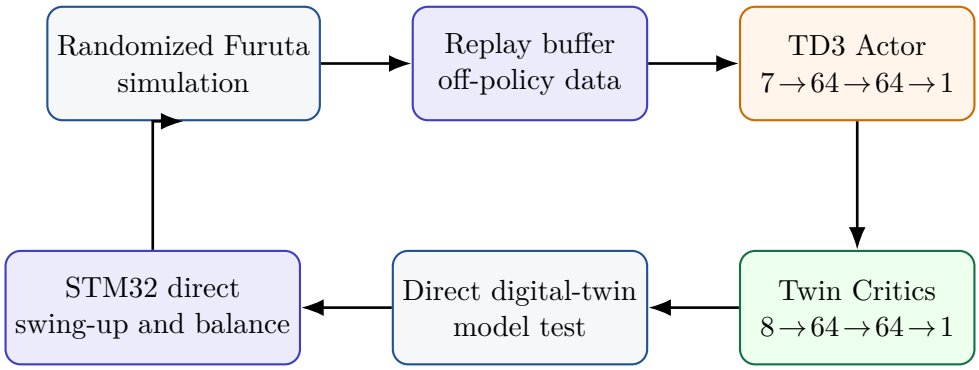
Twin Delayed Deep Deterministic Policy Gradient Control of a Rotary Inverted Pendulum

Class 13 Laboratory Technical and Operating Manual

AI Enabled Control Engineering



Laboratory design objective. Train one compact continuous-action TD3 Actor to perform the complete task: hanging-down swing-up, upright capture and long-duration balance. The same $7 \to 64 \to 64 \to 1$ Actor is tested in the supplied nonlinear digital twin and uploaded directly to the STM32. The trained Actor is already the final deployment-sized $7 \to 64 \to 64 \to 1$ network, so the same Actor is used from training through simulation and hardware.

IMPORTANT

Course model-selection rule. In both deployment panels, click **Choose Model File** and select the intended SB3 file named `best_model.zip`. For the supplied benchmark, select `runs/td3_rip_original_2m_then_dr_20260721_113956/best_nominal_model/best_model.zip`. Do not select the whole run folder and do not select a cached `*.td3_deploy.npz` for the benchmark exercise.

Course staff: Professor Xun Huang; TAs: Zhixiang Ju and Haozhe Wang
Version: July 2026 – standalone Direct64 revision

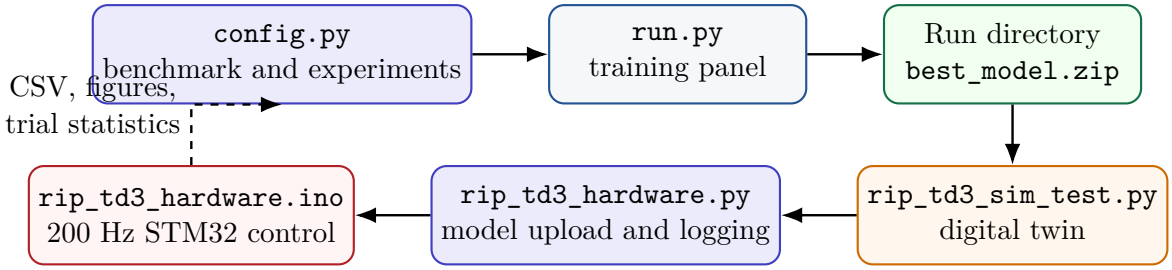
Contents

1 Learning Objectives and Complete Laboratory Architecture

By the end of this laboratory, students should be able to:

1. formulate continuous-action feedback control as a Markov decision process;
2. derive the deterministic policy-gradient and critic Bellman objectives;
3. explain overestimation bias, clipped double-Q learning, target-policy smoothing and delayed Actor updates;
4. explain why TD3 is off-policy and how replay data are reused;
5. interpret the exact seven-dimensional observation, continuous PWM action, reward, termination conditions and staged domain randomization;
6. operate every tab and button in the training panel launched by `python run.py`;
7. distinguish episode reward, moving-mean stability, deterministic evaluation reward, capture rate and stable-success rate;
8. select the correct `best_model.zip` checkpoint instead of a final or stale cached model;
9. conduct repeatable simulation tests through the digital-twin panel;
10. flash the STM32 firmware, calibrate sensors, upload the Actor with CRC verification, and execute controlled hardware trials;
11. improve the benchmark through controlled parameter experiments without changing the deployment contract accidentally.

1.1 End-to-end artifact flow



IMPORTANT

The supplied values are a **benchmark**. Students should improve curve stability, reduce late-training degradation, reduce PWM effort, or improve robustness. Change one controlled factor at a time and preserve the original configuration snapshot for every run.

2 Reinforcement Learning for Continuous Feedback Control

2.1 Agent, environment, observation, action and return

At policy step k, the rotary-pendulum environment provides an observation s_k, the Actor produces a continuous action $a_k \in [-1, 1]$, the simulator maps that action to PWM, and the environment returns reward r_k and next observation s_{k+1}. The discounted return is

$$G_k = \sum_{j=0}^{\infty} \gamma^j r_{k+j}, \qquad 0 < \gamma < 1. \tag{1}$$

The objective is to learn a feedback policy that maximizes expected return, not only the next reward.

2.2 Why continuous action matters

DQN selects one element from a finite action table. TD3 instead outputs a scalar continuously, which is then mapped to PWM. The benchmark uses

$$\mathrm{PWM}_k = 150\, a_k, \qquad a_k \in [-1, 1]. \tag{2}$$

Continuous action avoids a coarse ten-action table, but it requires an Actor to choose actions and one or more Critics to estimate their long-term value.

3 From DDPG to TD3

3.1 Deterministic Actor and action-value Critic

The Actor is a deterministic function

$$a = \mu_\phi(s), \tag{3}$$

with parameters ϕ. A Critic approximates

$$Q_\theta(s, a) = \mathbb{E}\left[G_k \mid s_k = s, a_k = a\right]. \tag{4}$$

The Actor is trained to choose actions that the Critic values highly.

3.2 Replay buffer and off-policy learning

Each transition

$$(s_k, a_k, r_k, s_{k+1}, d_k) \tag{5}$$

is stored in a replay buffer. Minibatches are sampled later, so the policy can learn from data generated by older versions of itself. This makes TD3 sample-efficient but introduces sensitivity to replay distribution, Critic learning rate and stage changes.

3.3 DDPG Bellman target and Critic loss

A basic deterministic Actor-Critic target is

$$y_k = r_k + \gamma(1 - d_k) Q_{\bar{\theta}}\left(s_{k+1}, \mu_{\bar{\phi}}(s_{k+1})\right), \tag{6}$$

where barred parameters denote slowly updated target networks. The Critic loss is

$$L_Q(\theta) = \mathbb{E}_{\mathcal{B}}\left[\left(Q_\theta(s, a) - y\right)^2\right]. \tag{7}$$

3.4 Deterministic policy-gradient update

The Actor objective is

$$J(\phi) = \mathbb{E}_{s\sim\mathcal{B}}\left[Q_\theta\left(s, \mu_\phi(s)\right)\right]. \tag{8}$$

The implementation minimizes

$$L_\mu(\phi) = -\mathbb{E}_{s\sim\mathcal{B}}\left[Q_{\theta_1}\left(s, \mu_\phi(s)\right)\right]. \tag{9}$$

The chain rule propagates the Critic's action gradient through the Actor.

3.5 Problem 1: overestimation bias

A learned Critic contains approximation error. Maximizing a noisy estimate tends to select actions whose errors are optimistic. DDPG can therefore drive the Actor toward incorrectly high Q-values, producing unstable learning or sudden collapse.

3.6 TD3 improvement 1: twin Critics and clipped double Q

TD3 maintains two independent Critics, Q_{θ_1} and Q_{θ_2}, and uses the smaller target value:

$$y = r + \gamma(1 - d) \min_{i\in\{1,2\}} Q_{\bar{\theta}_i}(s', \tilde{a}'). \tag{10}$$

The minimum is deliberately conservative and reduces positive estimation bias.

3.7 TD3 improvement 2: target-policy smoothing

The target action is perturbed by clipped Gaussian noise:

$$\epsilon \sim \text{clip}\left(\mathcal{N}(0, \sigma_t^2), -c, c\right), \tag{11}$$

$$\tilde{a}' = \text{clip}\left(\mu_{\bar{\phi}}(s') + \epsilon, -1, 1\right). \tag{12}$$

This discourages narrow Q-value peaks that would be difficult to reproduce on real hardware. The benchmark uses $\sigma_t = 0.2$ and $c = 0.5$.

3.8 TD3 improvement 3: delayed Actor and target updates

The Critics update every gradient step, but the Actor and target networks update only once every `policy_delay=2` Critic updates. The Critic therefore has more time to improve before its gradient is used to move the Actor.

3.9 Polyak target update

After a delayed Actor update, target parameters are moved slowly:

$$\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}, \qquad \bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}. \tag{13}$$

The benchmark uses $\tau = 0.005$.

3.10 Exploration noise

During training, the behavior action is

$$a_k = \text{clip}\left(\mu_\phi(s_k) + \eta_k, -1, 1\right), \qquad \eta_k \sim \mathcal{N}(0, 0.1^2). \tag{14}$$

This noise is used only for data collection. Deterministic evaluation uses the Actor without exploration noise.

3.11 One complete TD3 update

1. Execute the noisy Actor action and append the transition to replay.
2. Sample a minibatch of 128 transitions after 10,000 warm-up steps.
3. Compute a smoothed target action with the target Actor.
4. Evaluate both target Critics and keep their minimum.
5. Minimize the sum of the two Critic mean-square errors.
6. Clip Critic gradient norm to 1.0.
7. Every second update, maximize $Q_1(s, \mu(s))$ with respect to the Actor.
8. Clip Actor gradient norm to 1.0.
9. Polyak-update Actor and Critic targets.

3.12 TD3 compared with DQN and PPO

Property	DQN	PPO	TD3
Action	Discrete	Continuous stochastic	Continuous deterministic
Data usage	Off-policy replay	On-policy rollout	Off-policy replay
Value models	Q-network and target	Value Critic	Twin Critics and targets
Deployment	Direct small Q-network	Compact deployment Actor	Direct compact Actor
Main stability tool	Target and replay	Clipped probability ratio	Twin Q, smoothing, delayed policy

4 Benchmark Task and Exact Implementation

4.1 Task scope: one policy performs swing-up and balance

The TD3 task starts near the hanging-down configuration. The same Actor must inject energy, pass through upright, capture the pendulum and maintain balance. There is no external energy controller and no switching to LQR.

4.2 Timing and action

Quantity	Benchmark value
Physics and policy period	0.005 s
Control frequency	200 Hz
Action repeat	1
Maximum episode steps	1000
Maximum episode duration	5 s
Action space	one scalar in $[-1, 1]$
Nominal PWM mapping	$a \times 150$

4.3 Seven-dimensional observation and the only history term

The input order is fixed:

$$s_k = [\sin\theta_k,\ \cos\theta_k,\ \dot{\theta}_k,\ \sin\alpha_k,\ \cos\alpha_k,\ \dot{\alpha}_k,\ a_{k-1}]^{\mathsf{T}}. \tag{15}$$

The seventh input is the previous applied normalized control action. It is not a four-frame history. The deployment firmware updates this value after the final PWM safety limit and resets it to zero on model load, CONFIG, GO, STOP and safety termination.

IMPORTANT

Do not change `ENV.observation_type`, `observation_dim`, state/action history lengths, velocity limits, network architecture, activation function, normalization flags or `ENV.pwm_limit` when using the supplied deployment package. Its firmware and model loader expect exactly the benchmark contract.

4.4 Actor and twin-Critic networks

- Actor: $7 \to 64 \to 64 \to 1$, ReLU, ReLU, Tanh.
- Each Critic: concatenated state and action, $8 \to 64 \to 64 \to 1$, ReLU.
- Actor parameters: 4,737.
- Each Critic parameters: 4,801.
- STM32 upload blob: Actor parameters plus three metadata floats, 18,960 bytes total.

4.5 Exact reward

Let $\alpha_w = \mathrm{wrap}_{[-\pi,\pi]}(\alpha)$ and let u be the normalized action. The executable reward is

$$r_k = p_k - (0.001\theta^2 + 8\alpha_w^2 + 0.001\dot{\theta}^2 + 0.005\dot{\alpha}^2 + 0.5u^2 + 0 \cdot (u - u_{k-1})^2). \tag{16}$$

The gate bonus is $p_k = 1$ only when all of the following hold:

$$|\theta| < 60^\circ, \quad |\dot{\theta}| < 5, \quad |\alpha_w| < 12^\circ, \quad |\dot{\alpha}| < 2. \tag{17}$$

Otherwise $p_k = 0$.

4.6 Initial-state distribution and termination

The reset distribution is centered at $\theta = 0$ and $\alpha = 180^\circ$ with standard deviation 25.98° for both angles. This approximates a uniform $\pm 45^\circ$ range. Initial velocities are zero. Episodes terminate on non-finite dynamics or violations of the configured arm/velocity limits; the pendulum angle itself is not used as an early termination threshold.

4.7 Staged domain randomization

The configured full schedule is:

Stage	Name	Level	Steps
1	nominal original alignment	0.00	2,000,000
2	randomization 0.10	0.10	1,000,000
3	randomization 0.30	0.30	1,000,000
4	randomization 0.50	0.50	1,000,000

At each randomized-stage boundary, the package clears replay, resets both optimizers, synchronizes target networks, resets action noise and waits 25,000 new steps before gradient learning resumes. This prevents nominal replay data from dominating the new distribution.

4.8 Benchmark TD3 hyperparameters

Parameter	Value	Parameter	Value
Replay buffer	1,000,000	Learning starts	10,000
Batch size	128	Discount γ	0.99
Actor learning rate	10^{-4}	Critic learning rate	10^{-3}
Polyak τ	0.005	Train frequency	every step
Gradient steps	1	Policy delay	2
Target noise	0.2	Target-noise clip	0.5
Exploration sigma	0.1	Gradient clips	1.0
Observation normalization	off	Reward normalization	off

4.9 Evaluation, capture and stable success

Every 20,000 environment steps, five deterministic episodes are evaluated. Capture is recorded if $|\alpha| \leq 12^\circ$ at least once. Stable success requires $|\alpha| \leq 12^\circ$ and $|\dot{\alpha}| \leq 2$ continuously for at least 2 seconds. Best checkpoints are ranked by:

1. stable-success rate;
2. control score;
3. mean reward if the previous two tie.

The control score is

$$S = \bar{R} + 0.02\bar{L} - 2\overline{|\theta|} - 12\overline{|\alpha|} - 2\overline{|u|} - 20\, r_{\text{term}}. \tag{18}$$

4.10 Packaged reference-run result

The included reference archive contains a nominal run through approximately 1.36 million steps; it did not reach the configured domain-randomization stages. Its selected nominal best checkpoint occurred at 920,000 steps.

Metric at selected checkpoint	Value
Mean reward / 1000-step episode	−6222.514
Mean reward per step	−6.22251
Capture rate	100%
Stable-success rate	100%
Mean $\|\theta\|$	0.38998 rad
Mean $\|\alpha\|$	0.39322 rad
Mean |PWM|	32.783
Control score	−6208.449

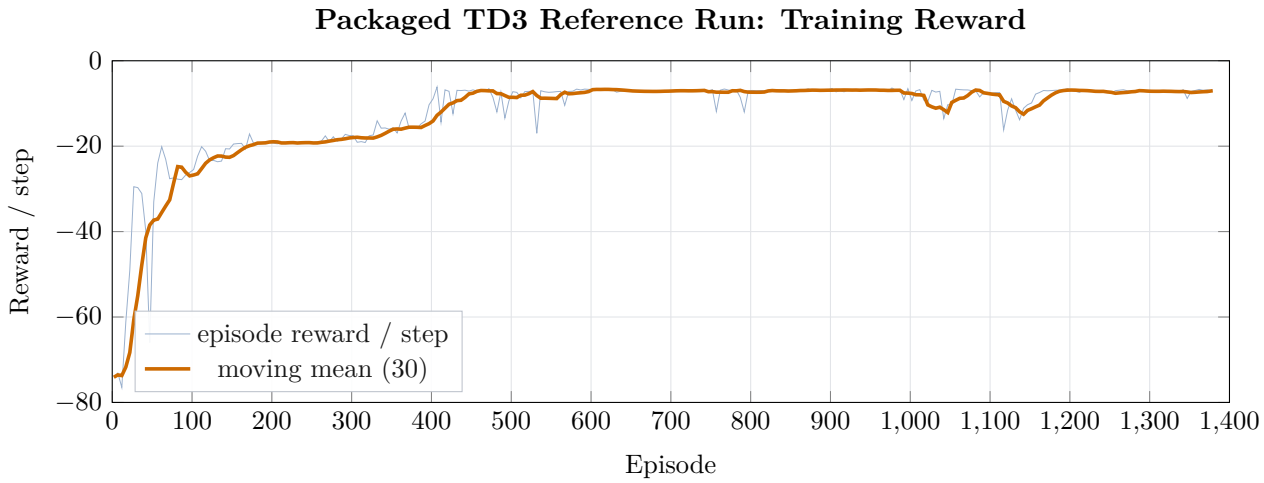


Figure 1: Packaged reference training curve. The moving mean improves and then shows late nominal-stage degradation. This provides an opportunity for students to improve stability. The plot is generated directly inside this LaTeX file; no external image is required.

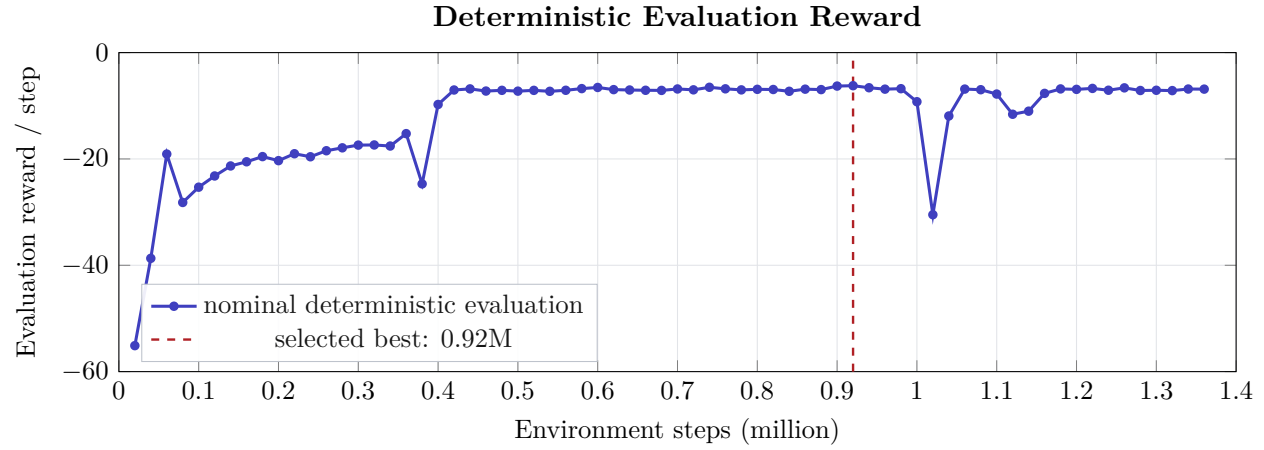


Figure 2: Deterministic evaluation reward. The best checkpoint is not the final checkpoint. The curve is embedded as native PGFPlots data.

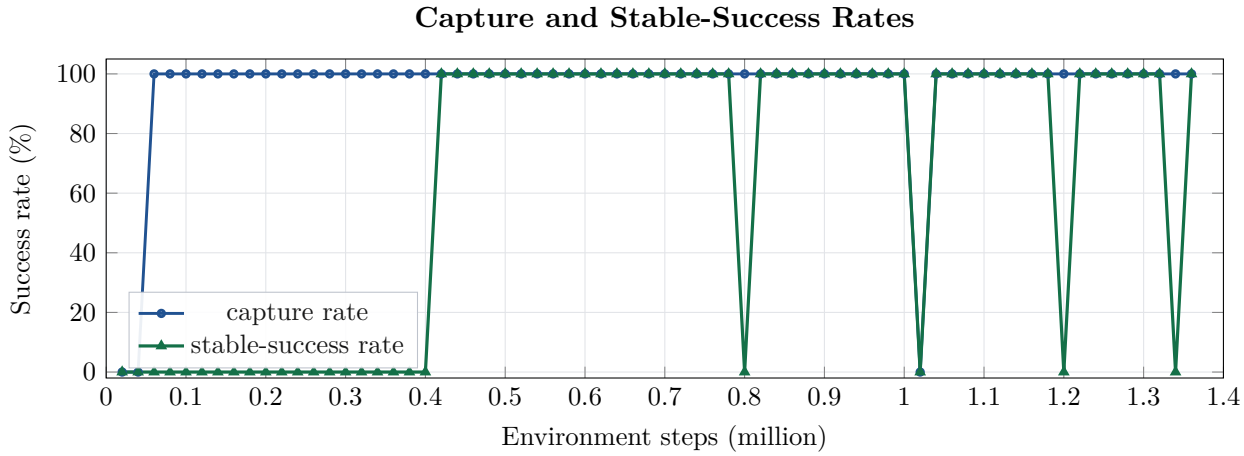


Figure 3: Capture and stable-success rates in the packaged nominal run. The figure is self-contained and does not depend on a separate image folder.

5 Training Package: File-by-File Reference

5.1 Package structure

```
td3/
  run.py
  panel.py
  config.py
  config_editor.py
  train_worker.py
  runtime.py
  rip_env/
  stable_baselines3/
  third_party/
  runs/
```

5.2 What every code file does

Direct deployment rule. The trained TD3 Actor already has the final $7 \to 64 \to 64 \to 1$ architecture used on the STM32. Do not create a second network. Simulation and hardware panels load the selected `best_model.zip` and deploy that Actor directly.

File or folder	Student-facing purpose
`run.py`	The only normal entry. Running without arguments opens the training panel. Internal worker arguments launch formal training or the short installation smoke test.
`panel.py`	Tkinter GUI. Safely edits configuration, starts subprocesses, reads structured events and plots training/evaluation curves.
`config.py`	Single source of truth for environment, TD3, reward, evaluation, domain randomization and GUI settings.
`config_editor.py`	Writes selected values back to `config.py` and creates timestamped backup copies before modification.
`train_worker.py`	Creates environments, builds the custom TD3 model, executes stages, evaluates deterministic policies, saves checkpoints and selects best models.

File or folder	Student-facing purpose
`runtime.py`	Seven-dimensional observation validator, action-repeat wrapper, separate Actor/Critic learning rates, custom TD3 update, metrics and environment constructors.
`rip_env/`	Furuta nonlinear dynamics, reward/done plumbing, logging and domain-randomization implementation.
`stable_baselines3/`	Bundled local SB3 source used by the package.
`third_party/`	Bundled Gymnasium-related dependencies.
`runs/`	Training outputs, checkpoints, metrics, logs and selected models.

5.3 Python environment

From the extracted `td3` directory, install or verify Python 3, NumPy, PyTorch, Matplotlib and Tkinter. The package already contains local SB3 and Gymnasium sources. A typical launch is:

```
cd /path/to/td3
python3 run.py
```

On macOS, the Python installation must include Tk support. Do not launch the script from inside the ZIP archive.

6 Operating the TD3 Training Panel

6.1 Opening the panel

Run:

```
python3 run.py
```

The header confirms the deployment contract: seven-dimensional input, a $7 \to 64 \to 64 \to 1$ Actor, twin $8 \to 64 \to 64 \to 1$ Critics, 2M nominal steps and three randomized stages.

6.2 Top toolbar: every button

Save to config.py Parse all visible Common Parameters and write them to `config.py`. A backup file is created automatically.

Reload Discard unsaved edits and reload the current `config.py`.

Start Full Training Auto-save the current fields, lock configuration, create a new run directory and start the configured 5M-stage workflow.

Run Full Smoke Test Run 512 total steps, a short stage transition, TD3 training/evaluation cycle and a short stage-transition check. Use this before a long run to detect installation or environment errors.

Emergency Stop Terminate the active worker. The current checkpoint may be incomplete; inspect the run directory afterward.

Open Runs Directory Open the latest run, or the configured `RUN.root_log_dir` when no run has started.

6.3 Tab 1: Common Parameters

This scrollable tab exposes grouped settings. Click in the value field, type a valid Python literal, then click **Save to config.py**. Examples:

- integer: `256`;

- float: `0.0005`;
- Boolean: `True` or `False`;
- tuple: `(64, 64)`;
- tuple of stages: `(0.0, 0.1, 0.3, 0.5)`.

The groups cover environment/observation, TD3, fixed-stage randomization, physical parameter ranges, reward, evaluation/best/early-stop and GUI settings.

6.4 Tab 2: All Config Parameters

This tab dynamically lists every dictionary field in `config.py`. To edit:

1. select one row in the table;
2. edit its value in the lower entry;
3. click **Apply Selected Value**;
4. confirm the backup filename in the message box.

Use this tab for fields not included in Common Parameters. Avoid changing multiple unrelated groups in one experimental run.

6.5 Tab 3: Training, Testing, and Curves

The top summary displays:

- current run directory;
- latest episode reward per step, moving mean over 30 episodes and full-episode rate;
- deterministic evaluation reward, capture and stable-success rate;
- active training stage, randomization level and stage-transition actions.

Three live plots show:

1. episode reward per step and 30-episode moving mean;
2. deterministic evaluation reward per step;
3. capture and stable-success rates.

6.5.1 Where model testing is performed

The training panel is used only for training, deterministic evaluation and checkpoint selection. Required closed-loop model testing is performed with the separate `rip_td3_sim_test.py` deployment panel described later. This keeps the tested network identical to the selected SB3 `best_model.zip`.

6.6 Tab 4: Full Log

This tab contains the exact worker command, SB3 logs, structured panel events, stage transitions, evaluation results and errors. When reporting a failure, copy the relevant lines from here rather than describing only the GUI symptom.

6.7 Recommended training-panel workflow

1. Extract the package to a normal writable directory.
2. Run `python3 run.py`.
3. Click **Run Full Smoke Test** once.
4. Open the Full Log and confirm the final `SMOKE_OK` message.
5. Click **Reload** and record the baseline values.
6. For the first formal run, change nothing.

7. Click **Start Full Training**.
8. Observe moving-mean reward and deterministic evaluation, not isolated episode spikes.
9. After completion or an intentional stop, open the run directory and preserve `config_snapshot.json`, logs and best models.
10. For each improvement run, change only one parameter group and use a new seed.

6.8 Worker command quick reference

Normal users need only `python3 run.py`. The internal commands are:

```
python3 run.py --worker train
python3 run.py --worker smoke
```

7 Training Outputs and Correct Model Selection

7.1 Expected full run directory

```
runs/td3_rip_original_2m_then_dr_TIMESTAMP/
  config_snapshot.json
  eval_history.json
  env_logs/env_0/episode_log.csv
  checkpoints/td3_XXXXXXXXX.zip
  stage_models/
  recovery_model/nominal_2m_last.zip
  best_nominal_model/best_model.zip
  best_nominal_model/best_metrics.json
  best_randomized_model/best_model.zip
  best_randomized_model/best_metrics.json
  best_model/best_model.zip
  best_model/selection.json
  selected_best_model.zip
  final_model.zip
  training_summary.json
```

A partial or interrupted nominal run may contain only checkpoints, `best_nominal_model` and logs.

7.2 Meaning of important models

best nominal	Best deterministic policy measured at randomization 0.0.
best randomized	Best policy during randomized evaluation, if the run reached that phase.
best model	Copy selected by the configured randomized-success threshold; inspect `selection.json`.
selected best model	Root-level convenience copy of the selected best.
final model	Actor at the final training step. It may be worse than an earlier best checkpoint and should not be selected by default.
checkpoint	Periodic recovery point, useful for diagnosing collapse or comparing different times.

7.3 Mandatory deployment selection procedure

DANGER

For this laboratory, do not deploy by selecting the run directory. Do not deploy a `*.td3_deploy.npz` cache. In the simulation or hardware panel click **Choose Model File** and select an SB3 ZIP whose filename is exactly `best_model.zip`.

For the supplied reference archive, select:

```
td3/runs/td3_rip_original_2m_then_dr_20260721_113956/
  best_nominal_model/best_model.zip
```

For a completed student run, first inspect `best_model/selection.json`; then normally select `best_model/best`
Select the nominal or randomized best explicitly when conducting a controlled comparison.

8 How to Improve the Benchmark Fairly

8.1 What "better" means

A student result can improve the benchmark in one or more measurable ways:

- higher deterministic reward per step than -6.2225;
- smaller 30-episode moving standard deviation than approximately 0.13 near the best region of the packaged reference run;
- less late-training degradation after the best checkpoint;
- 100% capture and stable success across more evaluation episodes/seeds;
- lower mean $|\alpha|$ than 0.393 rad;
- lower mean |PWM| than 32.8 while preserving capture;
- better performance under level-0.5 randomized evaluation;
- lower variance over at least three independent training seeds.

8.2 Settings that must remain fixed for deployment compatibility

Keep the following unchanged unless you also redesign and verify the deployment loader and firmware:

- `ENV.observation_type='trig_hist1_act1'` and dimension 7;
- state history 1 and action history 1;
- Actor architecture $(64, 64)$ and ReLU activation;
- one-dimensional continuous action;
- `normalize_obs=False` and `normalize_reward=False`;
- PWM scale 150 and velocity clips 45/40 rad/s.

8.3 Priority 1: reduce Critic-driven oscillation

The reference Critic learning rate, 10^{-3}, is aggressive. Test one of:

$$\texttt{critic\_learning\_rate} \in \{5 \times 10^{-4},\ 3 \times 10^{-4}\}. \tag{19}$$

Keep the Actor rate at 10^{-4} initially. A lower Critic rate often produces a smoother evaluation curve and reduces abrupt policy degradation, at the cost of slower early improvement.

Next, test `batch_size=256`. Larger batches reduce gradient noise, use more memory and may require longer wall-clock time. Do not change learning rate and batch size in the same first experiment.

8.4 Priority 2: exploration and target smoothing

Test exploration sigma values 0.08, 0.10 and 0.12. Lower noise can stabilize late training but may prevent discovery of a swing-up trajectory. Higher noise may improve exploration but increase Q-target variance.

If the Critic remains noisy, test target-policy noise 0.10 with clip 0.30. Keep exploration noise unchanged in that experiment so the source of improvement is identifiable.

8.5 Priority 3: delayed updates and target speed

Test `policy_delay=3` if Actor updates appear to chase unstable Q-values. Test `tau=0.003` for slower target motion. These settings can improve smoothness but slow adaptation after a domain-randomization stage transition.

8.6 Priority 4: smooth-control reward terms

The benchmark has no action-change penalty. After a reliable swing-up baseline exists, test:

$$\texttt{REWARD.a\_du} \in \{0.01, 0.02, 0.05\}. \tag{20}$$

Start at 0.01 or 0.02. A large value can suppress the rapid action reversals needed for swing-up. Alternatively, test `a_u=0.6` before 0.8 to reduce control effort. Report both success and PWM metrics; a smooth but unsuccessful policy is not an improvement.

8.7 Priority 5: robustness track

Only after nominal performance is reliable, conduct a separate robustness experiment. A gentler curriculum such as levels $(0, 0.05, 0.15, 0.30)$ can reduce stage shock, but it is an easier randomization task and must not be claimed as a direct benchmark improvement unless evaluated at the original level 0.5.

A valid robustness comparison keeps the final evaluation distribution fixed at level 0.5 even when the training curriculum changes.

8.8 Evaluation settings improve measurement, not policy

Increasing `EVAL.n_eval_episodes` from 5 to 10 or 20 makes success estimates more reliable. Reducing `eval_freq` to 10,000 provides more detailed curves. These changes do not directly improve the policy; they improve model selection and diagnosis.

8.9 Recommended experimental matrix

Run	Controlled change	Primary hypothesis	Compare
A	Baseline, seed 42	Reproduce reference	all metrics
B	Critic LR 5×10^{-4}	less Q oscillation	eval variance
C	Batch 256	smoother gradients	moving std
D	Exploration sigma 0.08	less late noise	reward and capture
E	Target noise 0.10, clip 0.30	smoother targets	late degradation
F	$a_{du} = 0.02$	smoother PWM	success and PWM
G	Best setting, seeds 7/42/123	repeatability	mean and std
H	Robustness curriculum	sim-to-real margin	level-0.5 eval

STUDENT TASK

For every formal run, save the run directory, configuration snapshot, training curve, evaluation curve, best metrics, chosen model path and random seed. A single visually good episode is not sufficient evidence.

9 Deployment Package: File-by-File Reference

```
deploy_td3/
  rip_td3_sim_test.py
  rip_td3_hardware.py
  rip_td3_hardware/rip_td3_hardware.ino
  convert_td3_model.py
```

File	Purpose
`rip_td3_sim_test.py`	Nonlinear digital twin, direct SB3 ZIP/NPZ/header loader, 7 → 64 → 64 → 1 NumPy inference, velocity estimator, 3-D animation, CSV/PNG results and headless mode.
`rip_td3_hardware.py`	PyQt hardware panel, manual serial refresh, sensor calibration, model adaptation, slow CRC-verified upload, CONFIG, GO/STOP, live telemetry and result logging.
`rip_td3_hardware.ino`	STM32 v22 firmware, 200 Hz sensor/control loop, Actor inference, motor drive, observer blending, safety, model-upload protocol and 20 Hz run telemetry.
`convert_td3_model.py`	Optional conversion utility that writes a canonical NPZ. It is not used in the required benchmark workflow; select `best_model.zip` directly instead.

9.1 Deployment dependencies

Install:

```
python3 -m pip install numpy matplotlib PyQt5 pyserial torch
```

PyTorch is needed when loading the required SB3 `best_model.zip`. The simulation and hardware panels adapt the Actor weights directly; stable-baselines3 is not required by the deployment folder.

10 Using the Digital-Twin Simulation Panel

10.1 Launch

From the deployment directory:

```
python3 rip_td3_sim_test.py
```

10.2 Model selection: exact required operation

1. Click **Choose Model File**.
2. Navigate into the training run directory.
3. Open `best_nominal_model` for the supplied benchmark.
4. Select `best_model.zip`.
5. Do not click **Choose Training Run Directory** for the benchmark.
6. Do not select a `*.td3_deploy.npz` cache.

After SAVE, the model line should report `compact7 7->64->64->1 ReLU/ReLU/Tanh`, PWM scale 150 and a 16-character model ID.

10.3 Every simulation setting

Initial condition Random hanging-down; exact hanging-down; near upright +8°; or near upright −8°.

Duration / s Simulated experiment duration. Use 30 s for formal tests.

Playback speed Wall-clock animation multiplier only; it does not change the 5 ms simulation timestep.

Domain randomization Level from 0 to 1. Use 0 for nominal validation and 0.5 for the benchmark robustness test.

Seed Controls initial condition and randomized physical parameters.

Safety PWM max Final command clamp. Start at 150.

Observer enter Angle below which the near-upright Luenberger estimate is activated; default 20°.

Observer exit Hysteresis exit angle; default 35°.

Observer blend First-order blending coefficient; default 0.18.

Large-angle velocity LPF Wrapped-difference velocity smoothing; default 0.25.

10.4 Every button and required sequence

1. Select `best_model.zip` with **Choose Model File**.
2. Set initial condition, duration, randomization, seed and safety PWM.
3. Click **SAVE / Load TD3 Model & Apply Settings**. This loads the weights and resets the simulation. A star on SAVE means a setting changed and must be saved again.
4. Click **GO**. The same TD3 Actor immediately performs swing-up and balance at 200 Hz.
5. Click **STOP** to end early and save available data.
6. Click **RESET** to restore the initial condition without starting.
7. Leave **Generate CSV log** checked and use **Browse** to select the output directory.
8. Click **Show Last Result Curves** after a completed or stopped run.

10.5 Live modes

The display reports `TD3_DIFF`, `TD3_BLEND` or `TD3_OBSERVER`. These names describe the velocity estimate supplied to the same Actor; they are not different controllers.

10.6 Headless simulation

```
python3 rip_td3_sim_test.py --headless \
  --model /path/to/best_model.zip \
  --duration 30 --seed 2026 \
  --randomization 0.5 --pwm-limit 150
```

10.7 Formal simulation acceptance procedure

1. Run at least ten trials using documented seeds.
2. Include nominal level 0 and randomized level 0.5 trials.
3. Record capture, stable duration, mean $|\alpha|$, mean |PWM| and termination.
4. Keep failures; do not repeat a seed until it succeeds.
5. Select a model for hardware only after repeatable simulated swing-up and balance are demonstrated.

11 Preparing the STM32 Firmware

11.1 Firmware pin contract

Function	Pin	Firmware symbol
Pendulum potentiometer	A0	`PIN_POT`
Encoder A/B	2 / 3	`PIN_ENC_A/B`
Motor direction	4 / 5	`PIN_IN1/IN2`
Driver standby	7	`PIN_STBY`
Motor PWM	10	`PIN_PWM`

Confirm that the physical wiring matches before applying power.

11.2 Flash procedure

1. Open `rip_td3_hardware/rip_td3_hardware.ino` in Arduino IDE.
2. Select the correct STM32 board, upload method and ST-Link interface.
3. Compile and upload.
4. Close Arduino Serial Monitor after flashing. It must not share the serial port with the Python panel.
5. The expected firmware identification is `READY,RIP_TD3_64_DIRECT_HW_GO_FIX,22`.

11.3 Firmware responsibilities

The firmware reads the potentiometer and encoder, estimates velocities, builds the seven-dimensional observation, evaluates the uploaded Actor, applies the safety-limited PWM, stores the previous normalized action, monitors state limits and sends telemetry. Control runs at 200 Hz, run telemetry at 20 Hz and idle monitoring at 4 Hz.

11.4 Model-upload protocol

The 18,960-byte Actor blob is transmitted as slow 32-byte ASCII-hex blocks. Each block includes offset, length and CRC32 and is acknowledged before the next block. The current Python panel uses manual port refresh/reconnect and retries corrupted blocks on the same connection. Do not click Refresh during upload.

12 Using the Physical-System Panel

12.1 Launch

```
python3 rip_td3_hardware.py
```

DANGER

Clear the mechanism's full motion envelope. Use an accessible emergency power switch. The default checkbox runs continuously until STOP. For the first test, uncheck continuous mode and use a 5–10 second timed run.

12.2 Serial Connection group

1. Close Arduino Serial Monitor and every other serial application.
2. Click **Refresh now** once.
3. Select the STM32 serial port.
4. Click **Connect**.
5. Wait until the status confirms protocol v22 and firmware READY.

Refresh and reconnect are manual. Do not repeatedly refresh after a successful connection because a board reset can clear the uploaded model and configuration.

12.3 TD3 Model Selection and STM32 Deployment group

Click **Choose Model File**; navigate to the intended run and select `best_model.zip`. For the supplied benchmark use the exact nominal-best path stated earlier. Do not use **Choose Run Folder** in the required workflow. The panel should display SB3 ZIP auto-adaptation, the compact7 architecture, float32 upload, model ID and 18,960-byte blob.

12.4 Direct TD3 Control and Velocity Estimation group

Run continuously until STOP
: Checked by default. Uncheck for a timed first trial.

Timed duration
: Used only when continuous mode is unchecked.

Safety PWM max
: Final command limit. Begin with 100–120 for a cautious wiring-direction check, then use 150 for benchmark comparison.

Observer enter/exit
: Defaults 20°/35° and provides hysteresis.

Observer blend
: Default 0.18.

Large-angle velocity LPF
: Default 0.25.

12.5 Sensor Calibration group

1. Hold the pendulum exactly upright and click **Hold upright** → **Calibrate** $\alpha = 0$.
2. Let it hang naturally downward and click **Hang down** → **Calibrate** $\alpha = \pi$.
3. Verify that upright and hanging raw values are different and stable.
4. Verify `theta rad / encoder count`. The supplied default is approximately 0.00583730846 rad/-count.
5. Set Motor sign to +1 initially. If the mechanism moves opposite to the expected simulation convention, STOP, power down and change the sign.

12.6 SAVE: upload model and parameters

After connection, calibration and model selection:

1. click **SAVE / Upload TD3 Model & Parameters**;
2. do not click Refresh or disconnect during upload;

3. wait for all 593 blocks, complete-model CRC and CONFIG acknowledgement;
4. the GO button becomes enabled only after model and configuration are both acknowledged;
5. if a chunk fails, click SAVE again on the same connection. Refresh only when the port truly disconnected.

12.7 GO, first-command diagnosis and STOP

Click GO only with a clear mechanism. Before sending GO, the host computes a preflight action from the current measured state. Firmware v22 also computes and applies the first Actor command synchronously and returns it in the ARMED reply. The status should report, for example, initial PWM, normalized action and raw network output. This confirms the software path from model to motor command.

Click STOP before touching the mechanism. STOP is transmitted several times for reliability and the firmware brakes/disables the driver.

12.8 Live display and logging

The right panel shows the 3-D mechanism, time, angles, PWM, velocity-estimator mode, blend, action and raw Actor output. With CSV enabled, the panel saves:

- time, θ, $\dot{\theta}$, α, $\dot{\alpha}$;
- actual PWM;
- mode and observer blend;
- normalized action and pre-Tanh raw output.

After STOP or timed completion, click **Show Last Result Curves**.

13 Safety Limits and Error Messages

Message or symptom	Meaning and action
GO remains disabled	Model or CONFIG has not been acknowledged, a field changed after SAVE, or upload is active.
`ERR,RUN_MODEL_REQUIRED`	Model was cleared or never uploaded. SAVE again without refreshing unnecessarily.
`ERR,RUN_CONFIG_REQUIRED`	CONFIG was not acknowledged. Recheck calibration values and SAVE.
`ERR,RUN_MODEL_INFERENCE`	Non-finite network output or invalid model data. STOP and reselect the SB3 `best_model.zip`.
`ERR,MODEL_HEX_CHUNK_FORMAT`	Serial line arrived truncated. Keep the same connection and click SAVE again; the current panel uses smaller paced blocks.
`ERR,SAFETY_STATE`	Angle/velocity became non-finite or exceeded safety limits. Inspect calibration, encoder sign and physical motion.
ARMED shows nonzero PWM but motor does not move	Software inference succeeded. Check STBY, IN1/IN2, PWM pin, driver power, common ground and motor wiring.
Motor moves in the wrong direction	STOP immediately, power down and change Motor sign.
Panel disconnected after Refresh	Refresh may reset/re-enumerate the board. Reconnect manually and upload model/CONFIG again.
Model controls in simulation but not hardware	Check sensor zero, encoder scale, motor sign, friction, dead zone, velocity estimates and domain-randomization coverage.

14 Complete Recommended Workflow

1. Run the training-package smoke test.

2. Train the unchanged baseline and preserve its run directory.
3. Conduct controlled improvement experiments, one factor at a time.
4. Select the best checkpoint using deterministic metrics, not the final training step.
5. In the digital-twin panel click Choose Model File and select `best_model.zip`.
6. Complete ten simulation trials and document every seed.
7. Flash the v22 firmware and close Serial Monitor.
8. Launch the hardware panel, Refresh once, Connect and calibrate.
9. Choose the same `best_model.zip`; SAVE and wait for CRC/CONFIG.
10. Perform a short low-PWM timed direction test.
11. Increase to the benchmark limit only after the sign and safety response are correct.
12. Complete ten documented hardware trials, including failures.

15 Required Experimental Work

15.1 Training-stage requirement

Students must submit the baseline and at least three controlled parameter runs. At least one run must address curve stability and at least one must address control smoothness or robustness. Use at least three seeds for the best setting.

15.2 Ten simulation trials

Record model path, randomization level, seed, initial condition, capture, stable duration, mean $|\alpha|$, mean |PWM| and termination reason.

15.3 Ten physical-system trials

Record calibration values, motor sign, safety PWM, model ID, initial first command, capture, stable duration, mean $|\alpha|$, mean |PWM| and failure mode. Never delete a failed trial.

16 Summary

TD3 combines an off-policy replay buffer, deterministic Actor, twin Critics, clipped double-Q targets, target-policy smoothing and delayed policy updates. The supplied benchmark uses one compact $7 \to 64 \to 64 \to 1$ Actor for both swing-up and balance. Students should train and improve the benchmark, select an SB3 `best_model.zip` explicitly, verify it through repeated nonlinear simulation trials, and deploy exactly the same Actor to the STM32 through the manual-refresh v22 hardware panel. Reproducible evidence, controlled parameter changes and safety discipline are more important than one successful video.