Autonomous Air Hockey System

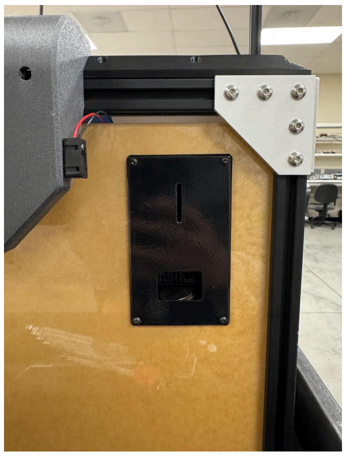
Team Members: Anthony Angeles, Zen Yamao

Project Description:

Our project will be an adaptation of an air hockey table with the play area being 35cm x 70cm, which will aim to have a player versus autonomous system, but will be based on a player versus player system to ensure completion. The idea for the base system is to have a camera sensor that will detect when a goal is scored throughout the game and an IR sensor to detect coin usage. This will both update the LCD screen and, given enough development time, actuate a system that dispenses another puck; However, the larger scoped version of this system is to include a sub-system that actuates a PP arm to simulate an opponent to allow for single player games. A camera will be used to detect positioning of the end-effector and puck to allow the correct commands to be given to move the end effector in the correct position or to transition between states.

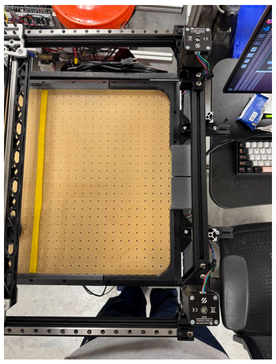
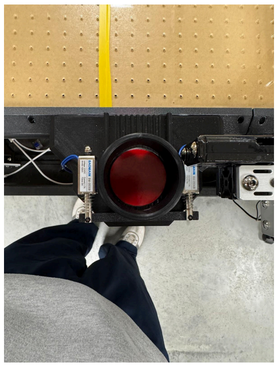
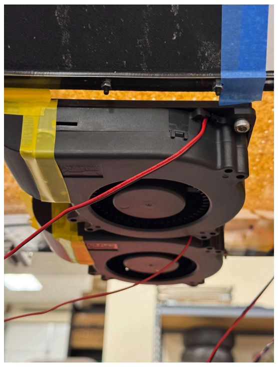
Sensors:

IR sensor (Coin/Goal), ZED 2i Camera

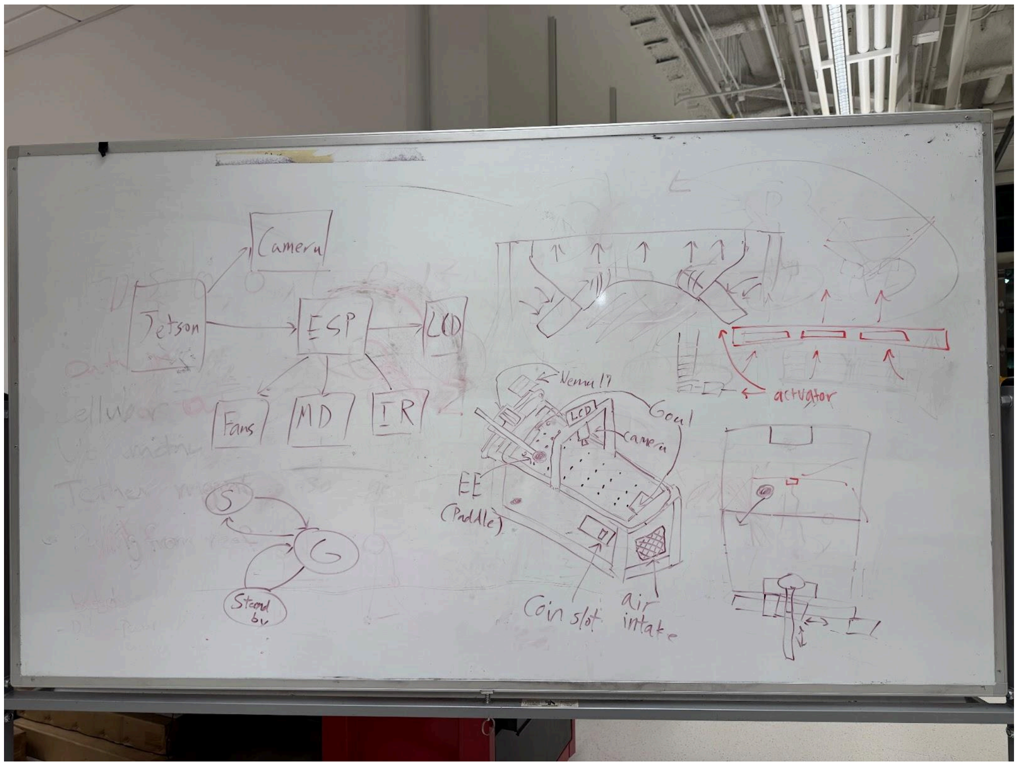


Actuators:

24V Blower fans, Nema 17 Steppers, Solenoid, E-Stop

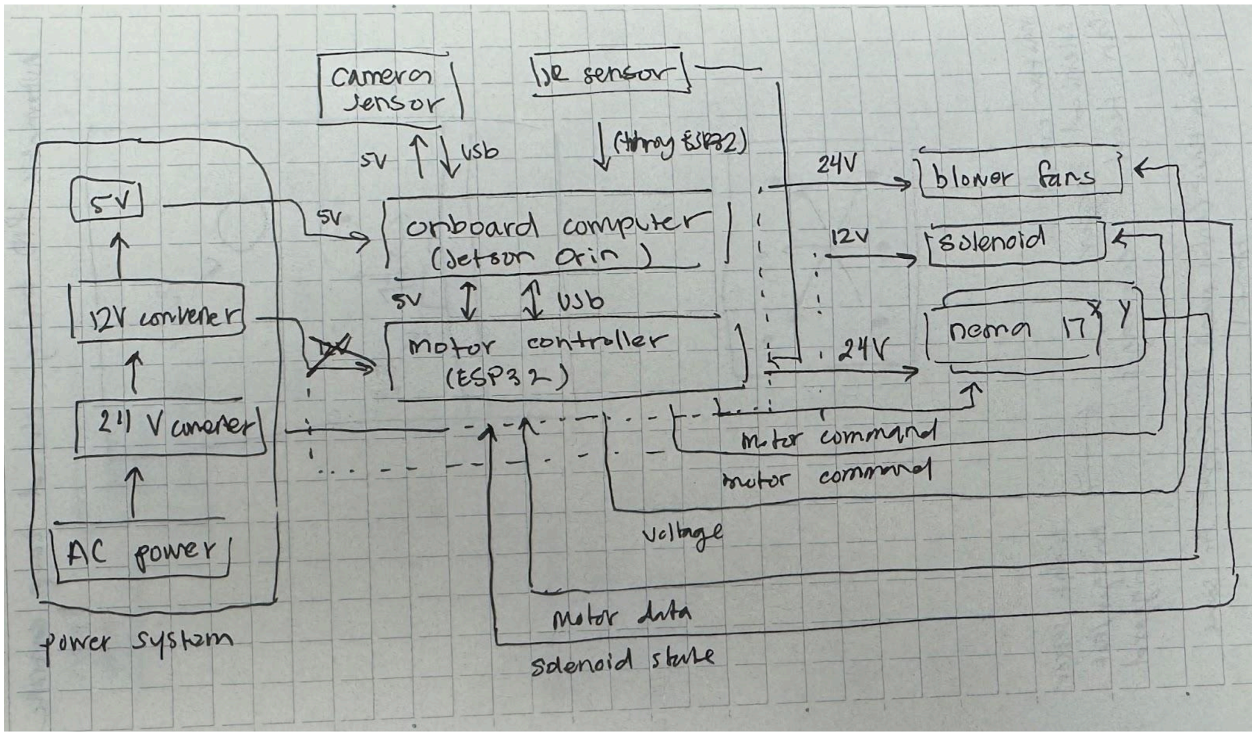




Configuration Sketch:

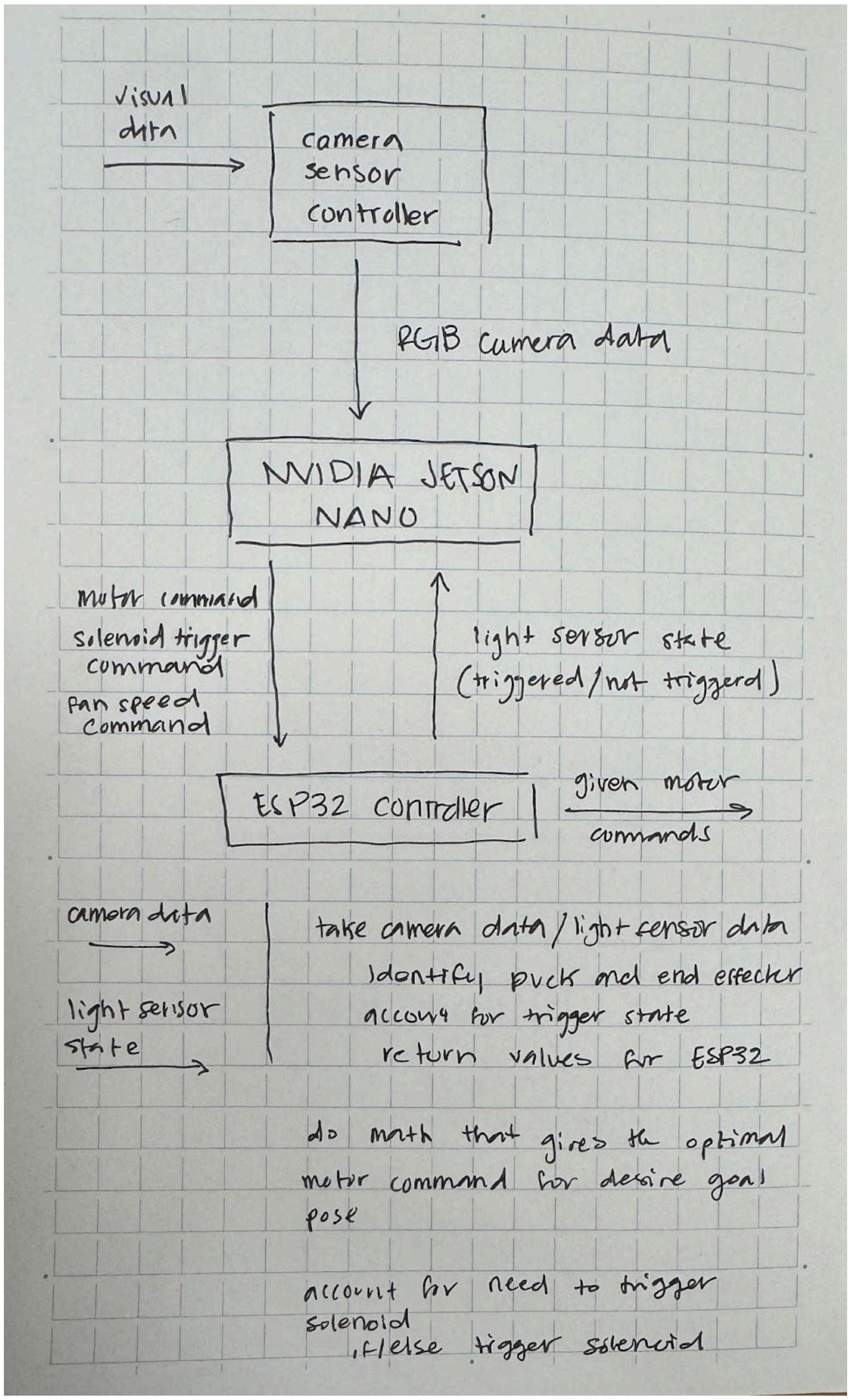


Component Block Diagram:

There is a small correction seen in the component block diagram where there is 12V being fed to the motor controller this has been crossed out and corrected, the 12V and 24V being fed to actuators can be denoted by the - - - lines crossing across the page and into the respective actuators



Software Functional Flowchart:



take camera data / light sensor data
identify puck and end effector
account for trigger state
return values for ESP32

do math that gives the optimal motor command for desire goal pose

account for need to trigger solenoid
if/else trigger solenoid

Software Implementation:

Jetson Orin Nano Code:

This Python application implements the high-level control system for the autonomous air hockey robot. It uses computer vision to track the puck and paddle positions, then applies PID control to command the gantry system to follow the puck in real-time.

System Architecture

The application uses a multithreaded design with three concurrent execution paths:

Thread	**Responsibility**
ZEDTracker Thread	Camera frame acquisition and object detection
Main Thread	PID control loop and display rendering
Motor Writer Thread	Serial communication with MCU at fixed rate

Computer Vision Pipeline

Hardware: ZED Stereo Camera at 720p, 30fps (depth disabled for speed)

Object Detection Algorithm:

1. Extract Region of Interest (ROI) from camera frame
2. Convert BGR to HSV color space
3. Apply color thresholding:
    a. Red puck: Dual-range mask (hue 0-10 and 160-180) to handle red wraparound
    b. Green paddle: Single-range mask (hue 35-85)
4. Morphological operations (open/close) to remove noise
5. Contour detection with circularity filtering (>0.5) to reject non-circular objects

6. Centroid calculation via image moments

PID Control System

Controller Structure: Two independent single-axis PID controllers (X and Y)

Control Law: velocity = Kp * error + Ki * integral(error) + Kd * derivative(error)

Parameters:

- Proportional gain (Kp): 10.0
- Integral gain (Ki): 0.01
- Derivative gain (Kd): 0.5
- Integral windup limit: 200.0

Coordinate Mapping: Camera pixel coordinates are mapped to motor axes with configurable axis swapping and sign inversion to account for physical mounting orientation.

Serial Communication Protocol

Connection: USB serial at 115200 baudStartup Sequence:

1. Open serial port
2. Send HOME command
3. Block until HOMEXY COMPLETE received (up to 120s timeout)
4. Start asynchronous velocity streaming

Runtime Operation:

1. Writer thread sends velocity commands at configurable rate (default 0.5s)
2. Only transmits when velocity changes (avoids redundant traffic)
3. Format: vx vy\n where vx/vy are floats in mm/s

Shutdown Sequence:

1. Send synchronous stop command (0.00 0.00)
2. Terminate writer thread
3. Close serial connection

Safety Features

- Pixel deadband (5 pixels) prevents jitter from sensor noise
- Velocity clamping to MAX_VELOCITY (10000 mm/s)
- Minimum velocity threshold (40 mm/s) to overcome static friction
- PID reset when target lost (prevents integral windup)
- Graceful shutdown on Ctrl+C with guaranteed stop command

Real-Time Performance

- Vision processing: ~30Hz (camera-limited)
- Control loop: ~500Hz (2ms sleep)
- Command transmission: 2Hz (configurable)
- End-to-end latency: Approximately 50-100ms from puck movement to paddle response

RP2350 - NEMA 17 Motor Controller:

This firmware controls the CoreXY gantry system that positions the air hockey paddle. It runs on a Seeed XIAO RP2350 microcontroller and utilizes both CPU cores for responsive real-time motion control.

Hardware Interface

- Microcontroller: RP2350 dual-core ARM Cortex-M33 at 150MHz
- Motor Drivers: Two TMC2209 stepper drivers communicating via UART
- Motors A & B: NEMA stepper motors in CoreXY kinematic configuration
- DIAG Pins: Stall detection outputs from TMC2209 for sensorless homing

Dual-Core Architecture

Core 0 - Communication Handler:

- Processes all serial commands without blocking
- Parses velocity commands (vx vy\n format in mm/s)
- Handles homing commands (HOME, HOMEX, HOMEY)
- Implements 2-second serial timeout safety stop

- Uses mutex-protected shared variables for thread-safe data exchange

Core 1 - Motion Executor:

- Executes motor stepping independently from serial processing
- Implements trapezoidal velocity ramping with configurable acceleration
- Checks for abort flags every 10 steps (~0.5ms) allowing rapid response to new commands
- Caps step execution to 1000 steps per update (~50ms) to maintain responsiveness

Key Algorithms

CoreXY Kinematics:

1. Transforms Cartesian (X,Y) motion into motor A/B step commands
2. Motor A = X + Y, Motor B = X - Y
3. Uses Bresenham-style interpolation for coordinated multi-axis motion

Sensorless Homing:

1. Moves slowly toward wall until TMC2209 detects motor stall
2. Backs off 10mm from wall
3. Moves to geometric center of workspace
4. Establishes coordinate system origin

Velocity Control Loop (50Hz):

1. Receives target velocity from Core 0
2. Applies acceleration ramping to smooth motion
3. Enforces software workspace bounds (300mm x 280mm)
4. Converts velocity to step pulses
5. Updates position estimate based on actual steps executed

Communication Protocol

Command	**Function**

vx vy	Set X/Y velocity in mm/s
HOME	Home both axes to center
HOMEX	Home X axis only
HOMEY	Home Y axis only

Safety Features

- Workspace bounding box prevents over-travel
- Serial timeout stops motors if communication lost for 2 seconds
- Sensorless stall detection prevents crashes during homing
- Maximum step limit per update prevents indefinite blocking

Design Rationale

The dual-core architecture solves a fundamental real-time control challenge: stepper motor pulse generation requires precise timing that blocks code execution, but serial command reception must remain responsive for smooth puck tracking. By dedicating Core 0 to communication and Core 1 to motion, the system achieves sub-millisecond response to new velocity commands even during active motor movement.

RP2350 - Sensor and Actuator Controller:

This firmware controls the auxiliary systems of an air hockey table: the air blower fan, puck dispensing solenoid, and goal detection sensors. It runs on a secondary microcontroller separate from the gantry motion controller.

Hardware Interface

- Fan (D6): PWM-controlled air blower that creates the air cushion for puck flotation

- Solenoid (D4): Puck dispenser mechanism triggered by a 100ms pulse
- IR Sensor D7: Coin/token insertion detector (triggers game start)
- IR Sensor D8: Goal 1 detection (beam-break sensor in goal opening)
- IR Sensor D9: Goal 2 detection (beam-break sensor in goal opening)

Functional Behavior

Game Start Sequence (D7 trigger):

1. Detects coin insertion via IR beam break
2. Activates the air blower fan at full speed (PWM 255)
3. Fires the solenoid for 100ms to dispense a puck onto the playing surface
4. Locks the trigger to prevent multiple dispenses until beam is restored

Goal Detection (D8/D9 triggers):

1. Monitors both goal sensors for puck entry
2. On goal detection, sends "STOP" and "RESET" commands over serial
3. These commands are intended for the main gantry controller to halt and re-home the paddle

Serial Command Interface:

1. fan <0-255>: Set fan speed (0=off, 255=full)
2. fan on / fan off: Convenience commands
3. pulse: Manually trigger solenoid for testing

Design Considerations

- Uses edge detection (HIGH-to-LOW transitions) to prevent repeated triggers
- Includes debounce logic via state tracking variables
- Sensors use INPUT_PULLUP configuration (active-low when beam is broken)
- 50ms polling interval balances responsiveness with CPU efficiency