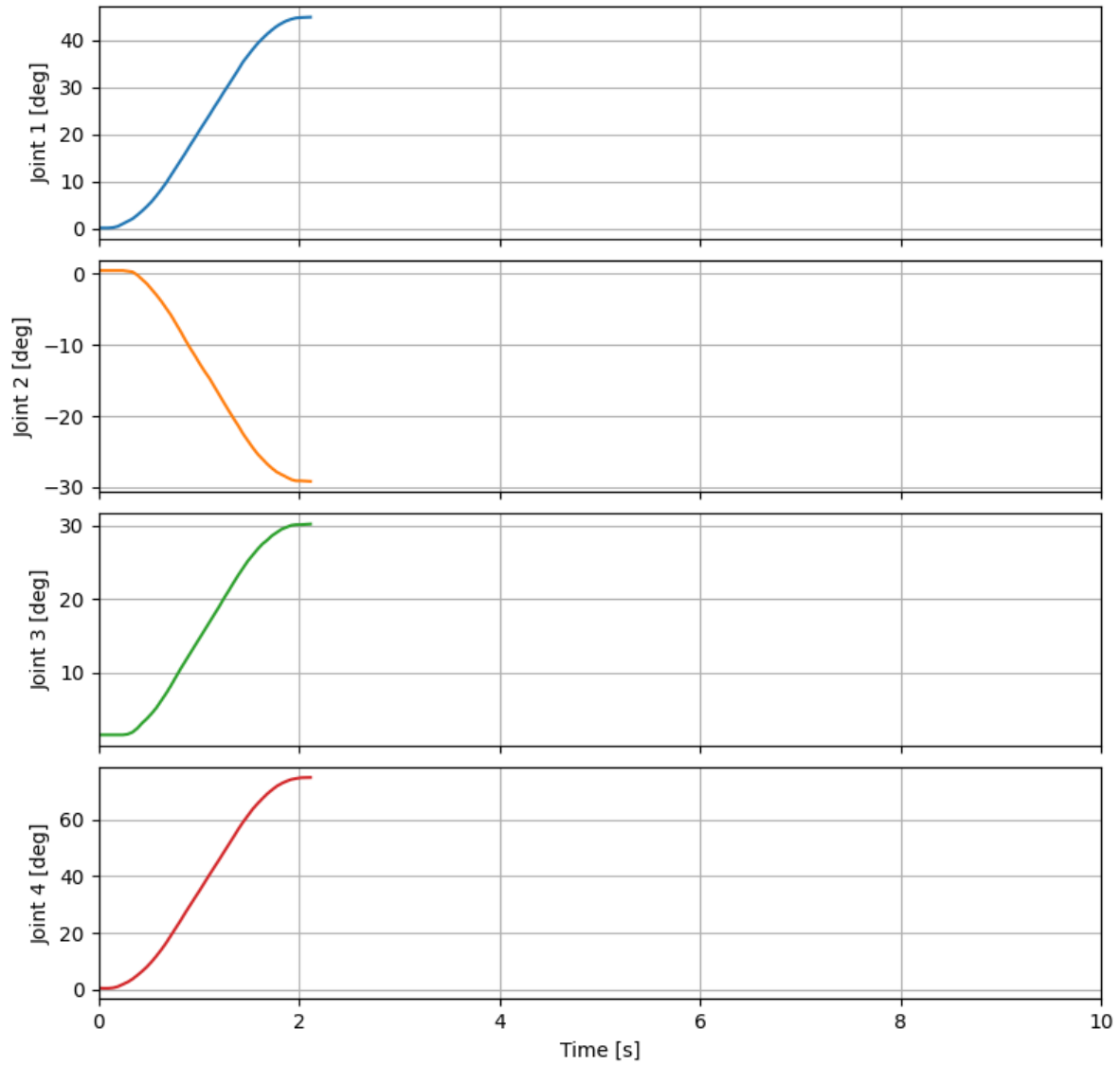

Figure 2: Joint Profile Plots (2 s trajectory time)

3) Histogram of sampling (10 s)

The histogram shows the distribution of the sampling time step ΔtΔt.

Values cluster around 0.048 s with small variations (±0.0002 s), confirming a stable sampling frequency of roughly 20 Hz. The distribution is normal, no outliers.

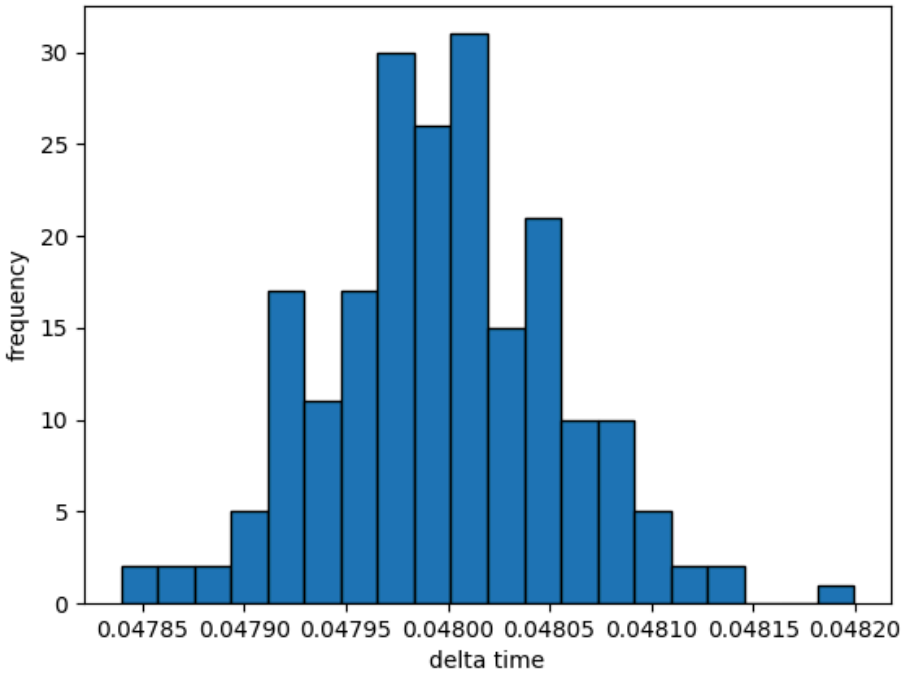

Figure 2: 10 second trajectory time Histogram

Output for the 10s run:
“the run had **209** sample, with a mean of **0.0480** , a median of **0.0480**, min and max respectively **0.0482** and **0.0478**, and a standard deviation of **0.0001**”

4) Histogram of sampling (2 s)

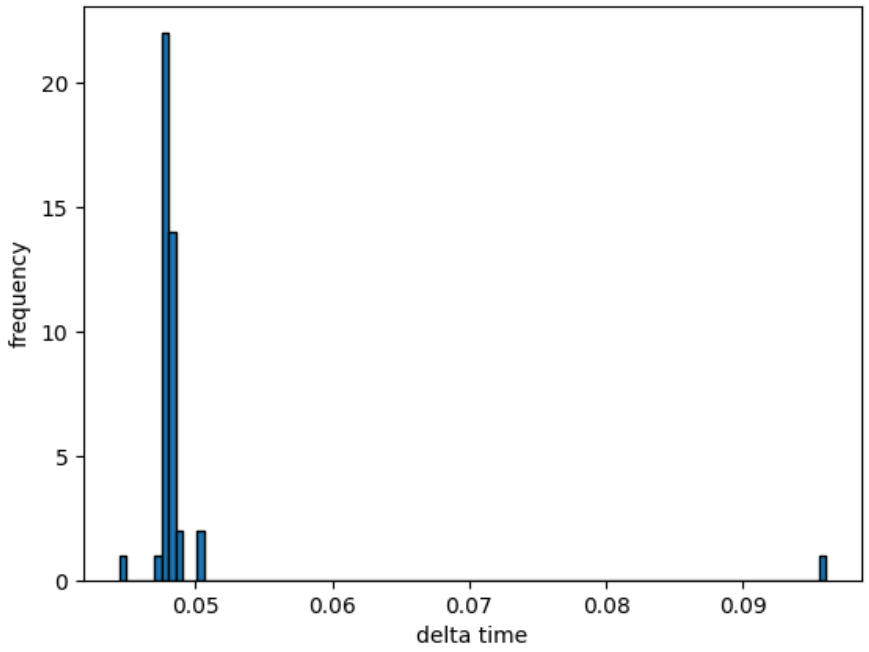

Figure 3: 2 second trajectory time Histogram

Since the raw time data contained visible outliers that distorted the histogram and made the analysis less readable, a filtering function was implemented to remove them.

The function computes the time differences between consecutive samples (np.diff(t)) and determines the first and third quartiles (Q1 and Q3). Using these, it calculates the interquartile range (IQR = Q3 − Q1) and removes all values lying outside the interval [Q1 − 1.5×IQR, Q3 + 1.5×IQR].

After filtering, the resulting distribution of time steps becomes approximately Gaussian, centered around the expected mean sampling interval, with symmetric tails. This confirms that the quartile-based method effectively eliminates abnormal timing spikes while preserving the core structure of the data.

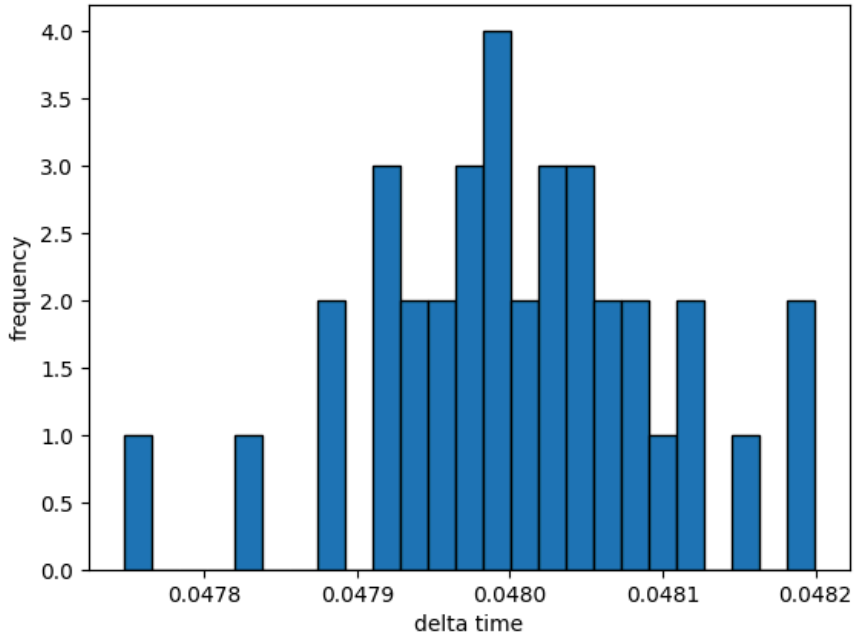

Figure 4: 2 second trajectory time Histogram, without outliers

Output for the 2s run:
“the run had **43** sample, with a mean of **0.0492** , a median of **0.0480**, min and max respectively **0.0961** and **0.0444**, and a standard deviation of **0.0073**”

Sampling Statistics Comparison

Run Duration	Samples	Mean [s]	Median [s]	Min [s]	Max [s]	Std Dev [s]
2 s	43	0.0492	0.0480	0.0444	0.0961	0.0073
10 s	209	0.0480	0.0480	0.0478	0.0482	0.0001

Discussion

Our results met the core goals: we sent a single coordinated command to all four joints, logged angles with timestamps, plotted time series and Δt histograms, and reproduced everything from pickle. The 10 s move was smoother with tighter Δt clustering, while the 2 s move reached the same targets but showed higher peak speeds, sharper transients, and slightly more timing jitter—an expected trade-off. Limitations included non-real-time Python timing (Δt spikes), sequential reads across joints (not perfectly simultaneous), small steady-state offsets, and only indirect control over the Dynamixel time profile. Despite these, final angles were consistent and the saved-data replots matched live figures, so the pipeline is reliable. For improvements, we'd try bulk/asynchronous reads to reduce overhead, thread or priority tweaks to cut jitter, tracking-error plots (commanded vs measured) with rise/settling metrics, focused Δt histograms/CDFs, richer metadata and CSV export, and testing other profiles (e.g., S-curve) and the gripper in coordinated moves. Overall, the data confirms the interface works and clarifies the smoothness-vs-responsiveness trade-off that will guide later labs.

Conclusion

In Lab 1 we took what we built in Lab 0 and turned it into a working joint-space control and logging setup for the robot arm. Using our Lab 0 code as a base (with a few small additions), we sent one coordinated move to all four joints and logged each joint angle and its timestamp while the arm moved to the target pose. The 10 s run looked smooth and finished close to the targets, and the sampling intervals were pretty consistent. When we ran the same move in 2 s, the arm still hit the targets but with higher speeds, sharper starts/stops, and a little more spread in the sampling intervals, which makes sense for the shorter profile. Saving the angle arrays and timestamps let us regenerate the plots later and compare runs easily. Overall, the setup works and gives us a clear way to test motions, measure timing, and organize results for future labs.