ArmLink

Sim2Real Robo Arm Control

Control a physical robotic arm from Unity using VR controllers, mouse, or keyboard. Includes inverse kinematics, live two-way sync with your servos, and a pose sequencer.

> **Documentation version 1.0**
> Covers ArmLink for Unity 2021.3 LTS and newer. For updates and support, visit the documentation page linked in the asset store by AiKodex.

1. Overview

ArmLink connects a Unity scene to a physical robotic arm. You move a 3D model in Unity, and the real arm follows. You move the real arm, and Unity updates to match.

The package handles the servo communication, the inverse kinematics, and the input handling, so you can focus on reinforcement learning, training and what you want the arm to do rather than how to talk to it.

What is included

- Direct servo control over USB serial, no external libraries needed
- A CCD inverse kinematics solver for reaching toward a target point
- Mouse gizmos that render in the Game view, not just the Scene view
- Full keyboard control scheme
- Meta Quest 2 teleoperation over a USB cable
- A pose sequencer for recording and replaying movements
- An in-game control panel to switch between all of the above

Supported hardware

ArmLink is built for 6-DOF arms using Feetech STS-series serial bus servos, such as the SO-101 and similar open source designs. The servos connect in a daisy chain to a USB-to-TTL adapter, which appears as a standard COM port on your machine.

Component	Requirement
Servos	Feetech STS3215 or compatible STS-series
Adapter	Waveshare bus servo adapter or equivalent USB-to-TTL

Component	Requirement
Power	12V for the follower arm, 5V for a leader arm
Unity	2021.3 LTS or newer
API Level	.NET Framework (not .NET Standard)

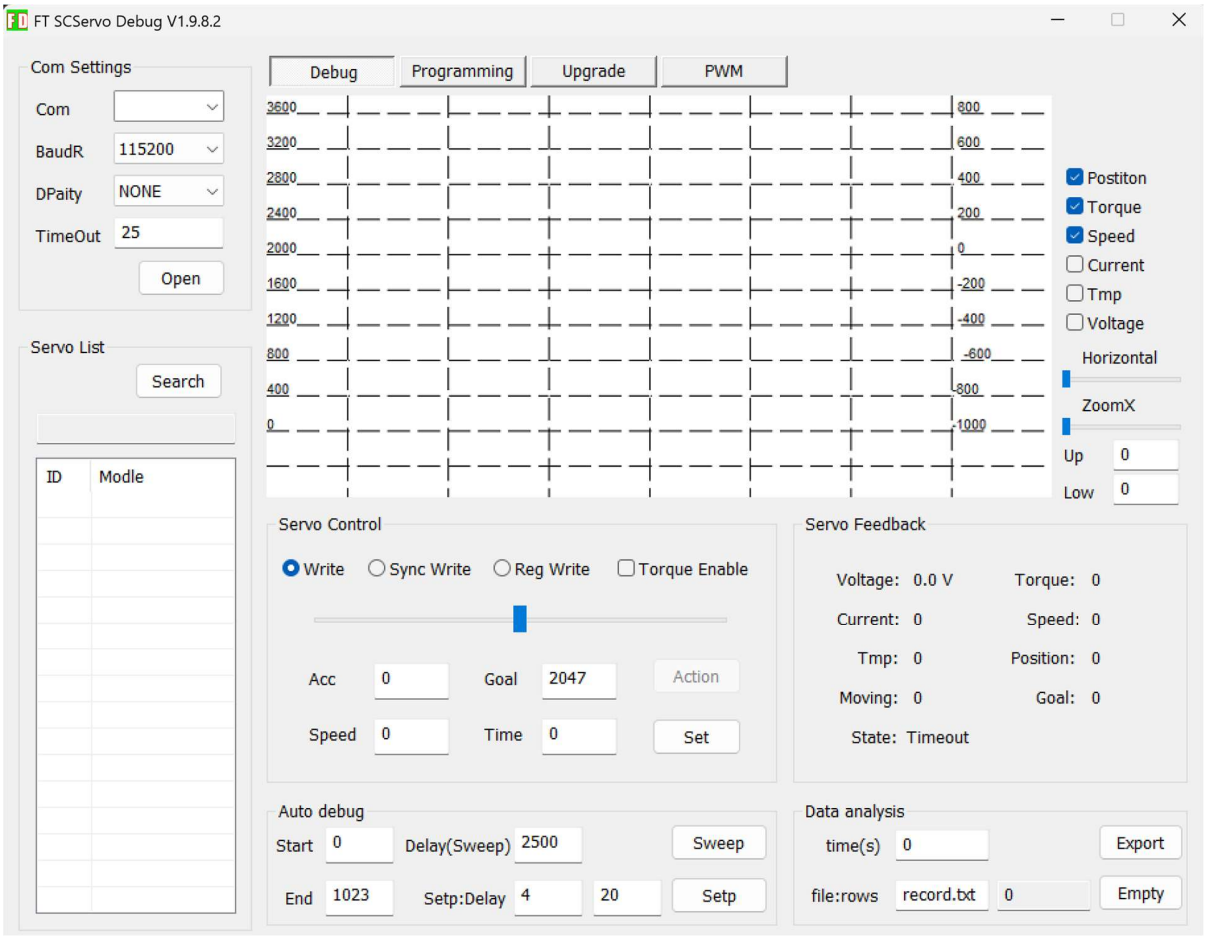
> **Important: API Compatibility Level**
> ArmLink uses System.IO.Ports for serial communication, which is not available in .NET Standard. Go to Edit > Project Settings > Player > Other Settings and set Api Compatibility Level to .NET Framework before using the package.

2. Getting Started

2.1 Before you open Unity

Your servos need unique IDs before ArmLink can talk to them individually. Every Feetech servo ships with ID 1, so if you daisy chain them straight out of the box, the controller cannot tell them apart.

Use the Feetech FD software (free, using FD1.9.8.2) to assign IDs one servo at a time, with only that servo connected:



Servo ID	Joint	Rotation Axis
1	Shoulder pan	Z
2	Shoulder lift	X
3	Elbow	X
4	Hand	X
5	Gripper	Y
6	Thumb	X

Once all six have unique IDs, daisy chain them together starting from the shoulder pan at the controller board end.

2.2 Finding your COM port

Open Device Manager on Windows and look under Ports (COM & LPT). Your adapter will appear as a COM port, usually COM3 or higher. If nothing shows up, you may need the CH340 or CP2102 driver for your adapter board.

2.3 Scene setup

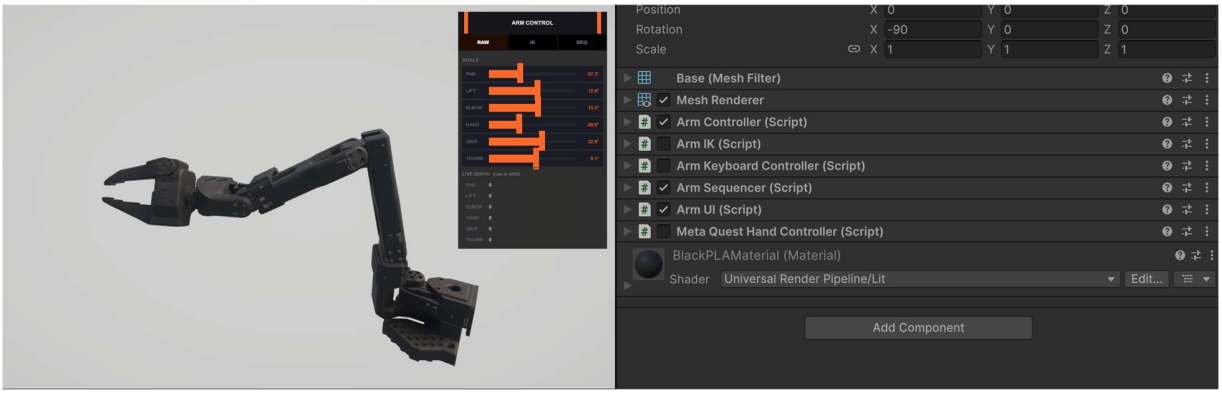
1. Create an empty GameObject and name it ArmRig.
2. Add the ArmController component to it.
3. Set Port Name to your COM port and leave Baud Rate at 1000000.
4. Drag your six joint transforms from the arm model into the joint slots.
5. Add the ArmIK component to the same GameObject and drag ArmController into its reference slot.
6. Add IKTargetHandle to your Main Camera and drag ArmIK into its slot.
7. Add ArmKeyboardController to any GameObject and assign ArmIK.
8. Add ArmSequencer and assign both ArmController and ArmIK.
9. Add ArmUI to any GameObject and assign all five components.

> **Joint hierarchy**
> The joint transforms must be nested in the same order as the physical arm: shoulder pan contains shoulder lift, which contains elbow, and so on. ArmLink reads local rotations, so a flat hierarchy will not work correctly.

2.4 First run

Press Play. The Console should log a connection message showing your COM port. In the on-screen panel, switch to the RAW tab and drag a slider. The corresponding joint should move on both the model and the physical arm.

If nothing moves, check the Console for the connection status and confirm your power supply is connected. USB alone does not power the servos. If the raw sliders on screen do not work, or jitter and snap back, that is because it is fighting against the IK, disable the ArmIK script manually.



3. Core Components

3.1 ArmController

The foundation of the package. Handles the serial connection, packet construction, and two-way sync between Unity transforms and physical servos.

Key settings

Field	Description
`portName`	Serial port, for example COM3
`baudRate`	Leave at 1000000 for STS servos
`speed`	Servo movement speed, 1 is slow and 254 is fastest
`threshold`	Minimum rotation change in degrees before a command is sent
`refreshInterval`	How often live positions are read back, in seconds
`rom_X_JointName`	Min and max safe servo positions per joint

How the sync works

Rotating a joint transform in the Scene sends a command to the matching servo. Dragging a goal slider in the Inspector updates the transform and sends the command. Live servo positions are read back on a timer and shown in the read-only position fields.

Angles map to servo counts linearly, with 0 degrees at the centre position of 2047, and plus or minus 90 degrees spanning 1023 counts either side.

> **Range of motion limits**

> Set the rom values for each joint after testing the physical range in FD software. Commanding a servo beyond its physical stop causes it to draw stall current continuously, which heats up the motor and gears. The rom clamps prevent this.

Public methods

Method	Purpose
SendGoalDirect(id, goal)	Send a position to one servo, bypassing transform detection
ExecutePose(g1...g6)	Set all six joints at once, clearing any external locks
ReadAllPositions()	Read live positions from all six servos
SnapshotAllTransforms()	Reset the change-detection baseline for all joints

3.2 ArmIK

A Cyclic Coordinate Descent solver that rotates the shoulder pan, shoulder lift, and elbow to bring the arm's tip toward a target position. The hand, gripper, and thumb are deliberately left out of the chain so you can control them manually without the solver fighting you.

On Play, ArmIK creates a small sphere at the current tip position. Move that sphere and the arm follows it.

Field	Description
`iterations`	CCD passes per frame, 15 is usually enough
`tolerance`	Stop solving when within this distance of the target
`maxStepDeg`	Maximum degrees a joint moves per pass, lower is smoother
`targetSize`	Radius of the generated target sphere

3.3 IKTargetHandle

Draws move and rotate gizmos directly in the Game view so you can position the IK target and rotate individual joints without switching to the Scene view. Attach this to your Main Camera.

Available handles

Handle	Colour	Action
X / Y / Z arrows	Red, green, blue	Move the IK target along one axis
XY / YZ / XZ quads	Tinted	Move freely on a plane
Hand ring	Red	Rotate the hand joint
Gripper ring	Blue	Rotate the gripper
Thumb ring	Green	Open and close the thumb

The influence setting scales how much a drag rotates the joint. Values below 1 give finer control, above 1 amplifies your movement.

4. Control Modes

ArmLink includes three control modes, and only one is active at a time. The in-game panel handles the switching, releasing any locks the previous mode held so the modes never fight each other.

4.1 Raw mode

Six sliders, one per joint, mapping directly to servo angles. Below them, a live readout shows the actual position each servo reports back, in raw 0 to 4095 counts.

Inverse kinematics is disabled in this mode, so the sliders have uncontested control. This is the mode to use when testing individual joints or finding the physical range of motion for your rom settings.

4.2 IK mode

Two toggles let you pick your input method, and both can be on at once.

Keyboard controls

Key	Action
W / S	Move target forward and back
A / D	Move target left and right
Left Shift / Left Ctrl	Move target up and down
Q / E	Rotate the gripper
R / F	Tilt the hand up and down
Space	Toggle the thumb open and closed

Move speed and rotation speed are adjustable from the settings sliders in the IK panel.

Mouse gizmos

Enable this toggle and the move and rotate handles appear on the IK target and joints. Drag an arrow to slide along one axis, drag a plane quad to move in two dimensions, or drag a coloured ring to rotate a joint.

4.3 Sequencing mode

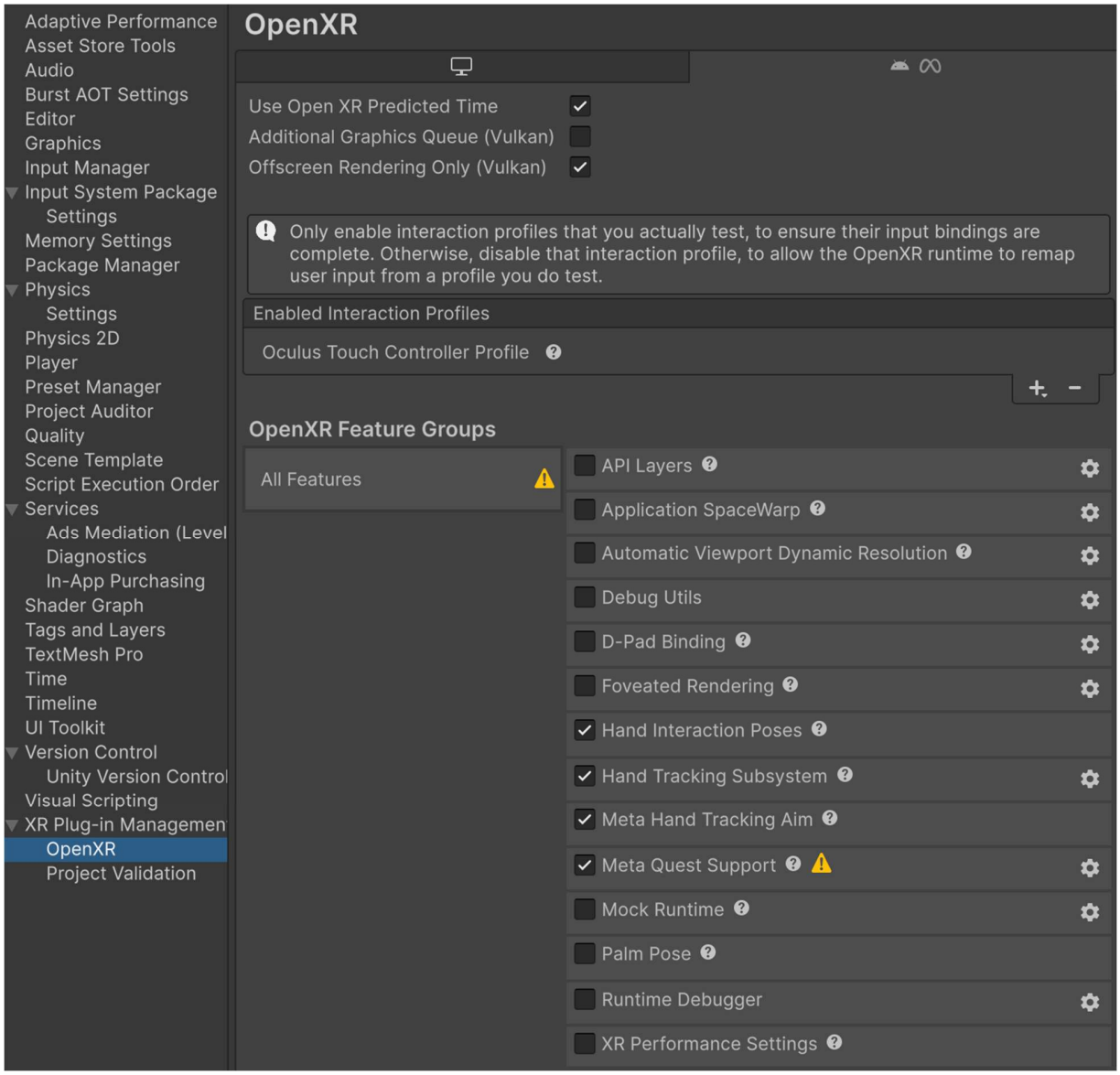
Record a series of poses and play them back. Each captured pose stores all six joint angles plus a duration, which controls how long the arm takes to reach that pose from the previous one.

1. Move the arm into position using any control method.
2. Switch to the SEQ tab and press Capture Pose.
3. Repeat for each pose in your sequence.
4. Adjust durations in the Inspector if needed.
5. Press Play to run the sequence.

Playback uses smoothstep easing so the arm accelerates and decelerates naturally rather than moving at a constant speed. Inverse kinematics is suspended during playback.

5. VR Teleoperation

ArmLink supports controlling the arm with a Meta Quest 2 controller. Rather than requiring Meta Link, which has GPU requirements many machines do not meet, ArmLink streams controller data over the USB cable using ADB port forwarding.



5.1 How it works

A small companion app runs on the headset and broadcasts controller position, rotation, and trigger data on a TCP port. ADB forwards that port over USB to your PC, where ArmLink reads it and drives the arm.

```
Quest 2 (companion app)        USB + ADB         PC (your Unity project)
QuestControllerSender    →   adb forward   →   MetaQuestHandController
  reads controller           tcp:7890            drives IK target
```

5.2 Setting up the companion app

1. Create a new Unity project for the companion app.
2. Install the OpenXR and XR Management packages.
3. Enable OpenXR and Meta Quest Support in the Android tab of XR Plug-in Management.
4. Add QuestControllerSender to a GameObject in an empty scene.
5. Optionally add PassthroughSetup to see your surroundings.
6. Build as an Android APK.

```
adb install YourCompanionApp.apk
adb forward tcp:7890 tcp:7890
```

> **Passthrough setup**
> If you use PassthroughSetup, select the OVRManager component in the Inspector and set Passthrough Support to Required before building. Without this the Android manifest will not declare passthrough and you will see a black screen.

5.3 Controls

All tracking is gated behind the A button. Hold it to drive the arm, release to freeze. Each press recalibrates the origin, so the arm never jumps when you re-engage.

Input	Action
A button (hold)	Activate tracking and recalibrate
Move controller	IK target follows in the same direction
Yaw (turn wrist left/right)	Shoulder pan
Pitch (tilt up/down)	Hand joint
Roll (tilt sideways)	Gripper rotation
Trigger	Thumb, mapped continuously from 0 to 90 degrees

Sensitivity for position and each rotation axis is adjustable in the Inspector, so you can tune how much wrist movement translates into joint rotation.

6. Troubleshooting

Nothing connects

- Check that Api Compatibility Level is set to .NET Framework, not .NET Standard
- Confirm the COM port in Device Manager matches what you entered
- Make sure the power adapter is connected, USB alone does not power the servos
- Close any other software that might be holding the port open, including FD

Only one servo responds

All Feetech servos ship with ID 1. If you daisy chained them before assigning unique IDs, they all respond to the same address. Disconnect them, assign IDs one at a time in FD software, then reconnect the chain.

A joint moves the wrong direction

Some servos are mounted mirrored relative to their neighbours. The gripper and thumb use inverted mapping by default. If another joint needs inverting, the mapping is a single sign flip in the angle-to-servo conversion for that joint.

Joints jitter or fight each other

This usually means two systems are commanding the same joint. Switching modes in the panel releases all locks automatically, but if you are calling the API directly, use SnapshotAllTransforms after any external control to reset the change-detection baseline. Manually disabling the ArmIK script usually works.

The arm reaches but the hand tilts oddly

The IK solver only controls the shoulder and elbow. The hand angle is whatever you last set it to. Use the red ring or the R and F keys to adjust it independently.

VR shows a black screen

Three things must all be correct for passthrough: the Android manifest must declare it, which comes from setting Passthrough Support to Required on OVRManager; the camera background alpha must be zero; and the passthrough layer must be set to Underlay so it renders behind your scene. PassthroughSetup handles the last two automatically.

VR connects but the arm does not move

- Confirm the companion app is running on the headset, not just installed
- Re-run adb forward after reconnecting the cable, the forward does not persist
- Hold the A button, tracking is gated behind it

7. Script Reference

A summary of every component in the package and what it does.

Script	Attach to	Purpose
ArmController	Arm root	Serial communication and two-way joint sync
ArmIK	Arm root	Inverse kinematics solver and target sphere
IKTargetHandle	Main Camera	Game view move and rotate gizmos
ArmKeyboardController	Any	Keyboard input handling
ArmSequencer	Any	Pose recording and playback
ArmUI	Any	In-game control panel
MetaQuestHandController	Any	Receives VR controller data over TCP
QuestControllerSender	Companion app	Streams controller data from the headset
PassthroughSetup	Companion app	Enables headset passthrough
ReadOnlyAttribute	N/A	Inspector attribute for display-only fields

Servo protocol notes

For anyone extending the package, ArmLink speaks the Feetech STS binary protocol directly. Packets are structured as two header bytes, servo ID, length, instruction, register address, data bytes, then a checksum. Position is read from register 0x38 and written to 0x2A, with torque enable at 0x28.

The bus is half duplex, so a short delay is needed between sending a command and reading the response. ArmLink uses 3 milliseconds, which works reliably at 1 Mbps.

Known limitations

- Supports Feetech STS-series servos only, other brands would need a new protocol layer
- VR tested on Meta Quest 2 over USB, other headsets are untested
- The VR companion app must be built and installed separately
- IK solves three joints, the hand, gripper, and thumb are manual
- Sequencer interpolation is linear between poses, no curve editing yet