Requirements for Dynamic Terrain Walking Robot

Requirement 1

When only three feet or less are in contact with the ground, **the locomotion system** shall execute a pose adjustment mechanism to regain stable footing while maintaining the current location and orientation.

Potential Design Solutions:

- Monitor the binary switched on each foot to determine if any of the legs is suspended in mid-air in the case of uneven terrain.
- If a leg or more is not touching the ground, execute a behavior for the robot to re-adjust the leg-surface topology in the same position and orientation.
- The mechanism could be done by moving each of the unstable legs separately in vertical motion till it touches the ground. The robot can then re-adjust its whole configuration according to torque values on different joints for maximum stability.

Requirement 2

When roll or pitch angles exceed ±15° or angular velocity exceeds 3 rad/s, **the locomotion system shall** execute corrective foot placements within 0.2s to restore balance and, if necessary, trigger a self-righting mechanism.

Potential Design Solutions:

- Using the built-in IMU sensors, the roll and pitch of each link can be detected. Therefore, we can estimate the incline of each leg.
- In the case of an unstable stance as defined in the requirement, the robot should take time to re-adjust its configuration through a self-righting mechanism.
- The self-righting mechanism can be implemented through zero point estimation to calculate where the weight should shift, and then getting the actual joint states using inverse kinematics.

Requirement 3

When all four feet unintentionally leave the ground (e.g., due to jumping or misalignment), **the locomotion system shall** adjust joint angles mid-air for a stable landing.

Potential Design Solutions:

- In case the robot falls or jumps, the robot should automatically switch to a fixed pose for landing, typically with all legs stretched out to absorb the impact.

Requirement 4

When the center of mass (CoM) deviates significantly from its expected location, **the autonomous system shall** execute a self-righting policy.

Potential Design Solutions:

- The 6D center of Mass can be estimated from the force-torque feedback on each of the joints.
- If the center of mass is found to be on the outside of the robot's body (not directly under it), then the robot is in a highly unstable configuration and should execute the self-righting mechanism.
- The self-righting mechanism was mentioned in requirement 2.

Requirement 5

When climbing an incline, **the autonomous system shall** minimize CoM height and decrease step frequency to enhance stability.

Potential Design Solutions:

- The robot should monitor the surface incline: either mechanically through the joint angles or visually through a depth camera.
- If the incline is more than ±15°, new joint states should be calculated to lower the overall height of the robot to be closer to the ground and decrease the possibility of a topple.
- The height of the CoM could be lowered by increasing knee flexion, which would distribute weight more evenly across the legs.

Requirement 6

When traversing an incline, **the autonomous system shall** adapt its stance with the topology of the terrain to compensate for gravity.

Potential Design Solutions:

- The incline of the terrain can be calculated as mentioned in requirement 5.
- If the incline angle is positive (ascending), the robot can lean forward slightly to counteract gravitational pull.
- If the incline angle is negative (descending), the robot can lean backward to avoid tipping forward.

Requirement 7

When the robot's linear velocity is zero while an actuation command is present, **the locomotion system shall** execute a "non-stuck" mechanism to break free.

Potential Design Solutions:

- Monitor joint force/torque feedback and foot contact sensors to detect excessive resistance or slippage.
- Execute a sequence of recovery motions, such as small lateral sways or backward steps, to shift weight distribution and escape static friction.
- Increase torque temporarily on specific joints to push past minor obstructions while ensuring stability constraints.
- If stuck persists, trigger a higher-level recovery routine such as a full-system reset to ensure safety.

Requirement 8

When a single joint fails to actuate within 50 ms, **the RL agent shall** redistribute force among other legs to compensate.

Potential Design Solutions:

- Detect non-responsive joint commands through real-time feedback from joint force/torque sensors.
- Reallocate load dynamically by increasing torque in the remaining functioning legs.
- Adjust foot placement strategy to shift weight away from the malfunctioning joint.
- Trigger asymmetric gait adaptation (e.g., three-legged stance or increased reliance on diagonally opposite legs).

Requirement 9

When a moving object (e.g., a human or another robot) enters within 1m of the robot's path, **the navigation system shall** signal a slow down to adjust trajectory accordingly.

Potential Design Solutions:

- Use vision-based tracking and motion prediction models to anticipate dynamic obstacles.
- Implement a safe zone radius where the robot reduces speed and prepares for avoidance maneuvers.
- If a fast-moving object is detected, execute a stop-and-reassess behavior to prevent collisions.
- Train the RL policy with realistic pedestrian and robot interactions to improve responsiveness.
- Integrate adaptive speed control algorithms that adjust velocity based on proximity risk assessment.

Requirement 10

When the RL policy selects an action, **the robot system shall** log the contributing state variables and policy decision for post-analysis.

Potential Design Solutions:

- Record key state variables (root orientation, joint angles, velocities, and foot contacts) at each decision step.
- Log the selected action along with a timestamp for synchronization with sensor data.
- Store action-policy mappings to enable retrospective analysis and debugging.
- Implement structured data storage (e.g., ROS2 bag files or structured CSV logs) for easy retrieval.