Build PyQt button layout

Description : Create a PyQt window named `W` and add four push buttons: Launch Gazebo, Circular, Forward, Square.

Map each button to a simple callback that prints the button name when clicked to verify wiring.

Map ROS2 commands into a dictionary

Description : Define a Python `dict` that maps each button label to its ROS2 launch command and node name.

Print and inspect each dict entry to confirm commands and node names match expected values.

Implement a non-blocking process starter

Description : Write a function that starts a ROS2 command with `subprocess` in the background and returns its process handle.

Test it by running a lightweight ROS2 command (e.g., `ros2 node list`) to confirm the GUI remains responsive.

Implement graceful stop with pkill fallback

Description : Implement a stop function that sends SIGINT to the process handle, waits ~2 seconds, then sends SIGKILL to the process group if still alive.

Verify behavior by launching a test ROS2 process and, if Gazebo leftovers persist, use `pkill` to remove gazebo-server/client processes.

Add AMCL waypoint save button

Description : Add a GUI button that queries the current AMCL pose and appends the pose to an in-memory list or JSON file as a sequential waypoint.

Drive the robot to two positions, press the button at each, and confirm the saved waypoint coordinates are recorded correctly.