Nice — congrats on getting the code base ready. Below is a **clear, step-by-step runbook** you can follow in Webots to get the whole simulator (world + PROTOs + advanced controller) running end-to-end. I'll include exact UI steps, terminal commands you may need, debugging checks and tips for tuning. Do each step in order.

Quick sanity checklist (before starting)

Make sure your project folder contains these exact folders/files (names matter):

```
<ProjectRoot>/
  controllers/
    stair_cleaner_advanced/          <-- controller folder
name MUST match PROTO controller
      stair_cleaner_advanced.py
      planner_utils.py
  protos/
    StairClimbCleaner.proto
    RealisticStairs.proto
    DirtPatch.proto
  worlds/
    StairClimbCleanerProject.wbt
```

In `StairClimbCleaner.proto` the controller line must match:

```
controller "stair_cleaner_advanced"
```

If you chose a different controller name, use that exact name everywhere.

Step-by-step: Open & run in Webots

1. **Open Webots.**
2. **Point Webots to your project folder (so relative paths work):**
   Menu → **File** → **Open Project Directory...** → choose `<ProjectRoot>` (the folder that contains `controllers`, `protos`, `worlds`).
3. **Open the world file:**
   Menu → **File** → **Open World...** → `worlds/StairClimbCleanerProject.wbt`.
4. **Confirm the robot controller setting (optional check):**
   - In the **Scene Tree** (left panel) expand the Robot node (e.g. `StairClimbCleaner` or `CleanerBot`).
   - Click the Robot node and look at the **Fields / Device** panel on the right. There is a `controller` field. It should show `stair_cleaner_advanced`.

- If it's different, set it exactly to `stair_cleaner_advanced` (no `.py`, no paths). Save the world.

5. **(Optional) Verify that PROTOs are visible:**
   - In Scene Tree → right-click → Add → PROTOs, you should see `RealisticStairs`, `DirtPatch`, `StairClimbCleaner`. If not, re-check that `protos/` is inside project root.
6. **Install Python dependencies (if using advanced controller):**
   - Open a terminal and run:
     ```
     pip install --user numpy pillow
     ```
   - ```
     # optional for faster camera dirt detection:
     pip install --user opencv-python
     ```
   - If your system Python is different from Webots', run the pip command for the Python used by Webots. (Usually `pip install --user ...` works. If you hit ImportError for numpy or cv2 when running, that means Webots' Python can't find them—install to the same interpreter.)
7. **Run the simulation:**
   - Click ▶ (Play). Webots will compile/load PROTO and launch the controller.
   - Open the **Console** (bottom) to see printouts from the controller (e.g. `[DIRT]`, `[STATE]`, map saving messages).
8. **Open camera views and LIDAR visuals:**
   - **Camera:** In Scene Tree expand your robot → `front_cam` → right-click → *Display Camera*. A video window appears.
   - **Follow camera:** Top-left camera dropdown → choose `FollowCam` (or use the TrackingCamera in the world).
   - **Lidar point cloud:** In Scene Tree expand robot → `lidar`. If Webots supports, right-click `lidar` → *Show Point Cloud* or check "Point cloud" in the device properties. (If not visible, check the GUI options or the Lidar node's fields.)
9. **Watch console outputs** and logs: controller prints, map saved path (`/tmp/stair_map.png` by default), dirt count, state transitions, errors. The console is the primary place to debug.

Quick functional smoke test (first run)

- With the world loaded and the controller running:
     ```

1. Robot should move forward slowly (CLEANING state). Brush motor should spin.
2. If you place a DirtPatch near the robot, the advanced controller should detect it (camera/lidar) and eventually plan → navigate → clean.
3. If robot sees stairs (bottom distance sensor), it should transition to DETECT_STAIRS → CLIMB flow.
4. Lidar obstacles should trigger AVOID behavior (back + rotate).

If nothing moves, check console for errors or device-not-found messages.

Troubleshooting & common fixes

1. **Controller not starting / "controller not found"**
   - Verify `controllers/<name>/` folder exists and contains `<name>.py`. The folder name must match the `controller "..."` string in the PROTO exactly.
   - Example: controller line `"stair_cleaner_advanced"` → folder `controllers/stair_cleaner_advanced/` and file `stair_cleaner_advanced.py`.
2. **ImportError: numpy / cv2**
   - Install packages for the Python interpreter used by Webots (see step 6). If unsure, run `python3 -m pip install --user numpy opencv-python` with the same Python binary Webots uses.
3. **Devices not found (NoneType errors)**
   - Controller prints helpfully when it can't find devices. Open PROTO and confirm device `name` strings exactly match what the controller expects (`left_motor`, `right_motor`, `cleaner_motor`, `front_distance`, `bottom_distance`, `imu`, `lidar`, `front_cam`). Fix names or modify controller to try alternative names.
4. **Robot slips / can't climb**
   - Tune physics in PROTO: reduce robot density, add `CoulombFriction` on wheels and robot body (e.g., 2–3). Adjust wheel friction or motor torque.
5. **Robot flips or tilts badly**
   - Decrease PID gains (Kp/Kd) in the controller or reduce `CLIMB` speed. Add weight/inertia by adjusting `density` in PROTO (but don't overdo it).
6. **Path planner fails to find a path**

    - Increase `MAP_RESOLUTION` (coarser grid) or lower `obstacle_threshold` in planner settings. Make sure your map origin and pose are reasonable.

7. **Map file not saved / can't find `/tmp/stair_map.png`**
    - On Windows, `/tmp` may not exist; change path in controller to a local path inside project (e.g. `os.path.join(os.getcwd(), 'stair_map.png')`) or `C:\Temp\stair_map.png`.

Tuning knobs (where to adjust things quickly)

- Controller constants at top of `stair_cleaner_advanced.py`:
    - wheel geometry: `WHEEL_RADIUS`, `WHEEL_BASE`
    - speeds: `MAX_SPEED`, `ACCEL`, `TURN_SPEED`
    - SLAM: `MAP_RESOLUTION`, `MAP_SIZE_M`
    - PID gains: `kp_lin`, `kp_ang`, etc.
    - Dirt detection thresholds (camera + lidar)

Change values, save the file, then **press Stop/Play** in Webots to reload the controller.

How to record a demo video (quick)

- Webots Menu → **Movie** → **Start Recording** before pressing Play.
- After run, Movie → Stop Recording → export video file.

If you want me to do any of these for you now:

- I can generate a ready-to-paste `run.sh` that:
    - installs deps,
    - opens Webots to the project,
    - tail the controller logs,
    - and periodically copies the map to the project folder.
- Or I can produce a short **README.md** with exact commands for your repo.

Tell me which (run script or README) and I'll give it instantly.