10-Day Light→Medium Study Plan for LLM-Augmented Human-in-the-Loop MARL

Using ENSIA RL Lectures + Sutton & Barto + Hands-on Tasks

FYP focus: LLM-augmented MARL with human-in-the-loop instruction and explanations

---

Contents

---

1 How to Use This Plan

Goal. In 10 days of light to medium intensity, refresh core RL (tabular → deep), spin up a minimal **MARL** task, and build a usable **LLM bridge** for (i) translating human instructions into structured goals and (ii) generating agent explanations.

Daily cadence (3–5h). Each day includes: *Objectives*, *Read/Watch*, *ENSIA slides / S&B chapters*, *Hands-on*, and *Deliverable*. Keep deliverables tiny and reproducible.

Assumed local material. ENSIA PDFs in your repo:

- `1-Intro.pdf`, `2-Bandits.pdf`, `4-DP.pdf`, `5-MC.pdf`, `6-TD.pdf`, `7-Dyna architecture.pdf`, `8-Prediction with Approximation.pdf`, `9-Control with Approximation.pdf`, `11-Advanced RL1.pdf`, `11-Advanced RL2.pdf`
- `SuttonBartoIPRLBook2ndEd.pdf` (Sutton & Barto, RL: An Introduction, draft 2e)

Software/Hardware Setup (Fedora + RTX 3050Ti)

Use **Conda**, **PyTorch (CUDA)**, **Gymnasium**, **PettingZoo**, **Stable-Baselines3**, and an **LLM runtime** (API or local).

Listing 1: One-time environment setup

```
# (1) Miniconda (if not installed)
# Visit the official Miniconda page and install for Linux x86_64, then:
conda create -n fyp-rl python=3.11 -y
conda activate fyp-rl

# (2) PyTorch (CUDA-enabled; adjust to your local CUDA compatibility)
pip install torch torchvision torchaudio --index-url https://download.
   pytorch.org/whl/cu118

# (3) Core RL stack
pip install gymnasium==0.29.1 gymnasium[classic-control]==0.29.1 \
            pettingzoo==1.24.3 supersuit==3.9.3 \
            stable-baselines3[extra]==2.3.2

# (4) Plotting, utils
pip install numpy scipy matplotlib seaborn pandas tqdm

# (5) LLM bridge (choose one: API or local)
# API-based (example): openai (or other provider sdks)
pip install pydantic==2.* langchain==0.2.* langchain-community

# Local (optional): llama-cpp-python (CPU/GPU build) or use Ollama CLI
pip install llama-cpp-python
# Or install Ollama from official docs and run models locally
```

Repo Skeleton (create once)

Listing 2: Suggested repo layout

```
fyp-rl/
  baselines/
    bandits/           # Day 2
    dp/                # Day 2
    mc/                # Day 3
    td/                # Day 4
    dyna/              # Day 5
    deep/              # Days 6-7 (PPO/SAC)
  marl/
    mpe_simple_spread/ # Day 8
  llm_bridge/
    goal_schema/       # Day 9
    explain/           # Day 10
  experiments/
    day01_api_check.ipynb
    ...
```

```
data/
README.md
requirements.txt
```

2 Day 1 — Reset & Tooling

Objectives. Refresh RL framing; ensure Gymnasium & PettingZoo loops run locally.
Read/Watch.

- **S&B** Ch. 1: The RL Problem.
- **ENSIA** `1-Intro.pdf`: presentation, course scope, basic definitions.

Hands-on.

1. Minimal Gymnasium loop: `FrozenLake-v1` or `CartPole-v1`.
2. Minimal PettingZoo loop: MPE `simple_v3` (reset, step random, render or log).

Deliverable. Notebook: `experiments/day01_api_check.ipynb` showing both loops running.

3 Day 2 — Bandits → MDPs → DP

Objectives. Move from stateless bandits to MDPs, then planning (policy/value iteration).
Read/Watch.

- **S&B** Ch. 2 (Bandits), Ch. 3 (Finite MDPs), Ch. 4 (Dynamic Programming).
- **ENSIA** `2-Bandits.pdf`, `4-DP.pdf`.

Hands-on.

1. Implement ϵ-greedy k-armed bandit (stationary, non-stationary).
2. Implement Value Iteration on a tiny gridworld (e.g., 4x4) and plot V convergence.

Deliverables.

- `baselines/bandits/bandit_egreedy.py`
- `baselines/dp/value_iteration.py` + plot of $\|V_{t+1} - V_t\|$ vs iterations.

4 Day 3 — Monte-Carlo (MC) Prediction & Control

Objectives. First model-free methods; connect returns to V/Q.
Read/Watch.

- **S&B** Ch. 5 (Monte Carlo Methods).
- **ENSIA** `5-MC.pdf`.

Hands-on.

1. First-visit MC prediction on a small episodic gridworld.
2. MC control with ϵ-greedy improvement; compare to DP policy.

Deliverables. `baselines/mc/mc_prediction.py`, `mc_control.py` + brief markdown comparing MC vs DP.

5 Day 4 — TD Learning: SARSA, Q-learning, Expected SARSA

Objectives. Core online TD control that you'll reuse as baselines.
Read/Watch.

- **S&B** Ch. 6 (Temporal-Difference Learning).
- **ENSIA** `6-TD.pdf`.

Hands-on.

1. Implement SARSA & Q-learning on `CliffWalking-v1` under identical ϵ schedule.
2. (Optional) Expected SARSA on the same environment.

Deliverables. `baselines/td/sarsa.py`, `q_learning.py`, with a single chart of episode return (SARSA vs QL).

6 Day 5 — Planning + Learning: Dyna-Q

Objectives. Bridge model-free and model-based; improve sample efficiency.
Read/Watch.

- **S&B** Ch. 8 (Planning & Learning with Tabular Methods).
- **ENSIA** `7-Dyna architecture.pdf`.

Hands-on.

1. Add a simple *model* (dictionary of $(s, a) \to (r, s')$) to Q-learning.
2. Run planning updates per real step with $k \in \{0, 5, 20\}$ and compare steps-to-threshold-return.

Deliverable. `baselines/dyna/dyna_q.py` + plot: env steps to reach a fixed average return vs k.

7 Day 6 — Function Approximation: Tabular → Linear/NN

Objectives. Understand approximation pitfalls (divergence), then use stable deep baselines.
Read/Watch.

- **S&B** Ch. 9 (On-policy prediction/control with approximation overview).
- **ENSIA** `8-Prediction with Approximation.pdf`, `9-Control with Approximation.pdf`.

Hands-on.

1. Train PPO on `CartPole-v1` using Stable-Baselines3 (short run, default hyperparams).
2. Save learning curves and a short rollout video/gif.

Deliverables. `baselines/deep/ppo_cartpole.py`, curve image(s), and notes on batch size, γ, clip ratio.

8 Day 7 — Advanced Deep RL (Policy Gradients, Actor-Critic, Off-Policy)

Objectives. Compare algorithms and *choose one* baseline family for MARL (PPO is a good default).

Read/Watch.

- **ENSIA** `11-Advanced RL1.pdf`, `11-Advanced RL2.pdf`: REINFORCE→TRPO→PPO; DPG→DDPG→TI A2C/ACKTR; SAC.
- **S&B** (Policy Gradient foundations; actor-critic intuition).

Hands-on.

1. Train PPO and SAC on `LunarLander-v2` or `BipedalWalker-v3` (short runs).
2. Compare stability, sensitivity to seeds; pick your "production" algo.

Deliverables. Two learning curves + a 5-line rationale for your choice (e.g., PPO for stability and simplicity).

9 Day 8 — Multi-Agent RL (PettingZoo MPE; Independent Learners)

Objectives. Stand up a **toy MARL** task you can later couple to the LLM bridge.

Read/Watch.

- PettingZoo MPE docs (conceptual); review CTDE vs. independent learners (from slides).

Hands-on.

1. Wrap PettingZoo `simple_spread_v3` for independent PPO agents (one policy per agent).
2. Verify reward shaping and reproducibility (fix seeds; log episode stats).

Deliverables. `marl/mpe_simple_spread/train_independent_ppo.py` + short eval video/gif.

10 Day 9 — LLM Bridge I: Instruction → Structured Goal Spec

Objectives. Convert human text into JSON goals your MARL code can consume (*without* training any model).

Design. Define a strict schema and parse LLM output into it (pydantic/JSON). Keep it deterministic and testable.

Schema (example).

Listing 3: Pydantic schema for task/goals

```python
from pydantic import BaseModel, Field
from typing import List, Optional

class Constraint(BaseModel):
    name: str
    value: float
    note: Optional[str] = None

class GoalSpec(BaseModel):
    task_name: str = "mpe_simple_spread"
    target_zone: Optional[str] = None     # e.g., "left", "north-west"
    formation: Optional[str] = None       # e.g., "line", "circle"
    max_steps: int = Field(200, ge=1, le=10000)
    priorities: List[str] = []            # e.g., ["avoid_collisions",
        "minimize_distance"]
    constraints: List[Constraint] = []
```

Hands-on.

1. Write a prompt template that instructs the LLM to *only* emit JSON matching the schema.
2. Implement a small parser with validation (raise on schema mismatch).
3. Unit tests: 3 natural-language inputs → validated JSON specs.

Deliverables. `llm_bridge/goal_schema/parse_goals.py` + `tests/test_goal_schema.py`.

11 Day 10 — LLM Bridge II: Explanations & Human Feedback Hook

Objectives. Produce post-hoc rationales and support lightweight human corrections, then re-run the episode.

Explain() design. After each episode, summarize in $\leq$5 sentences: (i) goal restatement, (ii) key behaviors, (iii) rule adherence/violations, (iv) improvement hint.

Feedback hook. If user says "avoid collisions and stay left; budget=200", update the JSON *spec* (not network weights), re-run, then regenerate explanation.

Hands-on.

1. Add `explain(policy_trace, goal_spec)` function using a compact prompt and a small model.
2. Build a tiny loop: `text instruction` → `JSON` → `run episode` → `explanation` → `user correction` → `updated JSON` → `re-run`.

Deliverables. `llm_bridge/explain/explain.py` + demo notebook `experiments/day10_hitl_demo.ipyn`

12 Daily Checklist (Quick Reference)

Day	Deliverables (minimal)
1	Jupyter notebook with Gymnasium + PettingZoo sanity loops.
2	Bandit + Value Iteration scripts & convergence plot.
3	MC prediction/control scripts & brief comparison to DP.
4	SARSA & Q-learning on `CliffWalking-v1` + return chart.
5	Dyna-Q script + sample-efficiency plot vs planning steps.
6	PPO on CartPole: learning curves, rollout video/gif.
7	PPO vs SAC curves + chosen baseline rationale.
8	Independent PPO on MPE `simple_spread`: training + eval gif.
9	Goal JSON parser & 3 unit tests.
10	HITL demo: instruction → JSON → run → explanation → correction → improved run.

13 Evaluation Metrics (to reuse in your FYP)

Use consistent metrics across baselines and ablations.

- **Task performance:** episodic return, success rate, time-to-goal, collision count.
- **Learning:** sample efficiency (steps to reach threshold), variance over seeds.
- **HITL quality:** instruction-to-goal validity (JSON pass rate), explanation length and rule coverage, number of human corrections to achieve target behavior.
- **Computational:** wall-clock time per episode, VRAM, LLM latency.

14 Notes on Scope Management

- Choose **one** domain for the first demo (MPE is perfect). Add cyber-defense or search-and-rescue later.
- Prefer **prompt-level** guidance first (no fine-tuning). If needed, distill rules into a small policy later.
- Keep **strict JSON** for all LLM I/O. Fail fast on schema mismatches.

15 Appendix A: Tiny PPO Script (SB3)

Listing 4: PPO on CartPole (minimal)

```
import gymnasium as gym
from stable_baselines3 import PPO

env = gym.make("CartPole-v1")
model = PPO("MlpPolicy", env, verbose=1, n_steps=2048, batch_size=64,
    gamma=0.99)
model.learn(total_timesteps=100_000)
model.save("ppo_cartpole")
```

16 Appendix B: Minimal PettingZoo Loop

Listing 5: Random loop over PettingZoo MPE simple_v3

```
from pettingzoo.mpe import simple_v3
import numpy as np

env = simple_v3.env(render_mode=None)
env.reset(seed=42)
for agent in env.agent_iter(max_iter=200):
    obs, reward, terminated, truncated, info = env.last()
    action = env.action_space(agent).sample() if not terminated else
        None
    env.step(action)
env.close()
```

17 Appendix C: Prompt Sketches (LLM Bridge)

Instruction → JSON (GoalSpec)

```
System: You generate ONLY a JSON object that matches this schema:
{ "task_name": "mpe_simple_spread", "target_zone": "...", "formation": "...",
  "max_steps": <int>, "priorities": [strings], "constraints": [{"name": "...",
  "value": <float>, "note": "..." }] }
User: Focus on the left-most target, avoid collisions, use at most 200 steps.
```

Episode Explanation

```
System: Summarize the episode in <=5 sentences:
- Restate the goal and key constraints.
- Describe main behaviors and any violations (collisions, wrong zone).
- Give one actionable improvement hint.
```

18 (Optional) Further Reading Map

- **Sutton & Barto (2e draft).** Ch. 1–6 (tabular foundations), Ch. 8 (Dyna), Ch. 9–11 (approx & policy gradients).
- **ENSIA slides.** `1-Intro`, `2-Bandits`, `4-DP`, `5-MC`, `6-TD`, `7-Dyna`, `8,9-Approx`, `11-Adv RL`.
- **LLM-agents inspiration (for literature review).** Zero-shot planning with LLMs; ReAct-style reasoning+acting; Reflexion-style self-critique.

By Day 10 you will have: (i) a clean bandit→DP→MC/TD→Dyna→deep ladder, (ii) a runnable MARL toy task, (iii) an LLM bridge for goals & explanations with a small HITL loop.