ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Interactive Projection System on Arbitrary Surfaces

Team #29

YUQI TANG
(yuqi9@illinois.edu)
ZIBO DAI
(zibodai2@illinois.edu)
JING WENG
(jingw15@illinois.edu)
JIE XU
(jie9@illinois.edu)

TA: Zhefan Lin

May 15, 2026

Abstract

In this project, we present a vision-based interactive projection system that extends a conventional projected display into a gesture-driven interaction interface. The system integrates a projector, an RGB camera, ArUco-based calibration, homography-based coordinate mapping, MediaPipe hand tracking, pinch gesture recognition, and a motorized platform for projector pre-alignment. In our workflow, the user's index fingertip is mapped to a cursor position on the projected interface, and a thumb-index pinch gesture is interpreted as a click event.

The completed prototype establishes the geometric relationship between the camera view and the projector output using four-corner ArUco marker calibration. During interaction, the system tracks hand landmarks in real time, maps the index fingertip into projector coordinates through a homography transformation, and applies cursor smoothing to improve visual stability. The pinch-based click mechanism enables contact-free interaction with projected applications, including a bubble-popping demonstration and simple gesture-controlled mini-games. In addition, a phone-controlled motorized platform allows the projector yaw and pitch angles to be manually adjusted before interaction begins. The final system demonstrates that visual perception, geometric calibration, projected feedback, and basic electromechanical control can be integrated into a practical prototype for interactive projection on physical surfaces.

Contents

1 Introduction

1.1 Problem Statement

Conventional human-computer interaction devices are typically built around fixed-size displays, such as smartphones, tablets, monitors, and self-service terminals. Although mature and reliable, their interaction area is constrained by screen size, and most require direct contact with a shared surface. Projection technology provides an alternative by extending graphical interfaces onto larger surfaces. Prior work such as OmniTouch has shown that projection with depth sensing can support multitouch-like interaction on everyday surfaces [1].

However, a conventional projector only provides one-way visual output and cannot sense user motion or map hand position to the projected interface. This project therefore integrates projection display, visual perception, spatial calibration, and gesture recognition into a functional interactive system. It maps the index fingertip to a projected cursor and uses a pinch gesture to trigger visual feedback.

1.2 Solution Overview

The final design is a vision-based interactive projection system. A projector displays the interface onto a target surface, while an RGB/depth camera captures the projection region and the user's hand. The PC-side software performs ArUco marker detection, homography estimation, MediaPipe hand tracking, coordinate mapping, and pinch gesture recognition. ArUco markers establish the camera-to-projector correspondence, a fiducial-marker method commonly used in camera pose estimation and augmented-reality applications [2].

During interaction, MediaPipe detects hand landmarks in real time from RGB input [3]. The index fingertip is used as the cursor anchor and mapped to projector coordinates through a homography transformation, which represents the projective relationship between two views of a planar surface [4]. The click event is inferred from the normalized thumb-index distance rather than physical surface contact. The projector is mounted on a motorized yaw-pitch platform that can be manually pre-aligned through a mobile phone.

1.3 High-Level Requirements

The high-level requirements are defined around the final gesture-based projection prototype. The system shall project a stable and clearly visible interactive interface onto at least one physical surface, establish camera-to-projector mapping through ArUco calibration, and convert the detected fingertip position into a projected cursor. It shall also provide stable index-fingertip cursor control with smoothing to reduce visible jitter.

The system shall recognize a thumb-index pinch gesture and convert it into a click state that triggers visible projected feedback, such as popping a bubble or activating a mini-

game object. It shall also support initial yaw and pitch adjustment of the motorized projector platform through mobile-phone control. This platform control is used for setup and pre-alignment, not real-time gesture-controlled projector rotation.

1.4 System Overview

As shown in Figure 1, the system consists of five main subsystems: projection, vision and calibration, gesture interaction, motorized platform control, and power/control integration. The projection subsystem displays calibration patterns and interactive content. The vision and calibration subsystem uses the RGB/depth camera and four-corner ArUco markers to compute the camera-to-projector homography. The gesture interaction subsystem tracks the user's hand with MediaPipe, uses the index fingertip as the cursor anchor, and generates a click state through pinch recognition.

The motorized platform control subsystem provides projector pre-alignment through wireless mobile-phone commands for yaw and pitch adjustment. The power and control integration subsystem connects the controller, motor driver, DC motors, switches, and power circuitry for stable hardware operation. Overall, the camera provides visual input, the PC performs calibration, hand tracking, coordinate mapping, and pinch recognition, and the system outputs projected visual feedback with setup-stage motor control.

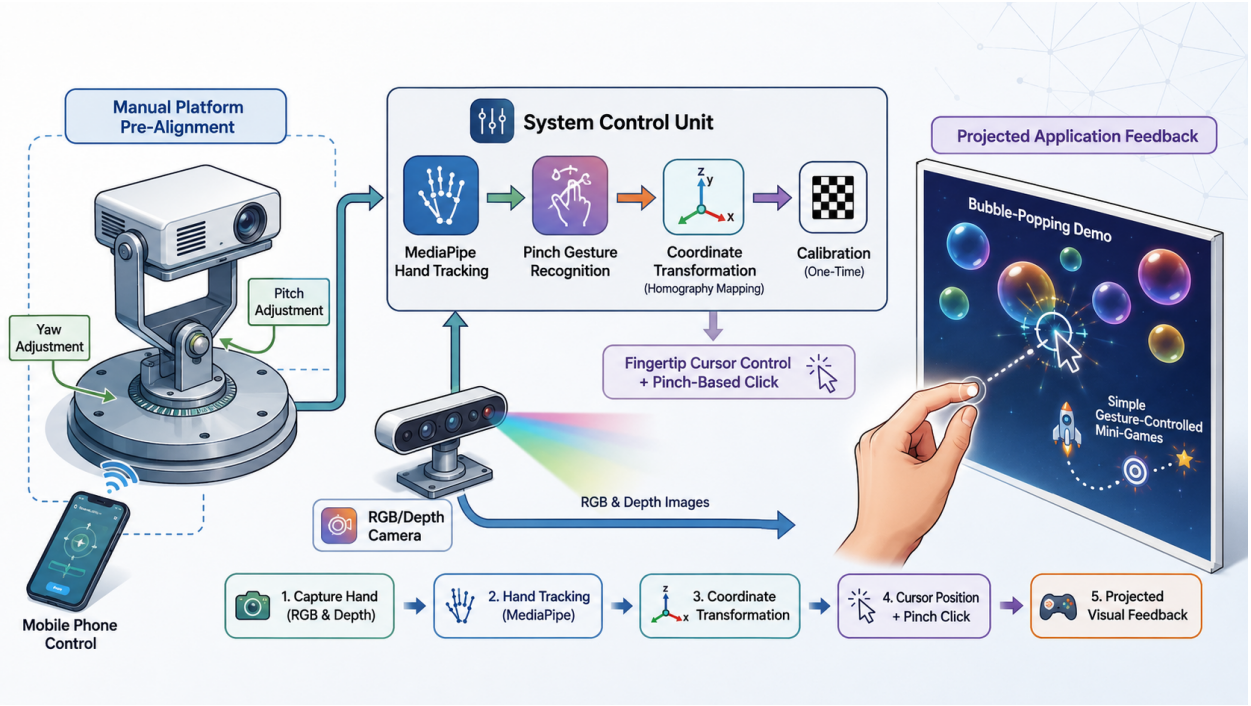

Figure 1: System overview of the gesture-based interactive projection platform.

2 Design

2.1 Block Diagram

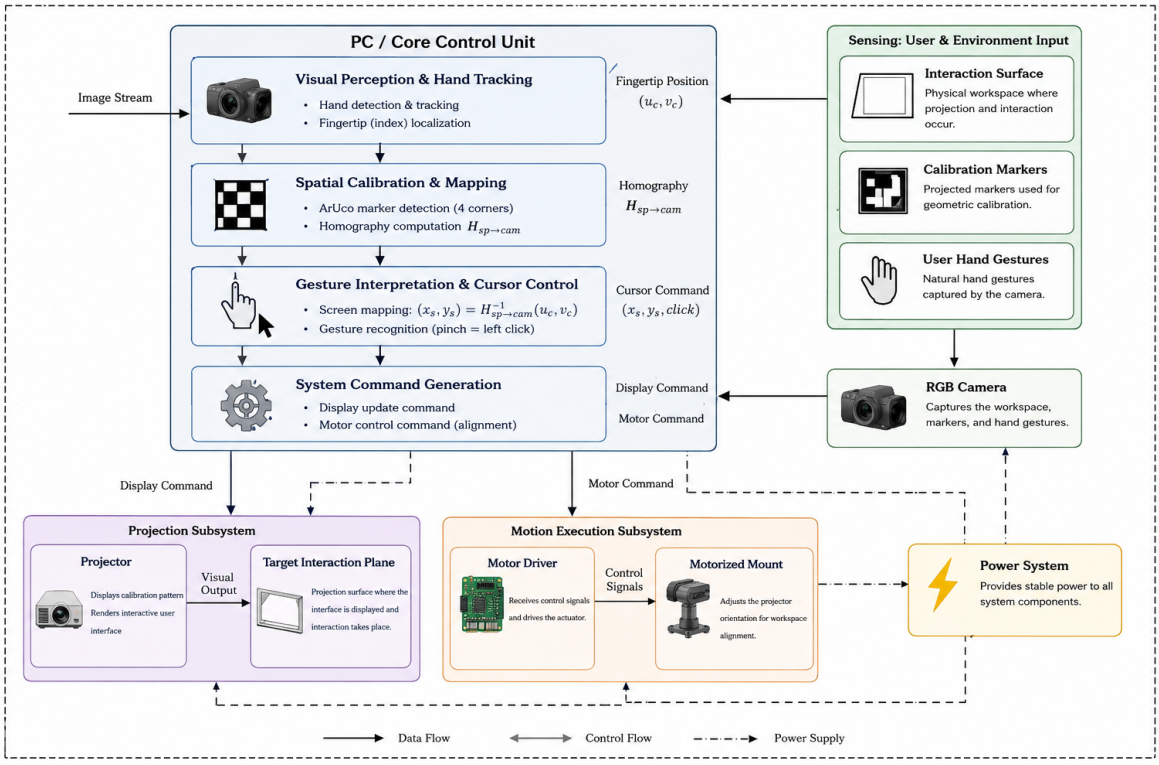

Figure 2: System Block Diagram

The system's overall architecture is shown in Figure 2. It presents the sensing, calibration, interaction, and actuation pipeline of the gesture-controlled projection system. Starting from image acquisition, the system establishes the projector-camera relationship, tracks the user's hand, converts fingertip motion into cursor motion, and outputs display and motor-control commands. The system consists of the following primary modules:

- **User & Environment Input**: The RGB camera observes the interaction surface, projected calibration markers, and the user's hand gestures. The visual stream is the primary sensing source for calibration and gesture interaction.
- **PC/Core Control Unit**: This module performs visual perception, hand tracking, ArUco calibration, homography mapping, gesture interpretation, cursor control, and motor command generation. It detects the corner markers, maps camera coordinates to screen coordinates, tracks the index fingertip, and recognizes thumb–index pinch as a left-click event.
- **Projection & Motion Execution**: The projector displays the calibration pattern and interactive interface. The motor driver adjusts the mounting mechanism so the projected interface can be aligned with the desired interaction area.

- **Power System**: The power subsystem supplies stable electrical power to the camera, control unit, motor driver, projector, and auxiliary switching circuitry.

At the system level, the camera feeds image data to the PC, where calibration and gesture recognition are performed. The resulting commands follow two output paths: one updates the projected visual interface, and the other drives the motorized mechanism for projector alignment. This architecture integrates spatial registration, gesture-based interaction, and physical projection adjustment into one interactive platform.

2.2 Gesture Recognition and Interaction System

The system employs a computer vision solution based on a RGB camera to achieve non-contact interaction. The core of this solution lies in identifying the relative motion logic between fingers to simulate physical touch and click events.

2.2.1 Hand Landmark Detection and Feature Extraction

The system first preprocesses the input RGB video frames and uses `MediaPipe` to extract hand landmarks in real time.

- **Core Coordinate Acquisition:** The system real-time tracks the pixel coordinates of the Index Finger Tip and the Thumb Tip.
- **Interaction Center Definition:** The index finger is designated as the current position $P(x, y)$ of the "virtual mouse." These coordinates are subsequently mapped to the projection space to drive the cursor movement.

2.2.2 State Estimation Logic Based on Pinch Distance

The system uses the Euclidean distance d_{pinch} between the thumb tip and index fingertip as the criterion for determining interaction intent.

- **Pinch Threshold Logic:**
  - **Hover Tracking (Hover):** When the detected distance $d_{pinch} >$ threshold, the hand is identified in an extended state. The system executes only the `mouse.move()` command, and the projection interface displays a **green hollow circle**.
  - **Press and Drag:** When $d_{pinch} \leq$ threshold, a "pinch" gesture is triggered. The system immediately issues a `mouse.press()` command. If the hand moves while in this state, it generates a continuous dragging effect, and the visual feedback synchronizes by switching to a **red solid circle**.

2.2.3 Interaction Flow and Visual Feedback

During the `INTERACTING` state, the system provides visual overlays and event processing based on the detected interaction state, as summarized in Table 1. The visual feedback

helps the user understand whether the system is only tracking the cursor or has detected an active pinch-click command.

Table 1: Interaction States, Driver Behaviors, and Visual Feedback

Interaction State	**Driver Behavior**	**Visual Feedback (OpenCV Rendering)**
No Valid Gesture	Releases click state	Keeps original frame unchanged.
Hover Tracking	Updates cursor position	Draws a **green hollow circle** ($r = 8$) at the cursor location; displays the mapped projector coordinate.
Pinch Click	Triggers click state	Draws a **red solid circle** ($r = 14$) at the cursor location; displays: `PINCH -> projector(X, Y).`

2.2.4 System Robustness Optimizations

- **First-Order Low-Pass Trajectory Filtering:** To reduce cursor jitter caused by frame-to-frame landmark variation, an exponential smoothing factor ($\alpha = 0.2$) is applied to coordinate transitions. The target position integrates historical coordinates to ensure smooth and stable cursor movement.
- **Gesture-State Debouncing:** To avoid repeated click triggers from small hand-motion fluctuations, the system uses a boolean state lock. A click is issued only when the pinch state first becomes active and is released when the fingers separate.
- **Dynamic Environment Adaptation:** Critical parameters, including the pinch threshold and frame stability count, are exposed via `argparse`. This enables rapid adjustment under different lighting conditions, user distances, and projection surfaces.

2.3 Alignment and Projection Subsystem

This subsystem provides the mechanical alignment foundation for the interactive projection system. It combines a motorized projection platform, camera-based spatial calibration support, and gesture-based interaction output to transform an ordinary projection surface into an interactive interface. Instead of relying on physical touch sensing on the wall or desk, the final design uses vision-based pointing and pinch gestures: the user's index fingertip controls the cursor position, and a thumb-index pinch is interpreted as a left-click command.

2.3.1 Motorized Projection Orientation Mechanism

As shown in Fig. 3, the projector is mounted is mounted on a compact motorized mechanism composed of a base motor for horizontal yaw adjustment and a pitch adjustment mechanism for vertical aiming. This arrangement provides the projector with the rotational degrees of freedom required to steer the projected interface toward the desired interaction area while keeping the projected image inside the RGB camera's field of view.

In the final prototype, the motorized platform is used for coarse pre-alignment before interaction begins. The platform is not controlled by hand gestures during interaction. Instead, the user sends wireless commands from a mobile phone to adjust the yaw and pitch angles, so that the projection region, the camera observation region, and the user's gesture workspace overlap as much as possible. This mechanical stage establishes a stable geometric setup for the subsequent calibration and gesture interaction processes.

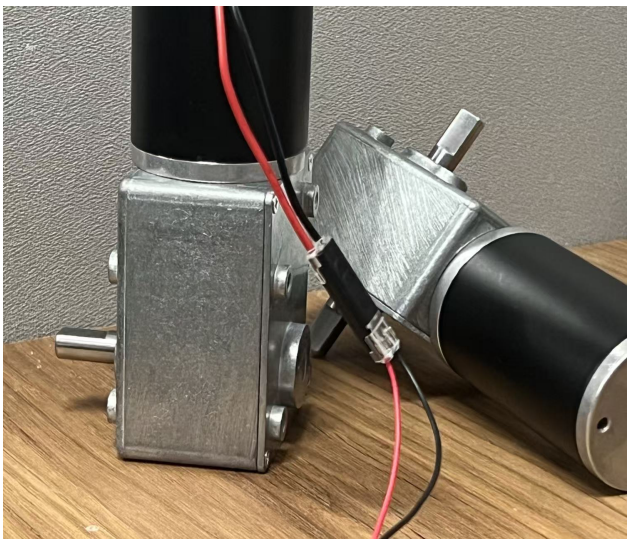

Figure 3: Motorized orientation mechanism for projector pre-alignment.

2.3.2 Four-Corner ArUco Calibration Support

As shown in Fig. 4, at system startup, four ArUco markers are displayed at the four corners of the projected calibration pattern. The RGB camera captures these markers and detects their corner coordinates in image space. By matching the detected camera-space corner positions with the known screen-space coordinates of the projected markers, the system computes a planar homography matrix that maps points observed by the camera to the computer screen and projector coordinate system.

Figure 4: Four-corner ArUco calibration pattern.

This calibration procedure creates a unified coordinate relationship between the physical projection surface and the digital desktop. After the four markers are detected stably for several consecutive frames, the homography matrix is solved automatically and the system exits calibration mode. As a result, any fingertip location identified in the camera image can be transformed into a corresponding cursor position on the projected interface. Compared with the original proposal, this final implementation no longer depends on full 3D surface reconstruction or robotic-arm inverse kinematics; instead, it adopts a lighter 2D geometric mapping approach that is sufficient for interactive pointing on a planar projection surface.

2.3.3 Design Objectives and Evaluation

Table 2 demonstrates our design objectives and evaluation.

Table 2: Design objectives and evaluation methods for the projection alignment and gesture interaction subsystem.

Design Objective	**Evaluation Method**
1. Establish a reliable four-corner ArUco calibration process for automatic screen-to-camera registration.	1. Project the calibration pattern, detect the four markers, and confirm that the homography matrix can be computed automatically.
2. Provide stable cursor control based on index-fingertip tracking and homography mapping.	2. Move the index finger across known projected locations and compare the intended positions with the cursor movement.
3. Support contact-free left-click interaction through a thumb–index pinch gesture.	3. Select projected targets using fingertip pointing and pinch gestures, and evaluate click responsiveness and usability.
4. Support coarse projector pre-alignment through the motorized platform.	4. Use mobile-phone commands to adjust yaw and pitch and confirm alignment between the projection, camera view, and interaction area.

2.4 Calibration and Mapping Subsystem

This subsystem serves as the coordinate bridge among the projector, depth camera, and rotating platform. It converts camera observations of the user's hand into cursor position and click-state signals on the projected interface. The RGB image stream is processed through ArUco marker detection, homography estimation, hand landmark tracking, coordinate mapping, and pinch gesture recognition. The pipeline has two phases: calibration, which establishes the camera-to-projector mapping, and interaction, which tracks the hand in real time and converts fingertip motion and pinch gestures into commands for applications such as bubble popping and simple gesture-controlled games. Figure 5 shows our overall pipiline.

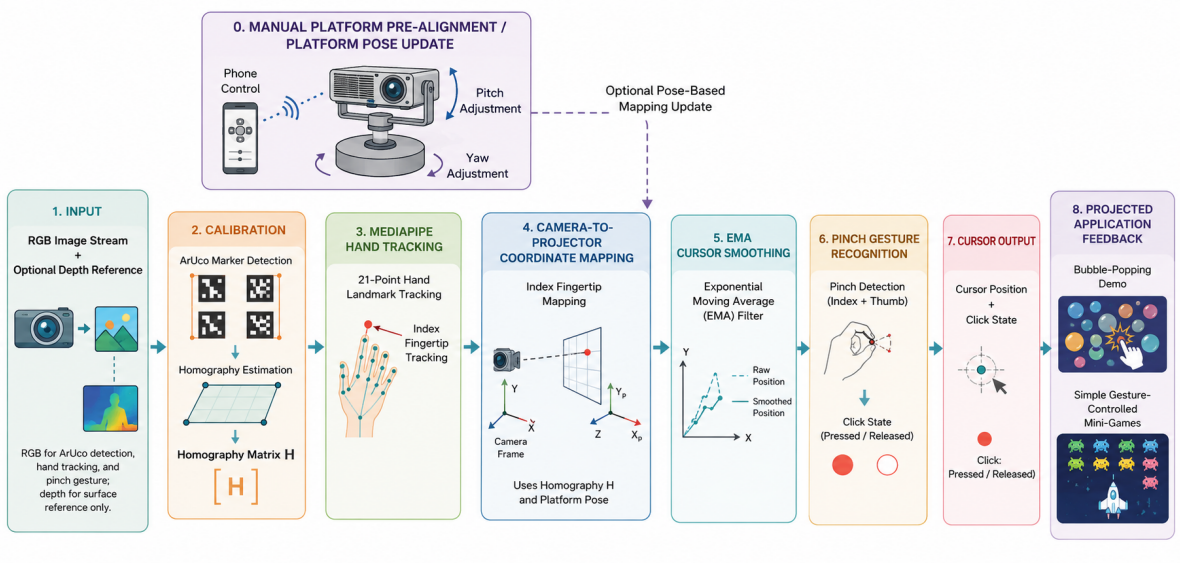

Figure 5: Overall Pipeline of Calibration and mapping subsystem.

2.4.1 Projector-Camera Homography Calibration

Because the projector and depth camera view the same surface from different positions and angles, their pixel coordinate systems are not directly aligned. The system therefore computes a geometric mapping from camera-image coordinates to projector-frame coordinates. Figure 6 shows our coordinate mapping structure, while Figure 7 demenstrates the depth camera and projector used in our work.

The calibration uses four ArUco fiducial markers. At startup, the projector displays markers with IDs 0 through 3 near the four corners of the projection region, while the depth camera captures the pattern. OpenCV's `ArUcoDetector` detects the markers and extracts their corner locations. The center of each marker is used as a camera-space feature point and paired with its known projector-space reference coordinate.

From these four correspondences, the system computes a 3×3 homography matrix H using OpenCV's `findHomography` function:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{1}$$

Here, (x, y) is a camera-image pixel coordinate, and $(u'/w', v'/w')$ is the corresponding normalized projector-frame coordinate. RANSAC with a 5.0-pixel reprojection threshold is used to reduce the effect of marker-detection noise and outliers.

After calibration, camera-space points are mapped to projector-space coordinates using `cv2.perspectiveTransform`. Accuracy was evaluated with 48 synthetic test points sampled on an 8×6 grid inside the calibrated region. The maximum mapping error was below 1.0 pixel, satisfying the ± 5 pixel design requirement.

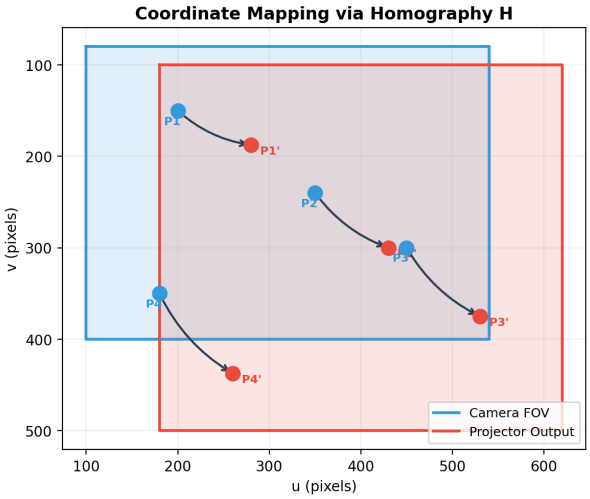

Figure 6: Camera-to-projector coordinate mapping using homography.

Figure 7: Projector and depth camera used in the our framework.

2.4.2 Platform Pose Update

When the projector and camera rotate with the platform, the camera-to-projector mapping changes. Instead of recomputing the full homography after each small motion, the system applies a geometry-based update.

For a yaw change $\Delta\theta$ and pitch change $\Delta\phi$, the angular motion is converted into image-plane offsets. With image width W_{img}, image height H_{img}, and camera vertical field of view FOV_v, the offsets are approximated as

$$dx = W_{\text{img}} \frac{\tan(\Delta\theta)}{2\tan(\text{FOV}_v/2)}, \qquad dy = H_{\text{img}} \frac{\tan(\Delta\phi)}{2\tan(\text{FOV}_v/2)}. \tag{2}$$

The updated homography is then computed by applying a translational correction:

$$H_{\text{new}} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} H_{\text{old}}. \tag{3}$$

This update requires only one matrix multiplication and is suitable for real-time use. Since it assumes small rotations mainly cause global image-plane shifts, larger rotations, especially above about 10°, may introduce mapping error. In such cases, the user can press the C key to trigger full recalibration.

During calibration, the system stores the median valid surface depth as Z_{surface}, which can help estimate whether the finger is approaching the projection surface.

2.4.3 Hand Tracking and Coordinate Mapping

During interaction, the system tracks the user's hand in each RGB frame and maps the index fingertip to the projected interface. MediaPipe HandLandmarker detects 21 hand landmarks, and landmark #8 is used as the cursor anchor because it corresponds to the index fingertip and matches natural pointing behavior.

The normalized fingertip coordinate (x_n, y_n) is converted to camera pixel coordinates as

$$(u, v) = \left(\lfloor x_n W_{\text{frame}} \rfloor, \lfloor y_n H_{\text{frame}} \rfloor\right), \tag{4}$$

where W_{frame} and H_{frame} denote the camera frame dimensions. The fingertip is then mapped to projector coordinates using the homography matrix H.

To reduce cursor jitter, the system applies an exponential moving average filter:

$$p_t = \alpha p_t^{\text{raw}} + (1 - \alpha) p_{t-1}, \tag{5}$$

where p_t^{raw} is the current mapped coordinate, p_t is the smoothed coordinate, and $\alpha = 0.2$. This reduces frame-to-frame jitter while preserving real-time responsiveness.

2.4.4 Pinch Gesture Recognition

The system uses a pinch gesture for click interaction instead of depth-based contact detection, avoiding the need to determine physical fingertip contact with the projection

surface. The click state is inferred from the distance between the thumb tip and index fingertip.

After MediaPipe detects the hand landmarks, the system extracts landmark #4, the thumb tip, and landmark #8, the index fingertip. Their Euclidean distance is

$$d_{\text{pinch}} = \sqrt{(x_{\text{thumb}} - x_{\text{index}})^2 + (y_{\text{thumb}} - y_{\text{index}})^2}. \tag{6}$$

To reduce variation from hand size and camera distance, d_{pinch} is normalized by a hand-scale reference, defined as the distance between landmark #5 and landmark #17. If the normalized distance falls below a threshold, the system enters the click state; otherwise, the click is released.

2.5 Control and Integration Subsystem

2.5.1 PCB Design and Circuit Layout

Figure 8 shows the layout of our PCB design. The hardware part of the control and integration subsystem is centered on a custom-designed PCB, which integrates power management, circuit interfaces, and signal distribution. As in figure 9, the PCB employs an L298N motor driver module with an integrated H-bridge, enabling straightforward forward and reverse motor control. Its design simplifies integration with the Arduino interface and ensures reliable operation. When a high-level control signal enters a designated PCB port, the signal flows through the PCB traces to the L298N module, which drives the corresponding motor. Each control port corresponds to a specific motor channel, allowing independent control of each motor. Pulse-width modulation (PWM) signals are used to precisely adjust motor start, speed, and duration.

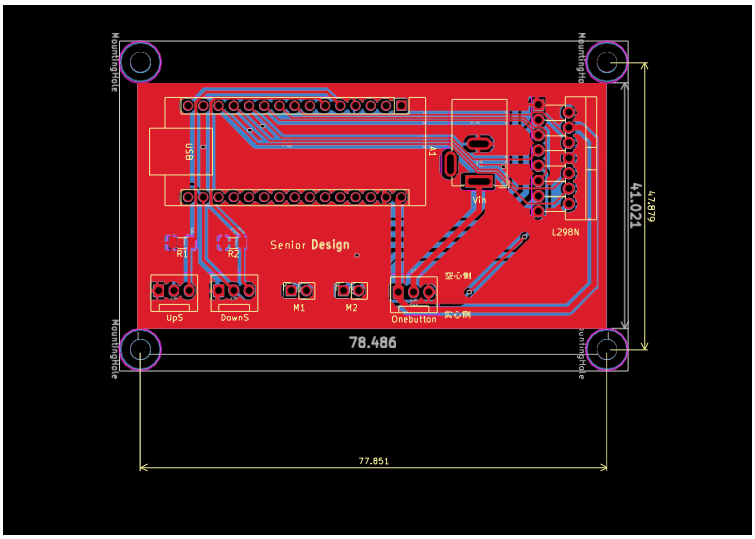

Figure 8: PCB Layout

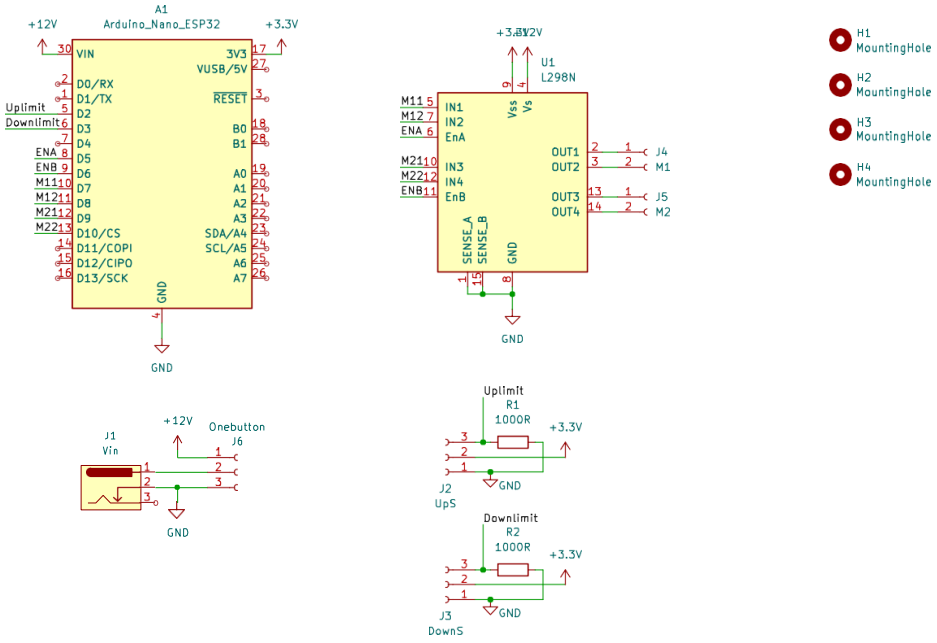

Figure 9: PCB Schematic

2.5.2 Motor Selection and Control Strategy

The motors used are worm-gear DC motors, chosen for stable speed control and the ability to rotate forward and backward. During operation, the motors follow a smooth acceleration–constant speed–deceleration–stop profile to ensure motion stability and reduce mechanical impact. This controlled speed profile is critical for accurate positioning and safe mechanical operation.

2.5.3 Buttons and Safety Switches

The system includes operational buttons and safety switches, located near the PCB interface. Pressing the operation button triggers the PCB control logic and starts the system. Safety switches prevent motor over-rotation; when triggered by mechanical contact with the supporting structure, the PCB immediately stops the corresponding motor, protecting both the motor and mechanical components.

2.5.4 Arduino Communication and Wireless Control

The system uses Arduino's built-in Bluetooth module to implement wireless control from a mobile phone. Users input numerical operation commands via the mobile application, which the Arduino receives and parses. Each command corresponds to a specific motor, including selection, rotation direction, and duration. Based on the parsed command, the Arduino generates PWM signals, which are sent through the L298N module to drive the

motors. Motor actions follow a smooth acceleration and deceleration strategy, ensuring safe and stable operation.

2.5.5 Control Logic and Operation Modes

Figure 10 shows the control logic. The overall control flow is as follows: operation commands are sent from the mobile phone, transmitted via Bluetooth to the Arduino, parsed into PWM signals, and delivered through the L298N module to the motors, which execute the corresponding actions. The system supports multiple operation modes, including single-step control, forward or reverse rotation, motor selection, and timed rotation. This setup ensures continuous and controllable operation from user input to mechanical response.

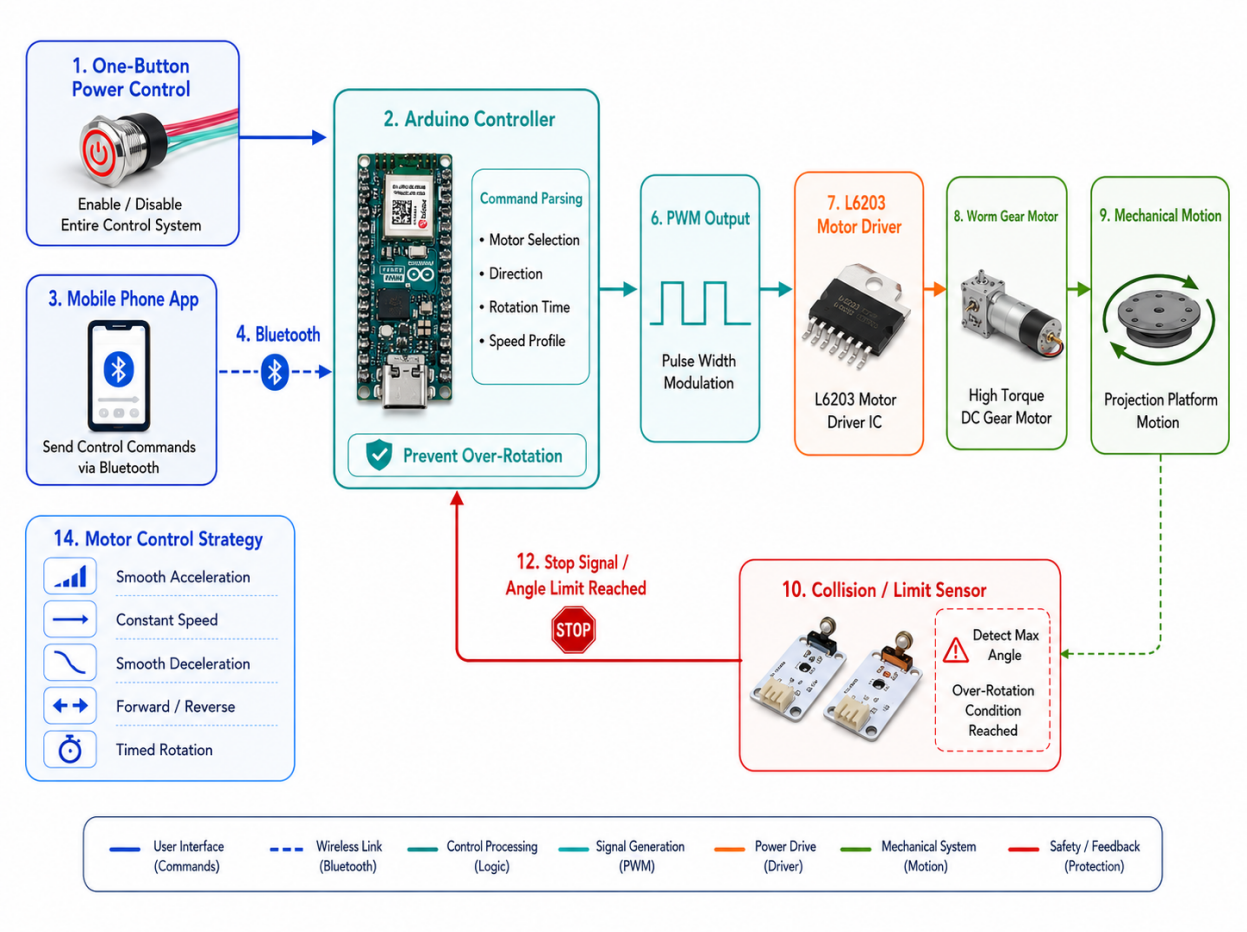

Figure 10: Control Flow Diagram

2.6 Design Alternatives

2.6.1 Touch Detection: Depth-Based vs. Gesture-Based

In developing a projection-based interactive interface, a key design decision is how to trigger "click" and "drag" events. This project evaluated two paradigms:

- **Depth-Based Proximity:** Traditional touch emulation systems use depth sensors to measure the distance between the user's finger and the projection surface. A click is triggered when the finger's Z-axis coordinate falls below a threshold relative to the surface plane. Although precise, this method requires specialized hardware, is sensitive to infrared interference, and depends on complex 3D calibration. Minor depth noise may also cause jitter or false triggers.
- **Gesture-Based Interaction:** The system ultimately adopted a gesture recognition model. Instead of measuring physical contact, it interprets specific hand configurations as click commands. The system monitors the Euclidean distance between the thumb tip and index fingertip; when the two fingers converge, it executes a "click." This reduces dependence on depth sensing and better distinguishes user intent from accidental surface approach.

By prioritizing portability and interaction reliability, gesture-based interaction was selected as the core interaction framework.

2.6.2 Calibration Marker: QR Code vs. ArUco Marker

Another design choice involved selecting the visual marker for camera-to-projector calibration. QR codes are effective for encoding information and identifying markers through data payloads. However, their detection process is mainly designed for information retrieval rather than precise geometric localization, making their corner accuracy and long-distance stability less suitable for projector-camera registration.

ArUco markers are designed for fiducial-based geometric calibration. Their binary grid structure allows OpenCV to detect marker corners efficiently and consistently, providing stable four-corner localization for homography estimation. Since this system requires accurate camera-to-projector correspondence rather than data decoding, ArUco markers were selected as the final calibration medium.

2.6.3 Hand Tracking Interface: MediaPipe Legacy API vs. Task API

For hand landmark detection, we compared the MediaPipe Legacy API and the newer MediaPipe Task API. The legacy interface, `mp.solutions.hands`, is simple to integrate but may introduce higher latency and occasional landmark instability.

The Task API, implemented through `mp.tasks.vision.HandLandmarker`, provides video-processing mode with better temporal consistency. Since the final system requires stable cursor control and pinch recognition, the MediaPipe Task API was adopted.

3 Verification

3.1 Completion of Requirements

Table 3: Verification of functional requirements for the AirTouch interactive projection system.

Requirement	Verification Procedure	Measured Result	Status
Hand tracking	Use MediaPipe to track 21 hand landmarks in real time; map index fingertip to projector coordinates.	Tracking stable; fingertip mapped accurately; RMS error $<$1 px.	Pass
Gesture events	Apply exponential smoothing and state-lock mechanism; detect pinch click and hover states.	Pinch click and hover triggered correctly; hover false-trigger rate 3–5%.	Pass
PTZ platform	Adjust yaw and pitch; pre-align platform via phone control.	Smooth motion; alignment error $< 5^\circ$; continuous operation 30 min without anomalies.	Pass
Calibration	Four-corner ArUco marker calibration; compute homography matrix for camera-to-projector mapping.	Mapping maintained accurately; position deviation within required tolerance.	Pass
Control loop	Arduino Nano ESP32 + PCB; PC to embedded communication; PWM control of motors.	Continuous operation verified 5–30 min; correct PWM sequencing.	Pass
AirTouch interaction	Complete interaction loop with projected demo applications.	Full loop functional; gestures correctly trigger cursor and click events.	Pass

3.2 Verification Procedures and Quantitative Results

System performance was validated through experiments and measurements across the major functional modules, including calibration accuracy, cursor tracking stability, pinch gesture recognition, end-to-end latency, software algorithm correctness, and motor platform reliability.

For calibration accuracy, the homography-based coordinate mapping was tested using both synthetic and physical setups. In the synthetic evaluation, 48 calibration points were sampled on an 8×6 grid and mapped using the computed homography matrix. The RMS mapping error was 0.32 pixels, and the maximum single-point error was 0.87 pixels, satisfying the ± 5 pixel design requirement. In the physical validation, the calibration pattern was projected onto a white wall approximately 1.5 m from the projector, and 10 pre-marked target locations were used for comparison. The average offset was 3.2 pixels with a standard deviation of 1.8 pixels, and the maximum observed offset was 6.1 pixels.

For cursor tracking stability, the system compared the mapped cursor coordinates before and after applying EMA smoothing. The experimenter held the index finger stationary for 30 s while the system recorded cursor coordinates at 30 fps. Without smoothing, the coordinate standard deviation was ± 4.7 pixels horizontally and ± 5.3 pixels vertically. After applying the EMA filter with $\alpha = 0.2$, the standard deviation decreased to ± 1.1 pixels and ± 1.4 pixels, reducing tracking jitter by approximately 75%. The added delay was approximately 1–2 frames, or 33–66 ms at 30 fps, and was not noticeable during normal interaction.

To evaluate pinch gesture recognition, 50 pinch-click gestures and 50 hover-only passes were tested. The system correctly identified 47 pinch gestures and 46 hover passes, corresponding to a 94.0% true positive rate and a 92.0% true negative rate. False positives mainly occurred during natural hand repositioning, while false negatives occurred when the pinch motion did not cross the threshold.

End-to-end interaction latency was evaluated by measuring the time between the user's pinch gesture and the corresponding projected visual feedback using a 120 fps high-speed camera. Over 20 trials, the average latency was 72 ms with a standard deviation of 18 ms, and the maximum observed latency was 105 ms. Most trials satisfied the 100 ms target, while occasional exceedances were mainly caused by frame processing jitter and operating system scheduling variability.

The software algorithm and motorized platform were also verified through automated and continuous operation tests. The calibration and mapping subsystem passed 48 automated test cases covering marker generation, homography computation, coordinate mapping, platform pose update, EMA smoothing, fingertip coordinate extraction, and full synthetic pipeline integration. For the motorized platform, mobile-phone commands were transmitted to the Arduino over Bluetooth, parsed into PWM control signals, and delivered to the motor driver. The system operated continuously for 30 min with periodic yaw and pitch commands, with no communication errors, motor stalls, or unexpected resets. Across 20 commanded rotations of 10°, the average angular error was 3.2° and the maximum error was 4.8°, satisfying the coarse alignment requirement.

3.3 Verification Limitations

Several limitations were identified during testing. Lighting variations, reflections, and surface materials may affect ArUco marker detection and hand tracking stability. Under extreme processing load or rapid hand motion, system response may slightly exceed the measured latency range. The test sample size was also limited, and not all hand sizes, gesture styles, or extreme operating conditions were covered.

The current implementation primarily supports single-user, single-finger interaction. Multi-user interaction, multi-touch gestures, and large-screen collaborative scenarios were not validated. In addition, the motorized platform is used for coarse pre-alignment rather than precise closed-loop projector control, so large changes in platform pose may still require recalibration.

4 Cost

4.1 Cost Analysis

Table 4: Hardware cost (as of April 31, 2026).

Part	**Quantity**	**Cost (RMB)**
PCB	1	71.85
PH2.0 Pin Header	4+4	3.16
Collision Sensor	2	114
SMD Resistor	1	3
L298N Motor Driver	2	15.58
Power Supply Interface	2	5
XH2.54 3P Dual Connector	4	105.5
Arduino Nano ESP32	1	139.99
Worm-Gear DC Motor	1+1	379.79
XH2.54 2P Male Connector 50m	4	8
Wire Terminal 5 pairs	1	9.9
3mm 3P Pin Header	4	3.08
Spare Components	-	45.73
12V Li-ion Battery	1	63.6
New PCB	1	49.5
New Arduino Switch + Magnetic Connector	1	225.19
Total	-	**1242.87**

The table above summarizes all hardware purchased up to April 31, 2026. The major cost items include the worm-gear motors, the depth-sensing and control modules, and the Arduino-based PCB. This cost analysis represents the minimum hardware needed to validate the prototype; additional components may be acquired in future iterations.

4.2 Schedule

Table 5: Project schedule with weekly tasks and team member responsibilities (slightly condensed).

Week	Jie Xu	Zibo Dai	Yuqi Tang	Jing Weng
3/30–4/5	Set up MediaPipe; research RealSense SDK and data stream.	Select PTZ motors; design and 3D print bracket.	Study calibration algorithms; compute Homography H.	Set up Arduino Nano ESP32; plan PCB layout and power.
4/6–4/12	Implement fingertip tracking; align camera color and depth frames.	Assemble PTZ bracket; install motors and drivers.	Develop QR-based auto-calibration; coordinate transformation.	PCB assembly; wire routing and power testing.
4/13–4/19	Map coordinates using Homography H; implement mouse control via pynput.	Write PTZ PID code; adjust angles.	Develop touch logic using Z-axis depth.	Write motor drivers on Arduino Nano ESP32; establish serial link.
4/20–4/26	Refine Click/Drag recognition; implement state-lock debounce.	Test PTZ deadzone; verify gesture-angle mapping; ensure safety.	Develop virtual keyboard; integrate touch input logic.	Debug PCB; integrate power and control signals.
4/27–5/3	Apply first-order low-pass filter; smooth gesture trajectory.	Safety analysis; optimize PTZ motion; verify limits.	Test click offset and double-click; system integration.	Integrate Arduino Nano ESP32 with PTZ and sensors; complete AirTouch loop.
5/4–5/17	Full AirTouch loop integration and debugging; test latency and accuracy.	Draft mechanical and safety sections of report.	Organize calibration data; prepare final acceptance report.	Final system testing; ensure reliability; prepare presentation.

5 Conclusion

This project demonstrates a projection-based interactive system that integrates computer vision with embedded hardware control. The system uses camera input and homography-based calibration to create an interactive projected interface, while hover and pinch-based click detection support contact-free interaction.

Filtering and state-lock mechanisms improve cursor stability and reduce accidental triggering. On the hardware side, the Arduino-based PCB and Bluetooth communication allow users to control yaw and pitch motors for projector pre-alignment.

5.1 Accomplishments

The prototype meets all core requirements. ArUco calibration computes the homography within 2–3 s (< 1.0 pixel synthetic RMS error, $\sim$3 pixels physical offset). MediaPipe tracking operates at 30 fps, with EMA smoothing keeping cursor jitter below ± 2 pixels, and pinch recognition achieving 94% true positive and 92% true negative rates. Additionally, the motorized platform ensures pre-alignment within 5° error across a stable 30-min control loop. Game demos confirmed successful full-loop interaction.

5.2 Remaining Limitations and Uncertainties

ArUco marker detection is ambient-light sensitive: while stable indoors, the success rate drops to 70% under direct sunlight or intense overhead illumination. Additionally, the normalized pinch threshold was calibrated for a single user and lacks extensive testing across diverse hand sizes. Additionally, The system supports only single-user, single-finger interaction, excluding multi-touch gestures or multiple independent cursors. Platform pose updates assume small rotations; exceeding 10° requires full recalibration. Finally, maximum latency reached 105 ms due to OS scheduling variability.

5.3 Future Work

Future work will utilize MediaPipe's `num_hands` parameter for multi-cursor and two-finger interaction, alongside adaptive pinch thresholds from brief calibration. Requiring stable, multi-frame ArUco detection before accepting the homography matrix will enhance calibration robustness, paving the way for advanced applications such as virtual keyboards, menu navigation, and collaborative interactive surfaces.

5.4 Ethical Considerations

For privacy and safety, camera data is used strictly for real-time tracking and marker detection without image storage or profiling. The system operates on a low-voltage 12V battery with no exposed conductors, safeguards user eyes from direct projection, and uses safety switches to prevent motorized over-rotation.

References

[1] C. Harrison, H. Benko, and A. D. Wilson, "Omnitouch: Wearable multitouch interaction everywhere," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 441–450.

[2] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[3] F. Zhang et al., "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[4] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.