Implementation

(a) Hardware Used:

- Rethink Sawyer robot with 7 DOF.
- Right gripper end-effector
- AR markers for block detection
- Right gripper camera system for AR marker detection.

(b) Parts Used:

- AR markers.
- Uniform blocks with AR markers mounted.
- Computing system running ROS with required packages:
  - moveit_msgs
  - sawyer_pykdl
  - ar_track_alvar
  - intera_interface
  - tf2_ros

(c) Software Architecture:

1. Perception System:
   - scan(marker_ids):
     - Orchestrates the entire perception process.
     - Uses a 5-point scanning pattern to ensure comprehensive marker detection.
     - Combines position and yaw information from all scanning angles
     - Returns both marker positions and their yaw adjustments.
   - get_all_ar_marker_ids(timeout_sec=5.0):
     - Listens to ROS topic /ar_pose_marker for AR marker detections.
     - Collects unique marker IDs over a specified timeout period.
     - Filters out marker ID 4 (used as center reference).
     - Uses a set to prevent duplicate detections.
     - Returns a list of detected marker IDs
   - get_yaws_and_adjustments(limb, marker_ids, yaws):
     - Calculates necessary yaw adjustments to make all blocks aligned.
     - Uses quaternion to Euler angle conversion for marker orientation.

    - Compares marker yaw with current end-effector orientation.
    - Calculates yaw difference needed for aligned block placements.
    - Maintains a record of previously calculated yaws.
  - lookup_tag(tag_number, AR_tag_pos)
    - Performs coordinate frame transformation from camera to robot base frame.
    - Uses tf2_ros for reliable transformations.
    - Handles special case for center reference marker (ID 4).
    - Normalizes z-height for regular blocks to ensure consistent placement.
    - Maintains global dictionary of marker positions.

   The perception system creates a comprehensive understanding of block positions, orientations, and spatial relationships, which are then used for planning pick and place operations, determining correct approach angles, ensuring proper block orientation during placement, and maintaining stack stability through precise positioning.

2. Planning phase:
   (a) Trajectory generation (get_trajector):
      - Gets current end-effector position via tf transforms.
      - Creates a linear trajectory as described in the design document between current and target positions
      - Converts RobotTrajectory using MotionPath
      - Applies calculated yaw adjustments to final joing configuration.
      - Returns a complete robot trajectory with velocity profiles.

   (b) Path Planning (PathPlanner class):
      - Uses MoveIt for collision free path planning
      - Plans movements to trajectory start positions
      - Retimes trajectories for smoother execution

3. Execution Phase:
   (a) For Pyramid Configuration:
      - Center placement (place_at_center):
        - Grabs block using grab_block
        - Uses tuck and pick_up for safe intermediate positions
        - Places block using place_block
      - Right and left placement (place_at_right, place_at_left):
      - Offset by block height in y-direction
      - Then adjust the orientation as described earlier

        - Repeat this procedure in a similar manner for the other 2 levels.

    (b) (b) For Vertical Configuration:

        - vertical function iterates through blocks.
        - Increments z-height by block height in each iteration
        - Maintains constant x, y position

4. Control System:

    - PID joint velocity controller is implemented as described in the design document also there is an option to use the Open Loop Controller.

This implementation provides robust trajectory generation and execution, multiple control strategies, safe intermediate positions, error handling and recovery, and precise positioning for stable stacking.

Execution Flow

1. System Initialization:

    - Start ROS node and system setup
    - Execute tuck position
    - Calibrate gripper
    - Initialize IK solver, Kinematics, Limb interface

2. Perception Flow:

    - ROS Topics Input:
        - ar_pose_marker: Provides marker poses
        - /tf: Provides coordinate transformations
    - Marker Detection Process:
        - scan() executes 5-point scanning pattern
        - get_all_ar_marker_ids() identifies visible markers
        - lookup_tag() transforms positions to base frame
        - get_yaws_and_adjustements() calculates orientation corrections.

3. Planning Flow:

    - get_trajectory
        - Receives current position from tf
        - Createss LinearTrajectory object
        - Sets target position and timing
    - Motion Path Generation
        - Converts LinearTrajectory to RobotTrajectory

    - Applies yaw corrections
    - Validates through MoveIt

4. Controller Flow:
    - Controller selection, either PID, Open Loop, or MoveIt controller.
    - Control Execution: Reads joint states from /robot/joint_states, generates velocity commands, provides feedback for position correction.

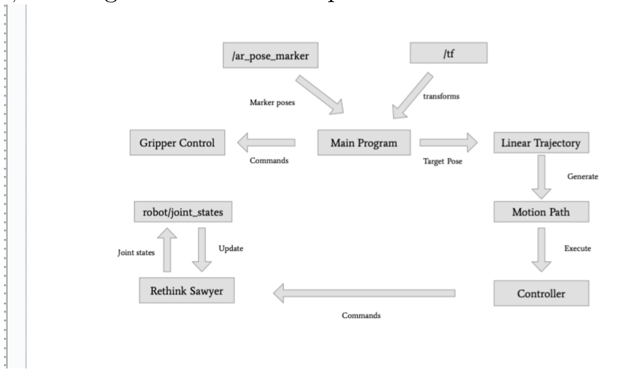
    Exection Flow:

    - Command processing: Converts trajectories to joint commands, sends commands to Rethink Sawyer, updates joint states.
    - Gripper Control:
        - Synchronizes with movement completion.
        - Executes grab_block() and place_block().

5. Stacking Execution Flow:
    - Pyramid Mode:
        - Level 1: place_at_center() → place_at_right() → place_at_left.
        - Level 2: place_skew_right → place_skew_left
        - Level 3: final_stack
    - Vertical Mode:
        - Sequential Stacking with height increments
        - Maintains constant x, y position

6. Feedback Loop:
    - Joint state Monitoring: continuous update from /robot/joint_states, position error calculation, velocity command adjustment

This flow chart represents the complete system architecture and execution flow, showing how different components interact with each other and input data.


/ar_pose_marker
/tf
Marker poses
transforms
Gripper Control
Commands
Main Program
Target Pose
Linear Trajectory
Generate
robot/joint_states
Motion Path
Joint states
Update
Execute
Rethink Sawyer
Commands
Controller