



URDF MODEL



TRAIN ENV



**DEPLOY ENV** 



STATE TRACKER



## Requirements

### **♦** Compulsory

Env description, task description, estimator, reward mapping table, safety criterion

#### Optional

Reference reward function, reference policy from previous work (teacher), additional prompts, custom observation





**Prompts** 

- Coding Tips
- Regulations
- **Feedback**



References **Custom Configs** 





#### Module 1

class AnybipeRewards(): def \_\_init\_\_(self, env): self.env = env

> def load\_env(self, env): self.env = env

#### \* Tracking Reward Functions

return torch.exp(-lin vel error / self.env.cfg.rewards.tracking sigma)

def \_tracking\_ang\_vel(self):

# Tracking of angular velocity commands (yaw)

ang vel error = torch.square(self.env.commands[:. 2] - self.env.base ang vel[:. 2]) return torch.exp(-ang vel error / self.env.cfg.rewards.tracking sigma)

# Module 2

**Previous** Work

Reference

Policy Supervised



We deployed the trained RL policy to our robot and evaluated its performance in reality. We converted the reward functions that can be calculated by real-world sensors and actuators

here are the results of the evaluation:







Best Model

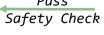
Eval

Feedback



Pass Eval

Homomorphic Evaluation







Estimator











**Observation** 

Top Policies

Top Rewards

Module 1: LLM Reward Function Generation Module 2: Teacher Guided **PPO Training** 

Module Function

Module 3: Deploy, Evaluation, and Feedback