Simulation of an autonomous and adaptive robotic arm

Juan Diego Lozada

Cristian Santiago López Cadena

Universidad Distrital Francisco Jose de Caldas

System Sciences Foundation

Carlos Andrés Sierra

09 de Abril del 2025

Systems Design Framework

Requirements

Functional Requirements.

- As a system developer, I want the robotic arm to detect and identify objects using visual input, so that it can interact intelligently with its environment.
  Acceptance Criteria: Given the visual feed from the camera, When an object enters the frame, Then the system should detect and correctly identify the object's class.

- As a developer, I want the system to locate objects in the workspace using image processing, so that the arm can interact accurately.
  Acceptance Criteria: Given the presence of an object in the workspace, When the system processes the frame, Then it must return the object's location coordinates.

- As a developer, I want the system to classify objects by size, shape, or type, so that the grip can be adjusted accordingly.
  Acceptance Criteria: Given an identified object, When it is analyzed by the classifier, Then the object should be correctly categorized (e.g., small/cylindrical/metal).

- As a system developer, I want the agent to calculate the best gripping point based on geometry and orientation, so that it ensures a successful grip.
  Acceptance Criteria: Given an object with known geometry and position, When the agent processes the input, Then it must return a valid grip point with coordinates and angle.

- As a system developer, I want the robotic arm to plan and follow a collision-free path, so that it can pick and place objects safely.
  Acceptance Criteria: Given the object location and destination, When the agent plans the trajectory, Then the movement must not collide with any obstacles.

- As a developer, I want the arm to adjust grip strength and angle using sensor feedback or object material, so that it handles different items securely.
  Acceptance Criteria: Given feedback from sensors, When the object is being gripped, Then the grip parameters must adjust accordingly.

- As a developer, I want the system to evaluate grip success, so that future performance improves with experience.
  Acceptance Criteria: Given an executed grip action, When the system checks placement outcome, Then it should log a success or failure and update the learning policy.

- As a stakeholder, I want the arm to retry with a different grip if the first attempt fails, so that it increases the chance of success.
  Acceptance Criteria: Given a failed grip attempt, When the system detects the failure, Then it must attempt an alternative gripping strategy.

Non-Functional Requirements.

- As a stakeholder, I want the system to respond to visual input in under 1 second per frame, so that the process runs in near real-time.
  Acceptance Criteria: Given a new frame from the camera, When the frame is processed, Then the system must respond in less than 1 second.

- As a stakeholder, I want the object recognition module to be at least 90 % accurate, so that the system can make reliable decisions.
  Acceptance Criteria: Given a set of test objects, When the system classifies them, Then the accuracy must be 90 %.

- As a stakeholder, I want the robotic arm to successfully complete pick-and-place tasks at least 98 % of the time after training, so that the process remains efficient.

Acceptance Criteria: Given 100 attempts post-training, When tasks are executed, Then at least 98 must succeed.

- As a stakeholder, I want the system to adapt to new object types incrementally, so that we avoid full retraining.
  Acceptance Criteria: Given new object characteristics, When introduced to the system, Then the model should update incrementally without retraining from scratch.

- As a system developer, I want the architecture to be modular, so that we can independently update components like vision or motion planning.
  Acceptance Criteria: Given an update to the computer vision module, When it is integrated, Then the motion planning module should remain unaffected.

- As a developer, I want to visualize logs and metrics, so that I can track and evaluate the learning process.
  Acceptance Criteria: Given a training session, When the system is running, Then it must generate logs and graphs of performance metrics.

- As a developer, I want the system to be compatible with standard computing hardware, so that we don't require expensive equipment.
  Acceptance Criteria: Given a mid-range machine, When the model is trained and executed, Then the system must run without crashing or excessive delays.

Use Cases

- Use Case 1: The system captures images of the environment and detects the presence of objects in the scene. To do this, the system first captures an image from the camera, then applies image processing techniques, and finally identifies the regions corresponding to objects.

- Use Case 2: The system determines the object type based on various parameters such as size or shape using ML models or neural networks. To do this, the system receives the region of the detected object, extracts relevant features such as shape, and finally uses a trained model to classify the object.

- Use Case 3: Determine the object's center of mass and geometry to select the best grip point. To do this, start with the object's silhouette or 3D shape, calculate its centroid, and evaluate possible grip points. The most stable one is chosen based on balance and shape.

- Use Case 4: Calculate the path that the robotic arm must follow to reach the grip point, avoiding obstacles. Based on the grip point, the path to this point is calculated, avoiding possible obstacles and staying within the arm's range of action.

- Use Case 5: Grasp the object, once the arm is positioned in the gripping position, the orientation of the gripper is adjusted, then it closes with the estimated force based on the object's material, the sensors verify whether the grip was successful.

- Determine the best path from the current position to the deposit point, once the object is secured in the gripper. To do this, based on images of the environment and different sensors, the best possible path is calculated, avoiding collisions and taking into account the movement limitations of the arm, such as the speed of movement and the field of action.

- Release the object in the target position. To do this, the final position is verified. Once it is adequate, the gripper opens in a controlled manner and images are used to verify whether the object was correctly placed in the expected position.

Components

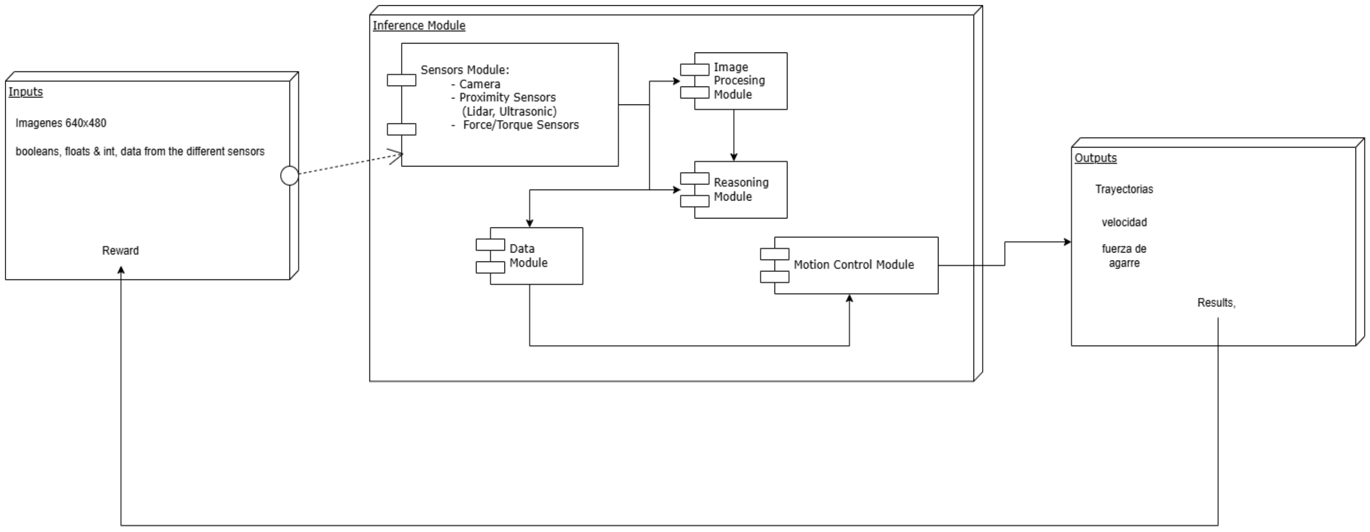

Figura 1

Component diagram

Sensor Module.

- Camera: Detect the object and analyze the shape of it
- Force/Torque Sensors: Pure Force, measure the pression made by the fingers, avoiding break the object; Pure torque Sensors, enable a sense of the torque being applied or experienced by a section of a robotic arm; Force/Torque (FT)
- Proximity Sensor: Calculate distance between the arm and the object

Image Processing Module.

- Object Detection: Use OpenCV to identify objects could appear inside the area of the camera, through contours and segmentation
- Object Classification: Implement AI models (TensorFlow) to recognize object types and determine optimal gripping strategies.
- Object Position: Calculate real-time object coordinates to optimize the robotic arm's approach and gripping precision.

Reasoning module.

- Reinforcement Learning: Train the robotic arm using reward-based feedback to improve object gripping efficiency over time.
- Environment Simulation: Use Gym/OpenAI Gym to train the robotic arm in a controlled, physics-based virtual environment.

Motion Control Module .

- Trayectory generation: Adjust paths without collision using reinforcement learning
- Grip Strenght & angle adjustment: Adjust the strenght depend on the object and the angle with the correct way to grab, and use reinforcement learning to improve the process

Data Module.

- Sensor Data Processing: Captures and filters data from the camera, force, and proximity sensors
- Grip Force Analysis: Records applied pressure to optimize gripping

Feedback Loops

- Reinforcement Reward Loop: Assigns rewards based on successful gripping and movement efficiency
- Trajectory Correction Loop: Adjusts the path in real-time to avoid errors or obstacles
- Vision-Based Feedback Loop: Recalculates object position if it moves unexpectedly

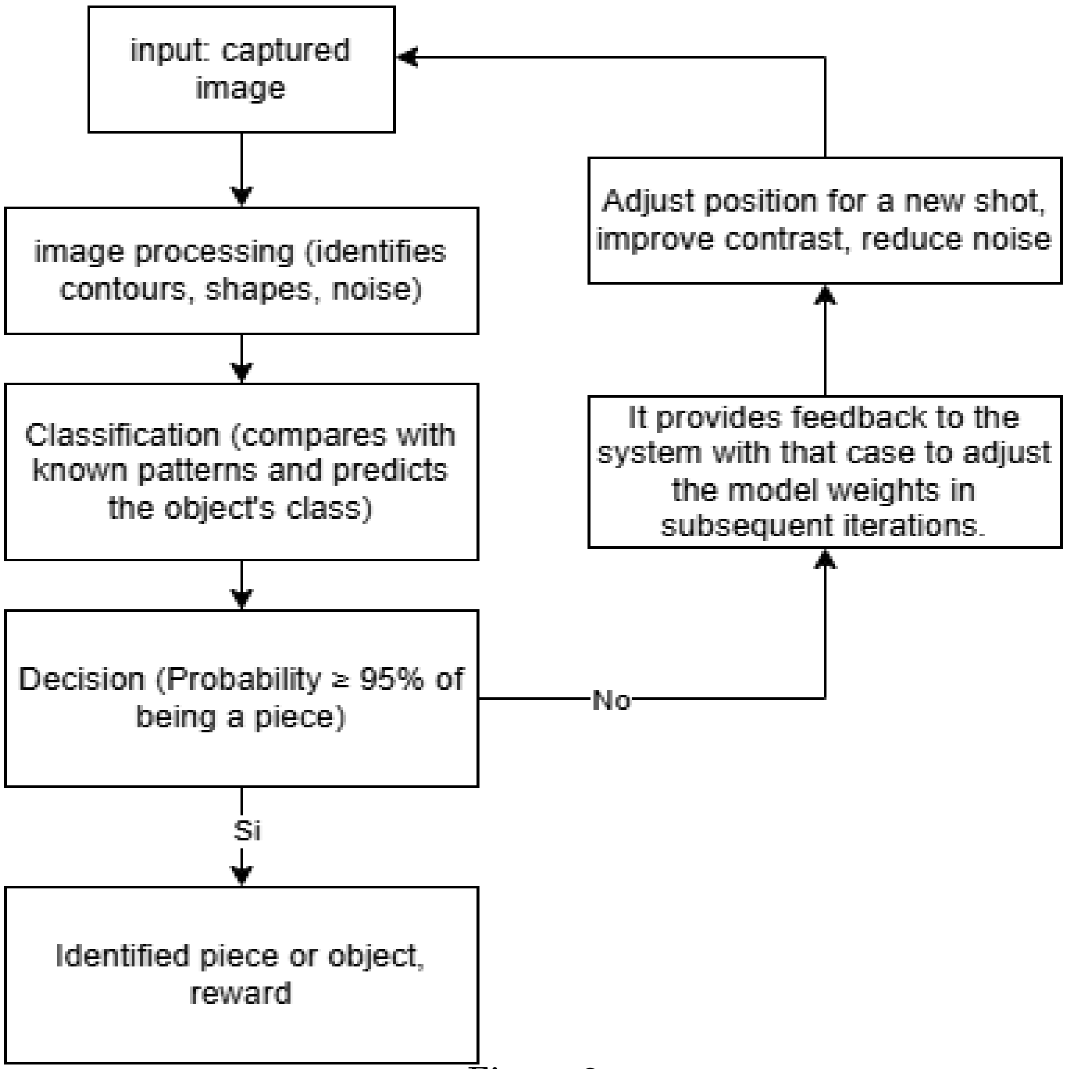
input: captured image
image processing (identifies contours, shapes, noise)
Classification (compares with known patterns and predicts the object's class)
Decision (Probability ≥ 95% of being a piece)
Si
Identified piece or object, reward
No
It provides feedback to the system with that case to adjust the model weights in subsequent iterations.
Adjust position for a new shot, improve contrast, reduce noise

Figura 2

feedback loop identify object/part

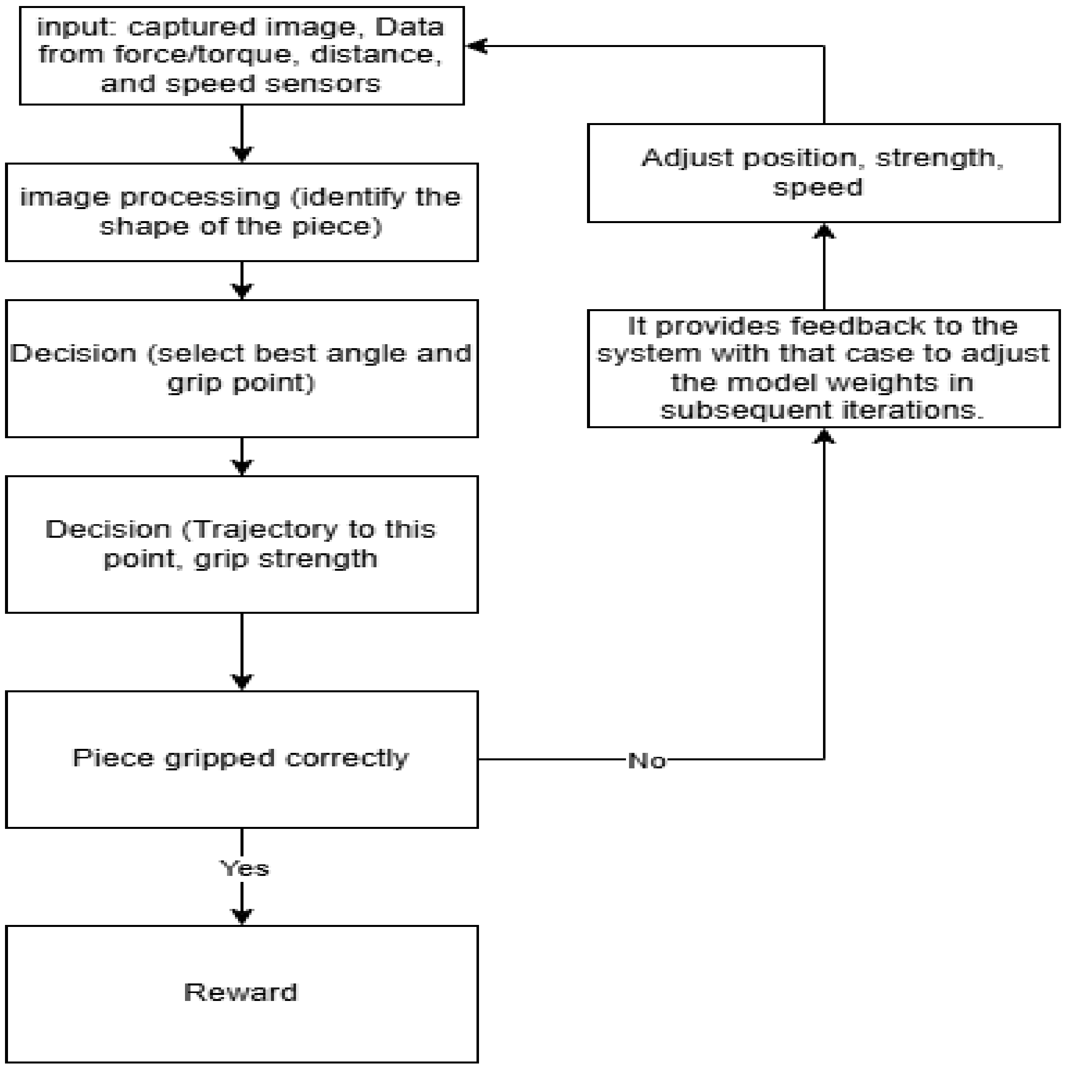
input: captured image, Data from force/torque, distance, and speed sensors
image processing (identify the shape of the piece)
Decision (select best angle and grip point)
Decision (Trajectory to this point, grip strength
Piece gripped correctly
Yes
Reward
No
It provides feedback to the system with that case to adjust the model weights in subsequent iterations.
Adjust position, strength, speed

Figura 3

feedback loop grab piece/object

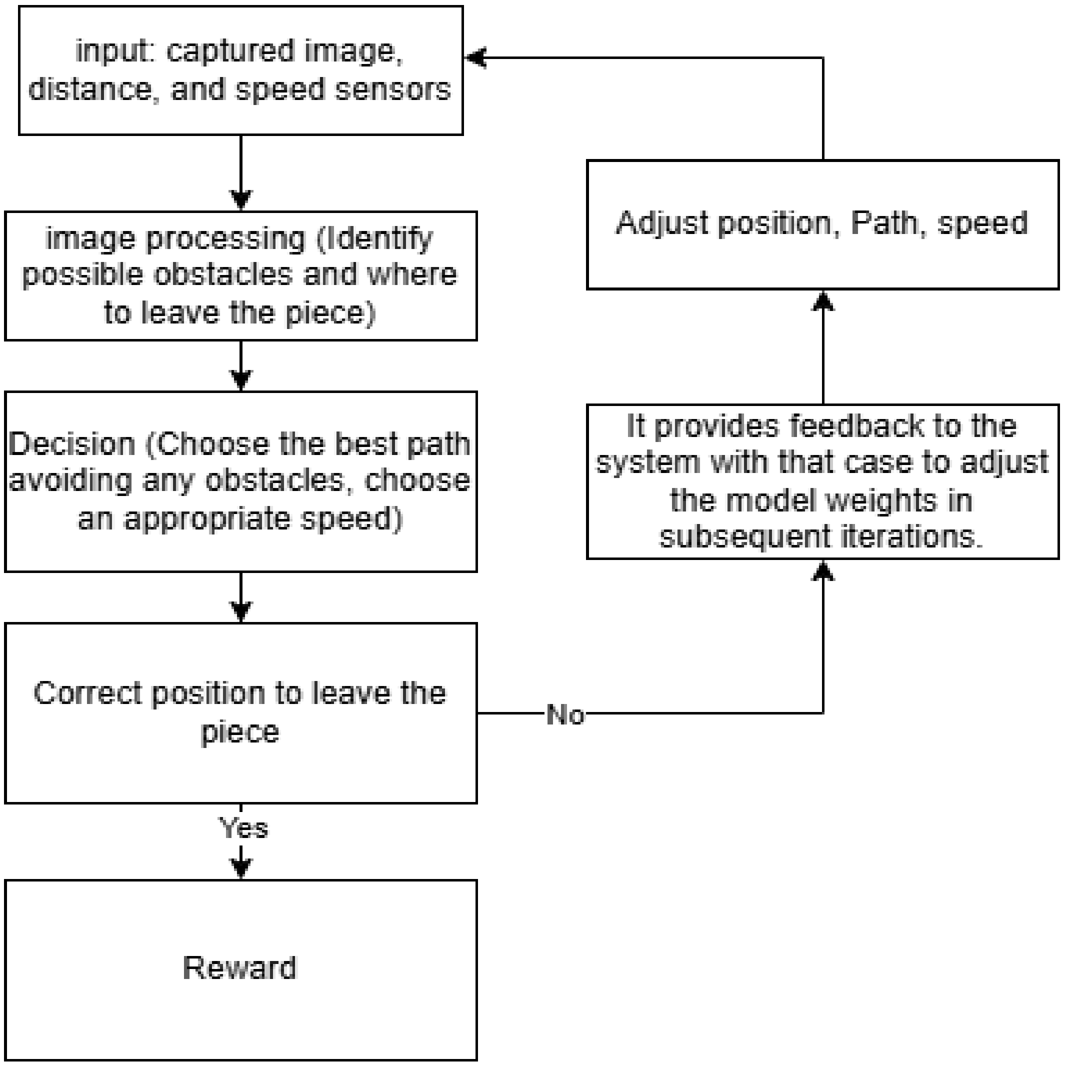
input: captured image, distance, and speed sensors
image processing (Identify possible obstacles and where to leave the piece)
Decision (Choose the best path avoiding any obstacles, choose an appropriate speed)
Correct position to leave the piece
Yes
Reward
No
It provides feedback to the system with that case to adjust the model weights in subsequent iterations.
Adjust position, Path, speed

Figura 4

feedback loop transporting part/object

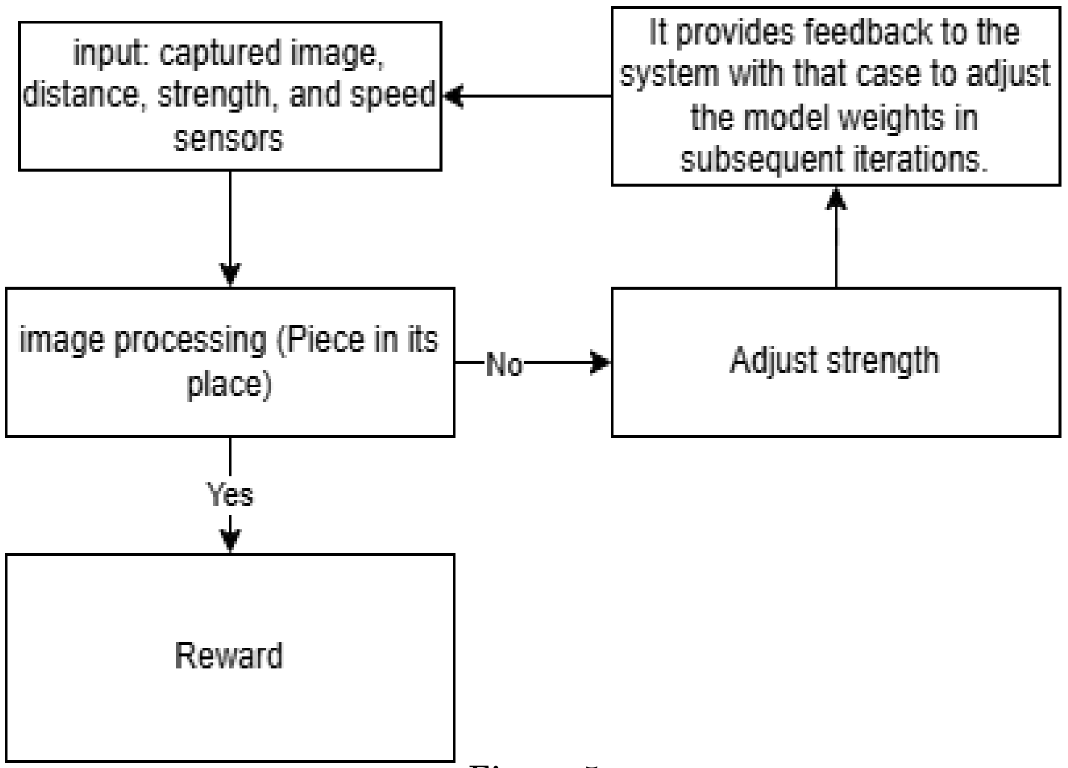
input: captured image, distance, strength, and speed sensors
It provides feedback to the system with that case to adjust the model weights in subsequent iterations.
image processing (Piece in its place)
No
Adjust strength
Yes
Reward

Figura 5

feedback loop drop piece/object

Potential frameworks

Some of the frameworks we will implement in the project that could help us with specific tasks, an advantage about those frameworks is simple to integrate all of them, those are:

- Gymnasium: Standard Environment about develop and test with Reinforcement Learning (RL). This will give us a structure clear about agents with states, actions and rewards, and create personalized environments

- Stable-Baselines3: It is a implementation RL based on PyTorch, it's useful for robust algorithms, so, we could implement SB3 for Train agents to plan optimal grasping routes, adjust force, and improve over time using success/failure rewards in pick-and-place tasks.

- PyBullet: It is a powerful open-source physics simulator that accurately models robotic arms, forces, collisions, and 3D movements. It integrates easily with Gymnasium, making it ideal for reinforcement learning experiments. It provides a realistic environment to test robotic arm control, ensuring safe and efficient motion planning before real-world deployment. In this project, it complements the Gym environment by simulating physical dynamics, refining motor control strategies, and optimizing the robotic arm's gripping performance.

Timeline Project

Week	**Tasks**
Week 0	W1 Analyze Uses of Cases and Components
Week 1	Mathematical Model
Week 2	Refinement Feedback - Loop Agent Stability
Week 3	Framework implemented
Week 4	Simulation Parameters - Scenario variations
Week 5	ML integration
Week 6	Cybernetic Control Mechanisms
Week 7	Finalize and Refine Feedback
Week 8	Self-regulation, adaptability and resilience
Week 9	Test Cases/Evaluation
Week 10	Environment Setup
Week 11	Agent Definition - RL implementation
Week 12	Visualization - Metrics
Week 13	Testing & Validation - Record Video

Referencias

Imran, A. (2019). Training a robotic arm to do human-like tasks using rl. *DataDrivenInvestor*. Descargado de `https://medium.datadriveninvestor.com/training-a-robotic-arm-to-do-human-like-tasks-using-rl-8d3106c87aaf`

Pelegri, J. (2019). Espacio de trabajo de un robot: Solución a las dimensiones de plantas de producción. *Universal Robots*. Descargado de `https://www.universal-robots.com/es/blog/espacio-de-trabajo-de-un-robot/`

ROBOTICS, R. (2023). Force and torque sensors why are they of interest in robotics. *REACH ROBOTICS*. Descargado de `https://reachrobotics-com.translate.goog/blog/force-and-torque-ft-why-are-they-of-interest-in-robotics/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc`

Pelegri (2019) ROBOTICS (2023) Imran (2019)