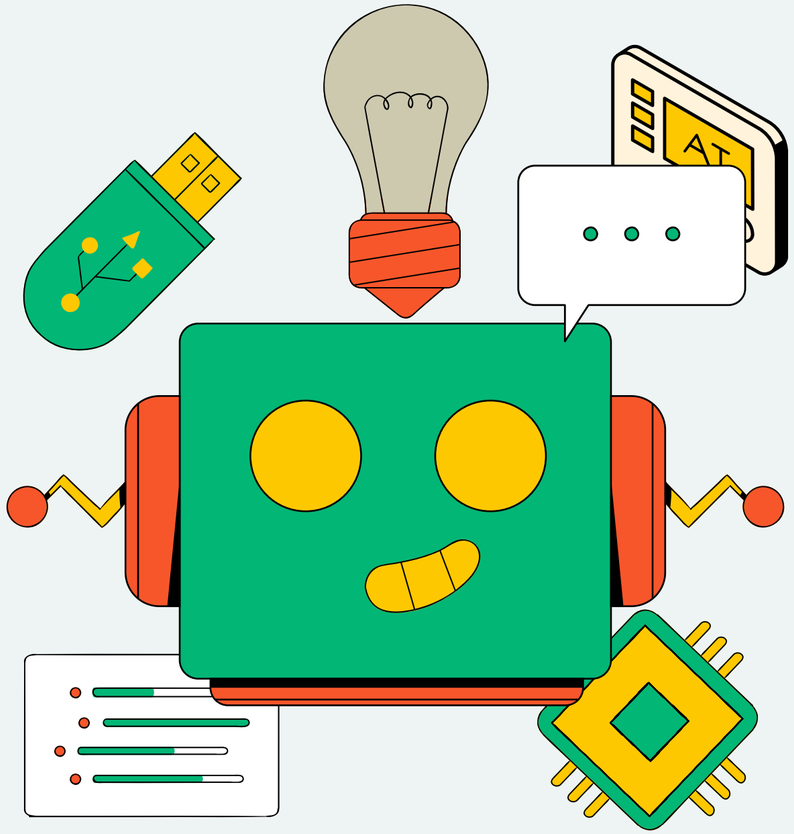
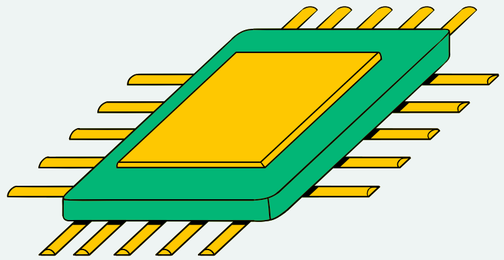


# DISTRIBUTED COORDINATION SYSTEM FOR MATERIAL TRANSPORT IN A WAREHOUSE




PRESENTED BY:

EMANUELE DI LUZIO

# PRESENTATION OUTLINE

- Project Overview
- Simulation Environment
- Reinforcement Learning Techniques
- Implementation Details
  - Negotiation Mechanism
  - Training Process
  - Representation with Pygame
- Transition from Centralized to CTDE Approach(and comparison)
- Results
- Conclusion
- Future Improvements

# PROJECT OVERVIEW

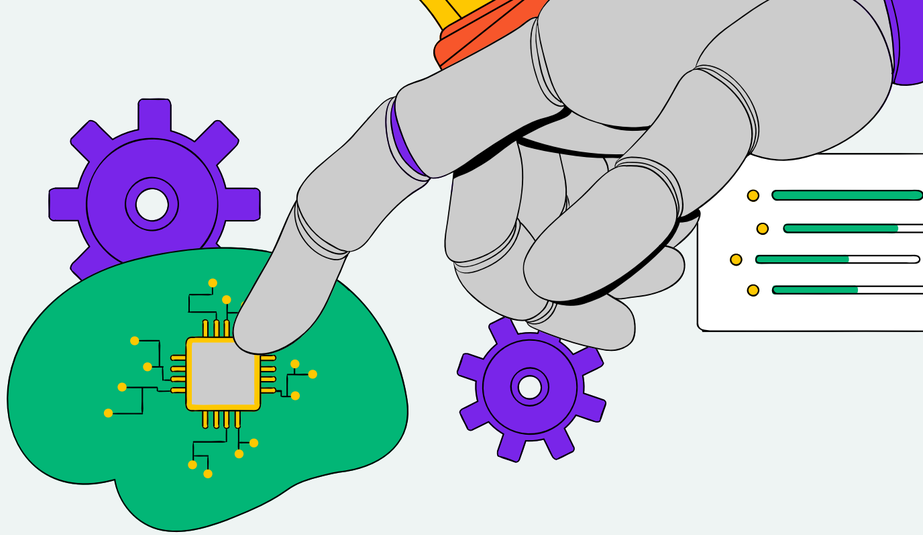
## Main Objective:

- Optimize material transport within a warehouse using autonomous robots. This project leverages distributed artificial intelligence (DAI) to enable robots to collaborate effectively in executing pick-and-deliver tasks while optimizing resource usage and ensuring robust coordination.

**Key Features**:

- Simulation Environment: A **grid-based warehouse** layout with randomly distributed, pickup zones, drop off zones,  obstacles, materials, and charging stations.
- **Reinforcement Learning (RL/MARL)**: Implementation of Proximal Policy Optimization (PPO) for training robots in a cooperative, multi-agent setting.
- **Communication Protocols**: Robots exchange essential task and status information using a bit-based message system.

# SIMULATION ENVIRONMENT



**Main Components:**

**Grid Layout**: A 2D space with adjustable dimensions, populated by robots, materials, obstacles, and charging stations.

**State Representation**:

1. Robot position, battery level, and communication status.
2. Task locations and statuses (picked or delivered).
3. Visual Representation: Rendered using pygame, featuring clear visual indicators for robots, tasks, and other critical components.

# REINFORCEMENT LEARNING TECHNIQUE: PPO

**Proximal Policy Optimization (PPO):**

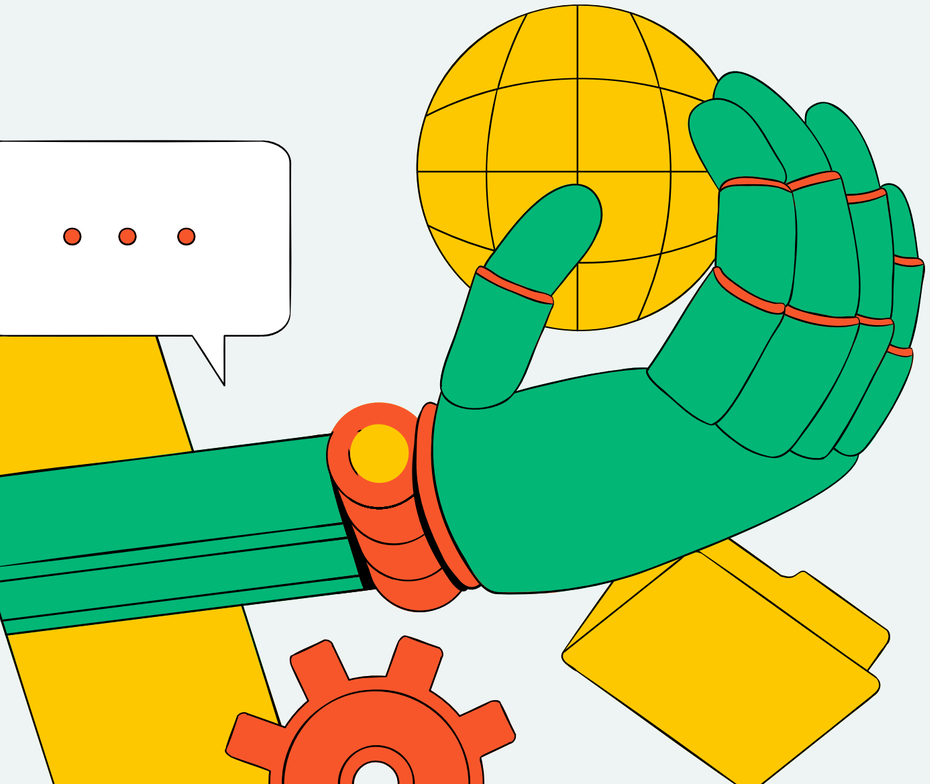
- A policy optimization algorithm selected for its ability to balance stability and performance in complex environments.

- **Implementation Details:**
  - **Model Architecture**: Neural networks with two hidden layers of 256 neurons each, activated by ReLU functions.
  - **Hyperparameters:**
    - Learning rate: Linear schedule from 0.0003 to 0.0001.
    - Rollout size: 2048 steps.
    - Batch size: 512.
  - **Training:** Conducted over 2,000,000 timesteps, using Stable-Baselines3.

**Reward Shaping:**

- **Positive Rewards**: For task pickups, deliveries, and proximity to goals.
- **Penalties**: For inactivity, collisions, and inefficient behaviors.
- **Communication Coherence**: Small rewards or penalties for accurate or inconsistent message bits.

# IMPLEMENTATION DETAILS



**Core Components:**

1. **warehouse_env.py**
   - Defines the warehouse environment using **PettingZoo's ParallelEnv framework.**
   - **Includes robust reward shaping and agent interaction logic.**
2. **wrappers.py**
   - Provides a wrapper to transform the multi-agent environment into a single-agent Gym-compatible interface for PPO training.
3. **ppo_training_script.py**
   - Central script for model training.
   - Includes custom callback for progress visualization and result logging.
4. **actions.py**
   - Enumerates robot actions, including movement, loading, and communication.
   - Supports advanced actions like SEND_BATTERY_STATUS and REQUEST_HELP.

# NEGOTIATION MECHANISM

**Communication Protocols:**

1. **Task Assignment:**
    - The **Contract Net Protocol** is used to allocate tasks dynamically based on the robots' capabilities and current states.
    - Criteria include proximity to the task, battery levels, and task urgency.

1. **Message-Based Coordination:**
    - Robots exchange messages using a 16-bit array to communicate essential information, such as:
        - Battery status.
        - Requests for assistance (e.g., REQUEST_HELP).
        - Path availability (e.g., SEND_PATH_BLOCKED).
    - **Coordination logic ensures that high-priority tasks are completed first, and resources like battery levels are effectively utilized.**

**Resource Management:**

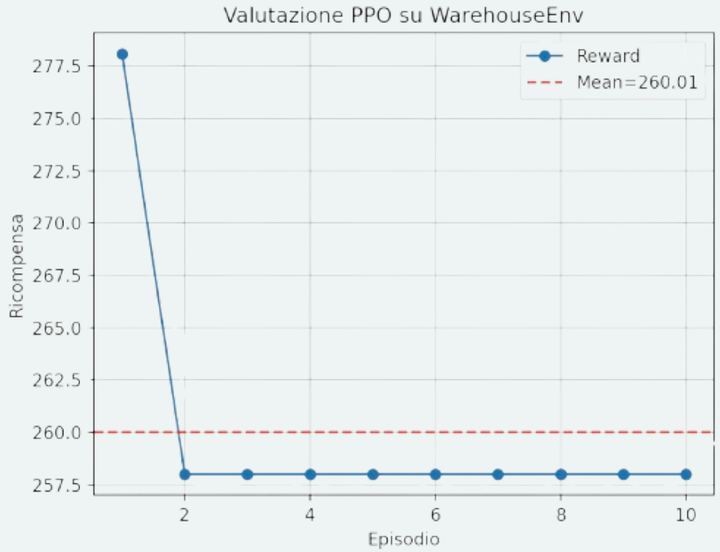
- **Task Prioritization: Robots dynamically reassign tasks based on environmental changes, such as obstacles or robot malfunctions**(Robots with low battery prioritize charging and defer tasks to other agents.).

# TRAINING PROCESS

- **Environment Preparation:**
  - The **ParallelEnvToGymEnv wrapper** is used to convert the multi-agent WarehouseEnv into a Gym-compatible environment suitable for single-agent PPO training.
  - **Each robot's local observation is concatenated into a single observation vector.**

- **Algorithm Configuration:**
  - Learning Rate: Scheduled to decay linearly during training.
  - Neural Network Architecture: Two layers of 256 neurons each with ReLU activation.
  - Batch and Rollout Sizes:
    - Rollout size: 2048 timesteps.
    - Batch size: 512 samples.
  - A tqdm-based callback tracks the training process in real-time.

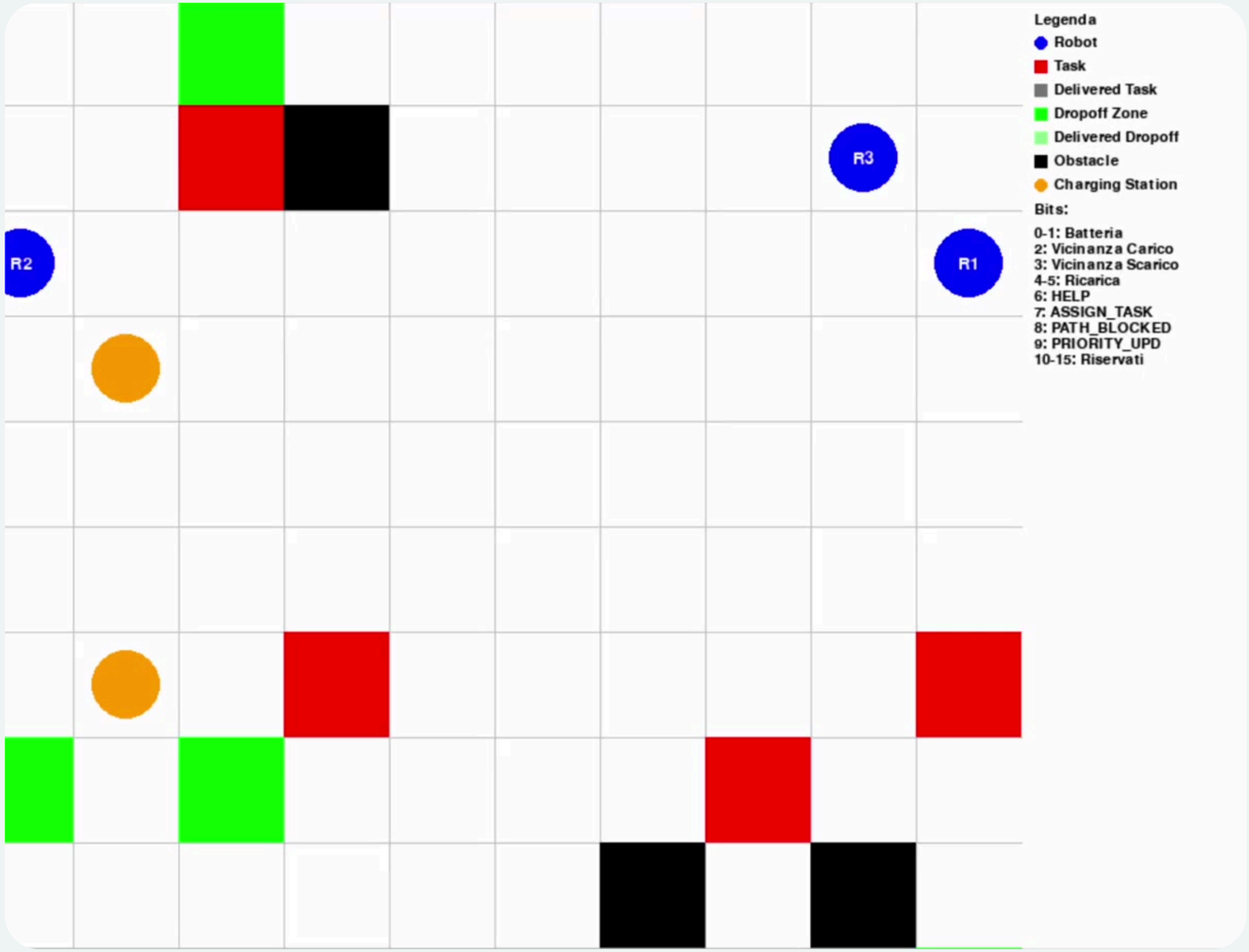
- **Evaluation Metrics:**
  - **The model's reward is evaluated every 100,000 timesteps.**
  - Test episodes are conducted to measure task completion rates and policy stability.




**Post-Training Evaluation:**

- Model Testing:
  - Conducted over 10+ episodes to verify task completion rates and reward stability.
  - Visual inspections using environment rendering confirm cooperative behaviors among robots.
  - Hyperparameters such as the learning rate or reward shaping can be adjusted to address specific shortcomings.

# REPRESENTATION WITH PYGAME




1. **Grid Rendering:**
   - A clear 2D grid layout representing the warehouse.
   - Cells distinguish obstacles, charging stations, task locations, and drop-off zones.

1. **Robot Visualization:**
   - **Robots** are displayed as **circles with unique IDs for easy identification.**

- Real-time visualization allows observation of robot behaviors during task execution.
- Debugging through visual cues, such as collision highlights or task completion indicators.

# TRANSITION FROM CENTRALIZED TO CTDE APPROACH

**Current Setup:**

- **Until now, we have implemented a centralized learning approach**, where both training and execution rely on a global controller. This means:
- **Centralized Training**: A single neural network (global policy) is trained using the concatenated observations of all robots.
- **Centralized Execution**: The same policy is deployed, and a central controller decides the actions for all robots based on the global observation.

**Limitations:**

- **Scalability Issues**: As the number of robots increases, the size of the global observation grows, making it computationally expensive.
- **Unrealistic Assumptions**: Centralized execution **assumes perfect and instant communication between robots and the central controller**, which is **often infeasible in real-world settings.**

**Shift to CTDE:**

To address these challenges, we are transitioning to a CTDE (Centralized Training and Decentralized Execution) paradigm. This approach introduces:

- **Local Observations:** Each robot operates based on its **local view of the environment**, aligning with real-world constraints.
- Centralized Critic: **During training, a centralized critic uses the global state to evaluate and improve the decentralized policies of individual robots.**
- This transition will make **the system more scalable and realistic**, allowing robots to act autonomously while still benefiting from centralized insights during training.

# COMPARISON: CENTRALIZED VS. CTDE (MAPPO)

## ARCHITECTURAL DIFFERENCES:

**Centralized Approach:**

- Observations: Global observation concatenating all robot data.
- Policy: Single neural network acting as a global controller.
- Execution: Centralized control with a single decision-maker.

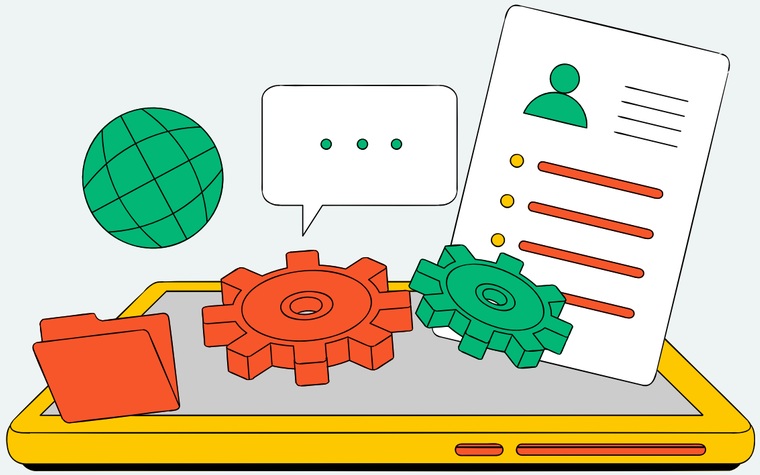
**CTDE (MAPPO):**

- Observations: Local observations for each robot.
- Policy: Decentralized policies, with a centralized critic for training.
- Execution: Fully distributed, where each robot operates independent

## ADVANTAGES AND LIMITATIONS:

- **Centralized Approach:**
  - Pros: Simple to implement and train; suitable for small-scale scenarios.
  - Cons: Limited scalability and unrealistic in decentralized real-world environments.
- **CTDE (MAPPO):**
  - Pros: Scales efficiently; realistic for autonomous robot systems.
  - Cons: Requires more complex implementation and training setups.

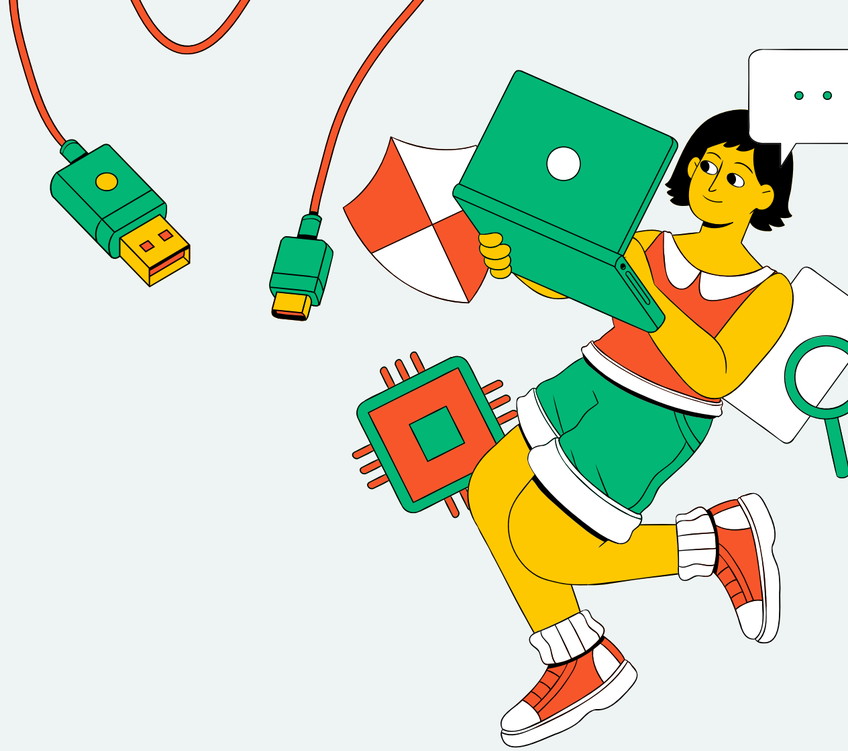
# RESULTS



**Performance Metrics:**

1. **Task Completion Rate**:
   - PPO (Centralized): Achieved an average task completion rate of approximately 80% across 10 test episodes in smaller environments (3 robots).
   - CTDE (MAPPO): Improved scalability, achieving approximately 84% task completion in environments while maintaining performance stability.
2. **Efficiency**:
   - CTDE (MAPPO): Further reduced task times leveraging **localized decision-making to minimize idle time and optimize resource use in larger environments.**

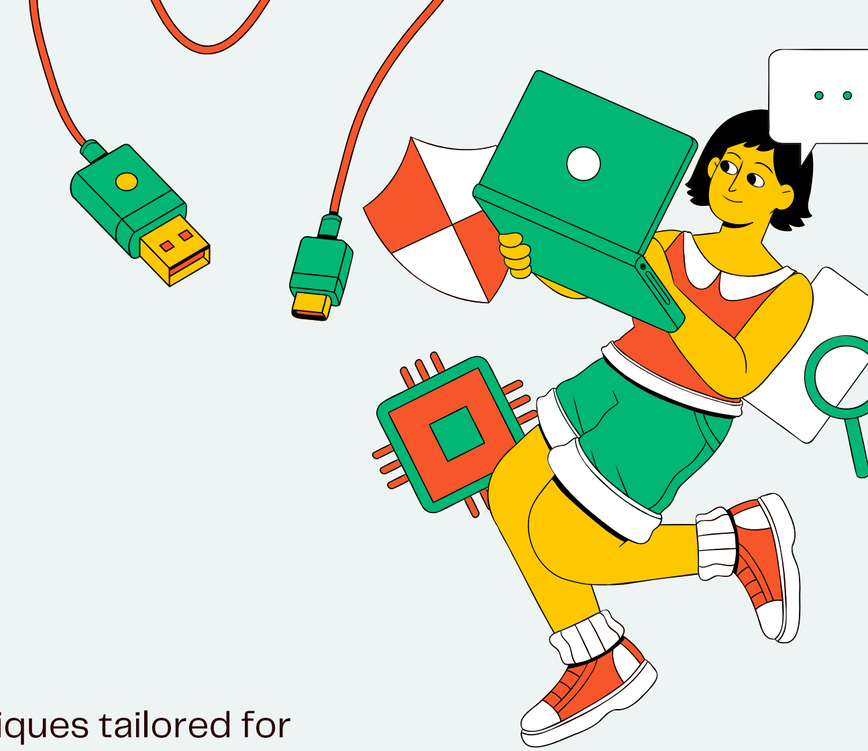
# CONCLUSION



This project demonstrates the efficacy of distributed artificial intelligence and reinforcement learning for multi-robot coordination in warehouse logistics. Key outcomes include:

- Improved Efficiency: **Both centralized and CTDE methods achieved high task completion rates, but CTDE provided superior scalability and adaptability.**
- Realistic Execution: The CTDE paradigm aligned more closely with real-world constraints, allowing decentralized decision-making while leveraging centralized training benefits.
- Structured Learning: **Reward mechanisms and communication protocols ensured efficient and cooperative behaviors across different scenarios.**

# FUTURE IMPROVEMENTS



**Scalability**: Extend the CTDE framework to support:

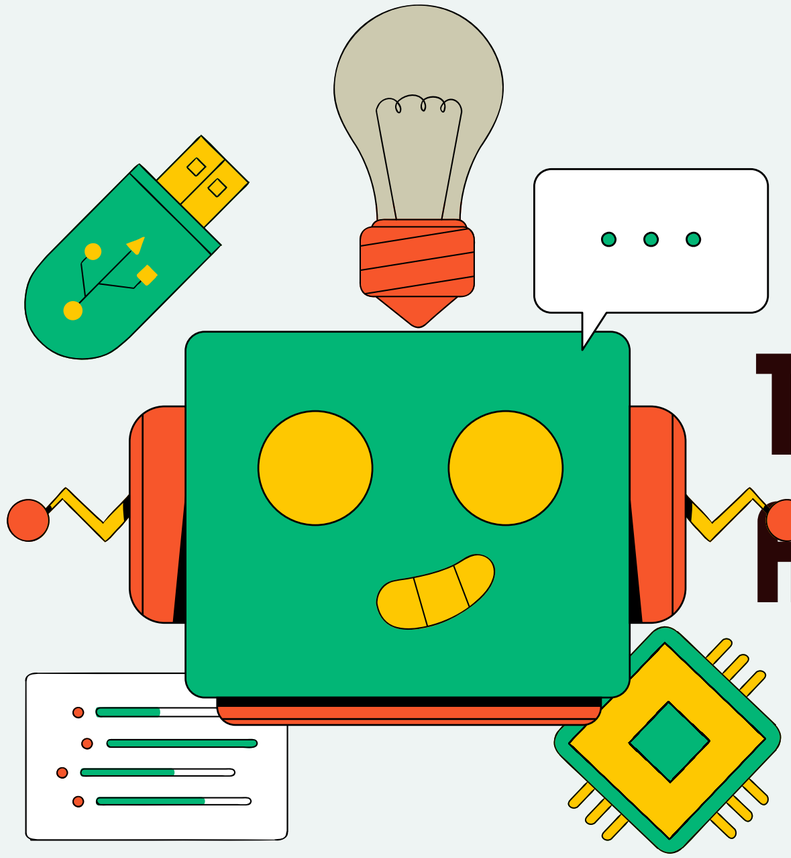
- **Larger warehouses** with more complex task layouts.
- An increased number of robots while maintaining efficient coordination.

**Advanced Communication Protocols**: Develop sophisticated message-passing techniques tailored for decentralized systems:

- **Hierarchical or auction-based task** allocation to improve **efficiency.**
- Improved communication protocols that minimize overhead while maintaining effectiveness.

**Real-Time Adaptation: Leverage online learning strategies within CTDE to**:

- Enable robots to adapt their policies dynamically to changing environments.
- Ensure robust performance in unforeseen scenarios, such as new obstacles or failures.



# THANKS FOR THE ATTENTION