Tecnológico de Monterrey

Evidencia 1. Actividad Integradora

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 302)

Jesús I. Hernández | Iván A. Dounce | Javier F. Rendón | Eduardo Morales

José Emilio Inzunza García | A01644973

Yael García Morelos | A01352461

Juan Pablo Torres Guillén | A01743268

Liliana Ramos Vázquez | A01644969

Diana Fernanda Delgado Salcedo | A01644911

24 de noviembre del 2025

Objetivo y solución del problema:

Esta solución presenta el diseño, arquitectura y comportamiento de un sistema multiagente orientado a automatizar la organización de un almacén mediante cinco robots autónomos. El entorno consiste en una matriz bidimensional donde se distribuyen cajas en posiciones aleatorias. El objetivo general del sistema es agrupar todas las cajas en pilas ordenadas de máximo cinco unidades dentro de una zona de entrega designada.

La solución se basa en un enfoque descentralizado, donde cada robot percibe el entorno, planifica rutas y toma decisiones de manera independiente, utilizando técnicas de pathfinding y coordinación indirecta. La estrategia fomenta la eficiencia, reduce colisiones y evita bloqueos, optimizando el desempeño colectivo del sistema.

Protocolo de los agentes:

Cada robot está diseñado mediante una Máquina de Estados Finitos (FSM) que define de forma clara cada fase de comportamiento, sus acciones y transiciones. Esto asegura que el flujo del robot sea consistente y fácil de modelar.

Estados del agente robot:

1. **SEARCH: Búsqueda:**
    - **Acción:** Recorre la matriz escaneando celdas con cajas.
    - **Lógica:** Calcula la distancia a todas las cajas y selecciona la más cercana.
    - **Transición:**
        - Si encuentra caja: Calcula ruta BFS: pasa a GOING_BOX.
        - Si el mapa cambia o se bloquea: permanece en SEARCH.
2. **GOING_BOX: Desplazamiento hacia la caja:**
    - **Acción:** Se mueve paso por paso siguiendo la ruta BFS.
    - **Validación:** En cada paso verifica que la celda destino esté libre.
    - **Transición:**
        - Si llega adyacente: recoge la caja: PLAN_DROP.
        - Si la ruta se bloquea o la caja desaparece: volver a SEARCH.
3. **PLAN_DROP: Selección de pila de entrega:**
    - **Acción:** Evalúa la zona de entrega.

- **Lógica:**
    - Prioriza pilas semi-completas.
    - Si no existen, inicia una nueva.
- **Transición:**
    - Si encuentra una pila válida: Calcula la ruta BFS: GOING_DROP.

4. **GOING_DROP: Transporte hacia la zona de entrega:**
    - **Acción**: Se desplaza hacia la pila objetivo.
    - **Transición**:
        - Al llegar adyacente: suelta caja: SEARCH.
        - Si la pila se llena o hay bloqueo: PLAN_DROP.

Estrategia cooperativa para la solución del problema:

El enfoque cooperativo implementado utiliza una arquitectura reactiva descentralizada, donde los robots no intercambian mensajes directamente, sino que emplean un mecanismo en el que la información se transmite a través del entorno compartido (la matriz del almacén).

Los robots comparten el mismo mapa global (warehouse matrix). Este funciona como medio indirecto de comunicación:

- Cuando un robot recoge una caja, actualiza esa celda a vacío.
- Si otro robot tenía seleccionada la misma caja como objetivo, al observar el cambio en el entorno:
    - Descarta su objetivo anterior.
    - Recalcula un nuevo destino automáticamente.

De esta manera, se evita la necesidad de un agente coordinador, subastador o servidor de tareas.

El sistema implementa dos mecanismos clave:

1. **Planificación Dinámica con BFS:** Cada robot utiliza BFS para calcular rutas hacia cajas o pilas de entrega. Durante la planificación, trata a otros robots como obstáculos temporales, lo cual:
    - Previene colisiones.

- Distribuye mejor las cargas de trabajo.
- Evita rutas que pudieran colapsar el flujo.

2. **Detección y Resolución de Atascos:** Si un robot no avanza durante 3 segundos:
    - Activa un periodo aleatorio de espera (wait_frames) para desincronizarse.
    - Recalcula la ruta ignorando temporalmente a los otros robots, con la expectativa de que se moverán antes de que llegue.

Estos mecanismos garantizan fluidez y reducen bloqueos en espacios estrechos.

Simulación en Python:

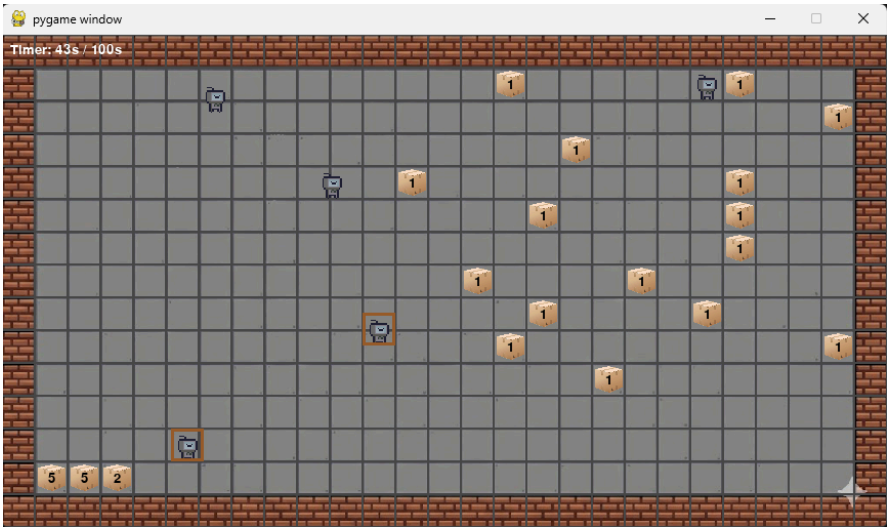


Diagrama de Clases:

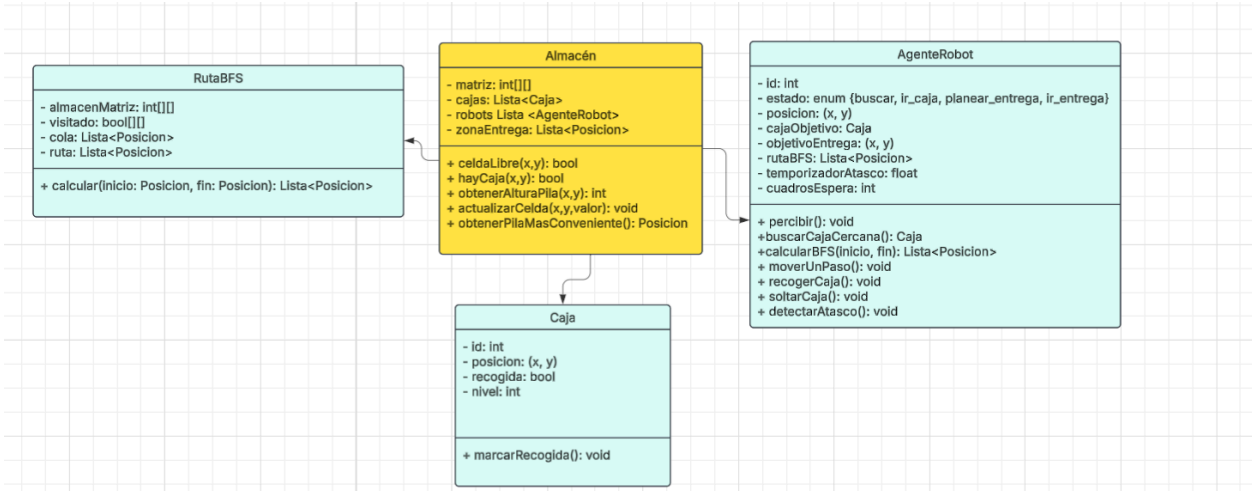
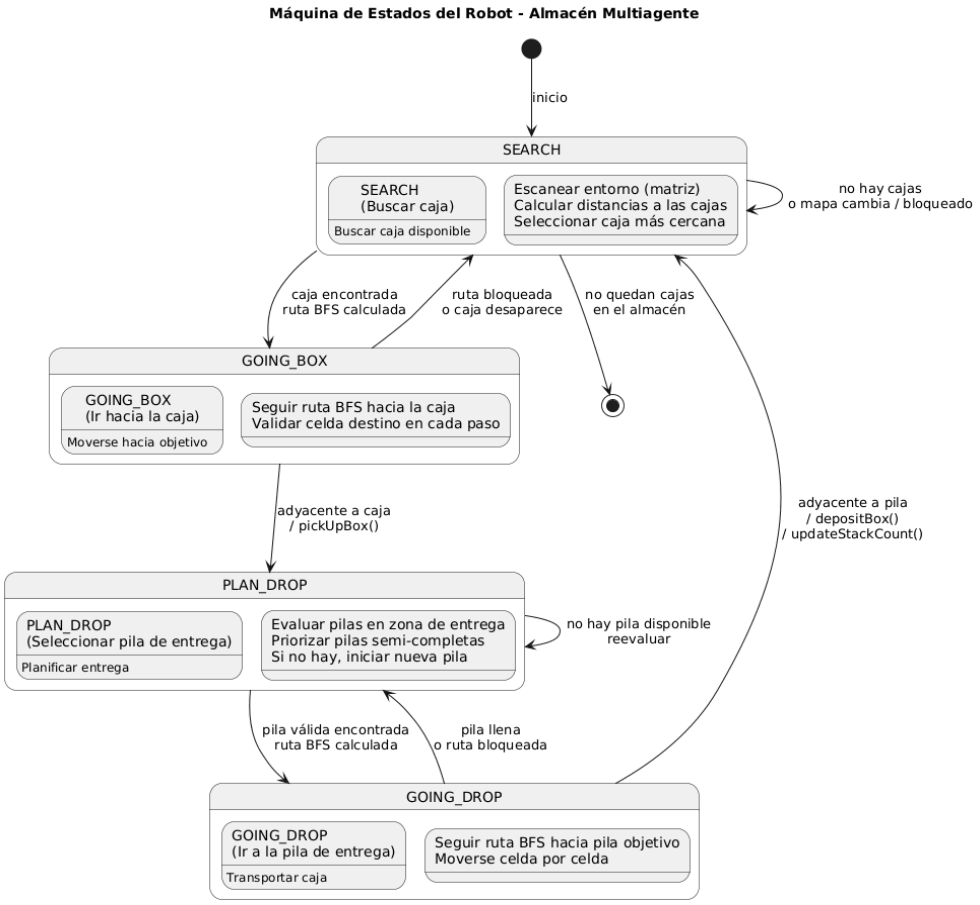


Diagrama de Flujo (Máquina de estados del agente):



Warehouse Virtual Environment Unity.