

Automation Script Editor – Web Application

The screenshot shows a web application titled "AUTOMATION SCRIPTS" on a dark background. At the top, it says "Deleting mission..." above a text input field containing "Input json cmd here.". Below the input field are four buttons: "Set:a", "Set:b", "Set:c", and "Set:d". A red button labeled "Set:On Boot" is positioned below these. A larger grey button labeled "STOP MISSION" is centered below the red button. The section "Run Mission" follows, with four buttons: "Run:a", "Run:b", "Run:c", and "Run:d". A red button labeled "Run:On Boot (Once)" is below them. The "Delete Mission" section has four buttons: "Del:a", "Del:b", "Del:c", and "Del:d". A red button labeled "Del:On Boot" is at the bottom.

The Web App includes a built-in **Automation Scripts** editor that allows you to combine multiple JSON commands into a single mission file (`*.mission`).

These scripts are stored in the ESP32-S3's internal flash and **persist across power cycles**, enabling the robotic arm to automatically execute motion sequences after reboot.

1. Script Management Buttons

Set:a / Set:b / Set:c / Set:d

These buttons upload the script you composed into the controller and save it as a mission file.

How it works:

- When you click **Set:a**, the Web App:
 1. Creates a mission file named `a.mission` on the controller
 2. Writes each JSON command (line by line) into the file
 3. Saves the task for later execution
- You can then run this scripted task anytime using **Run:a**.
- Each button corresponds to:

- `a.mission`
- `b.mission`
- `c.mission`
- `d.mission`

This mapping is useful if you later trigger missions from JSON commands instead of the Web UI.

On Boot

This saves the script as `boot_user.mission`, which runs **automatically on every startup** and loops indefinitely in a background thread.

Other system functions remain unaffected.

STOP MISSION

Stops any mission currently running.

Useful when:

- You configured a boot script

- The arm starts running automatically
 - You want to immediately halt motion
-

Delete Mission

Buttons in this section delete the corresponding mission files from flash memory.

Automation Script Example



In this example, we will teach the robot to trace a square in 3D space (like the illustration below).

We will compose a sequence of JSON commands, upload them as `a.mission`, and play them back.

Step 1 — Open the Automation Scripts Editor

Inside the **Automation Scripts** component (not the *JSON Interface*), drag the bottom-right corner of the text box to enlarge the editor area:

AUTOMATION SCRIPTS

Mission upload status.

Input json cmd here.

Set:a

Set:b

Set:c

Set:d

Set:On Boot

STOP MISSION

Run Mission

Run:a

Run:b

Run:c

Run:d

Run:On Boot (Once)

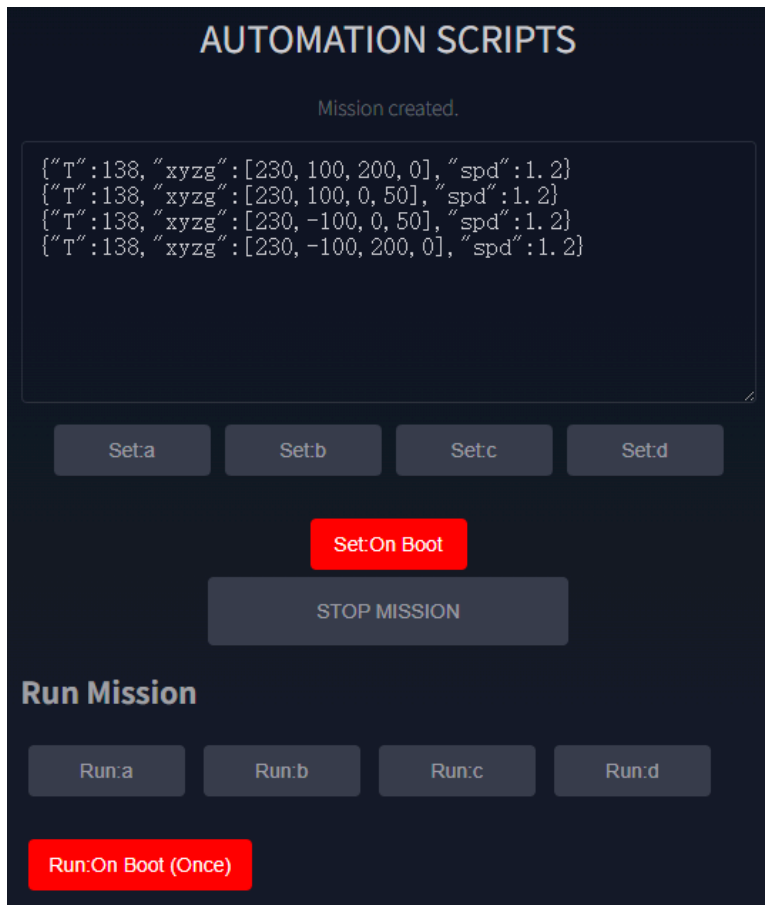
Step 2 — Enter the JSON Commands

Each line corresponds to one waypoint of the square:

💡 Important:

- One JSON command **per line**
- English punctuation only
- No spaces

```
{"T":138,"xyzg":[230,100,200,0],"spd":1.2}
{"T":138,"xyzg":[230,100,0,50],"spd":1.2}
{"T":138,"xyzg":[230,-100,0,50],"spd":1.2}
{"T":138,"xyzg":[230,-100,200,0],"spd":1.2}
```



Optional — Add Delays Between Points

If you want the robotic arm to pause briefly at each corner, insert delay commands:

```
{"T":138,"xyzg":[230,100,200,0],"spd":1.2}
{"T":51,"delay":1500}
{"T":138,"xyzg":[230,100,0,50],"spd":1.2}
{"T":51,"delay":1500}
{"T":138,"xyzg":[230,-100,0,50],"spd":1.2}
{"T":51,"delay":1500}
{"T":138,"xyzg":[230,-100,200,0],"spd":1.2}
{"T":51,"delay":1500}
```

Step 3 — Upload the Mission

Click **Set:a** and wait for the Web App to upload and save the script as `a.mission`.

Step 4 — Run the Mission

To execute the square trajectory:

- Go to **Run Mission**
- Click **Run:a**

The robotic arm will now play the scripted motion sequence.

Step 5 — Edit or Delete the Mission

- To replace the mission → upload a new script using **Set:a**
- To remove it completely → click **Del:a**