Quick Start Guide – EduSCARA

26/05/2025

# Introduction

The EduSCARA is still under development. This guide is intended to help you get the manipulator up and running as a demonstration of its capabilities in its current state. Rather than using this guide to develop a full application, please use it to observe and understand the current shortcomings and limitations of the EduSCARA system.

# Current limitations – what to expect

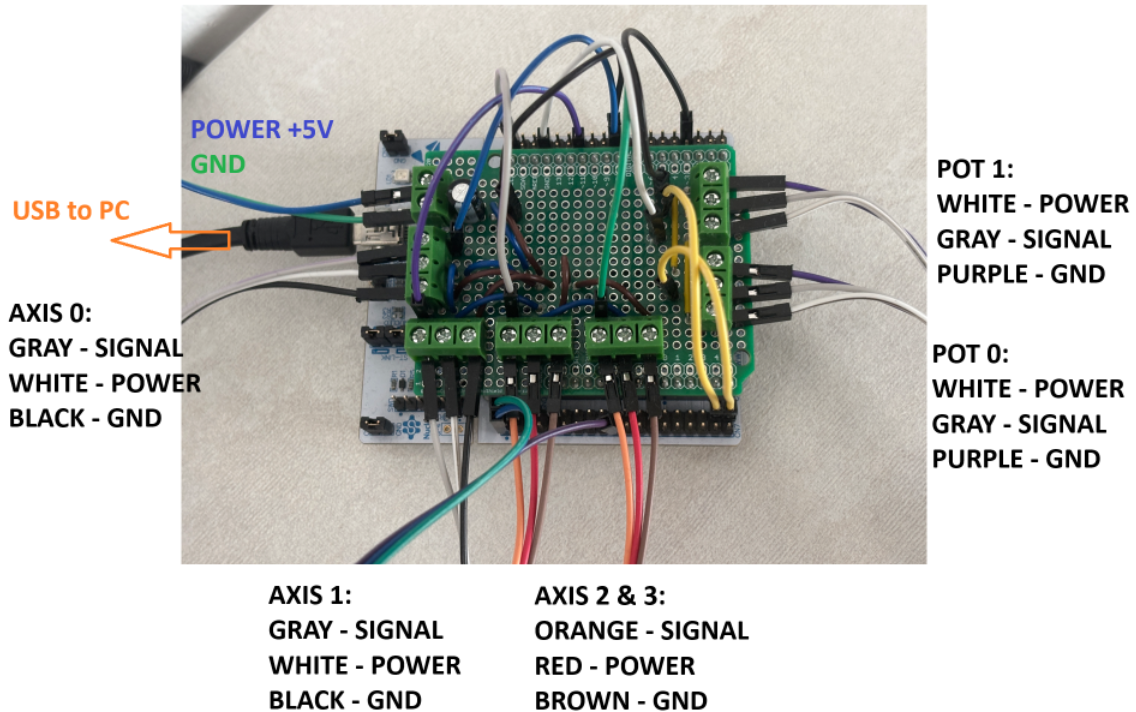
As mentioned in the report, there are limitations to the manipulator’s accuracy due to backlash in the servos and noise from the sensors. While backlash mitigation was discussed, replacing the servos with stepper motors was identified as the most robust solution.

However, an additional issue has been observed today (26/05/2025): the analog potentiometer readings used for joint position feedback are interfering with one another. Specifically, changes in the voltage on one potentiometer are causing consistent shifts in the readings from the other, even when its physical position of the shifted potentiometer remains unchanged. This interference is likely caused by the high output impedance of the potentiometers, which is unsuitable for directly driving ADC inputs.

This results in cross-talk between channels, where one reading is slightly dependent on the state of the other. Typical mitigation strategies such as delays between conversions, star-ground wiring, and bypass capacitors were tested but did not fully resolve the issue. Another potential solution would be to buffer each potentiometer output using an op-amp configured as a unity-gain buffer, providing a low-impedance source suitable for the ADC. Alternatively, replacing the potentiometers entirely with incremental encoders would eliminate the analog signal integrity issues altogether.

As a result, the EduSCARA in its current state has reduced accuracy in real-world positioning, particularly at larger joint angles until this sensor interference issue is resolved. The remainder of this document will now focus on how to operate the system.

# Wiring



# Tools

* STM32CubeIDE
* Visual Studio Code
* Unity

# Demo sequence

The built-in demo sequence can be activated by pulling PB7 down to GND, using a jumper:

A computer screen shot of a computer

AI-generated content may be incorrect.

# Potentiometer calibration

The accuracy of the EduSCARA is ultimately limited by the precision of its position sensors. Therefore, proper calibration of the potentiometers is essential. Currently, calibration requires the user to manually move each joint to predefined positions indicated by calibration markers, and record the corresponding raw ADC values. These values must then be entered into the firmware.

To begin, open the **servo\_motion\_controller** project in STM32CubeIDE. You will primarily be working with two source files: **application.c** and **potentiometers.c**. In **application.c**, comment out or remove the **API macro**, then start a **debugging session** to proceed with calibration.

A screenshot of a computer

AI-generated content may be incorrect.

Then, in **Live Expressions** tab, add the **adc\_raw** array to the list:

A screenshot of a computer

AI-generated content may be incorrect.

Now switch to **potentiometers.c** and locate the **potentiometers\_init()** function. You will use this function to manually calibrate each joint using the ADC values from the potentiometers.

* First, rotate **Axis 0** to **-90°** using the calibration marker as a guide. Note the corresponding value in **adc\_raw[0]**, and assign it to **pots[0].min\_raw\_value**.
* Then, rotate **Axis 0** to **+90°**, record **adc\_raw[0]** again, and assign it to **pots[0].max\_raw\_value**.
* Next, rotate **Axis 1** to **0°**, record **adc\_raw[1]**, and assign it to **pots[1].min\_raw\_value**.
* Finally, rotate **Axis 1** to **+90°**, record the new **adc\_raw[1]** value, and assign it to **pots[1].max\_raw\_value**.

A screenshot of a computer program

AI-generated content may be incorrect.

Save the modified files, return to **application.c**, and re-enable the **API macro**. Then rebuild and flash the firmware to the microcontroller.

# Using the Motion Controller Python API

Simply place the **scara\_motion\_controller\_api.py** script in your working directory, and call any of the available functions. The script below establishes a connection to the Motion Controller, then issues initialisation, calibration, move, and read commands.

A screen shot of a computer program

AI-generated content may be incorrect.

# Using the Simulator and Python API

Open the **mini\_scara\_simulator** project in Unity, play the scene and start the server.

A computer screen shot of a computer program

AI-generated content may be incorrect.

Now, run the **sim\_resistor\_application.py** script:

A computer screen shot of a machine

AI-generated content may be incorrect.

The Simulator will run only while Unity is active window.