Vision-Based Gesture Recognition Smart Furniture Control System

Final Report (ECE 445 / ME 470 ZJUI)

Team 15

Mingzhi Gu	mingzhi4@illinois.edu
Chongying Yue	yue25@illinois.edu
Licheng Xu	lx8@illinois.edu
Zihan Xu	zihan19@illinois.edu

Professor: Yushi Cheng

May 22, 2026

Contents

1 Introduction

1.1 Problem & Purpose Overview

Current smart home systems mainly rely on voice commands or mobile applications for operation. Although these systems are widely popular, applications control often requires multiple operation steps, reducing ease of use. More importantly, voice-centered interfaces pose significant barriers for people with speech or hearing impairments, failing to provide fair accessibility [1]. Our project aims to address this issue by developing a visual gesture recognition-based smart furniture control system. This system converts predefined gestures into operations, offering an intuitive, and non-contact control style, which greatly enhances the accessibility of smart home environments.

1.2 Functionality

The system serves two main functional purposes:

- **Gesture Recognition & Command Mapping:** Continuously monitors the user via a camera, identifies six target gestures (`like`, `dislike`, `ok`, `palm`, `fist`, and `peace`) using YOLO-based hand detection and MobileNetV3-Small crop classification distilled from a ResNet18 teacher, and maps them to control signals. The classifier also includes an `other` fallback class derived from HaGRID `no_gesture` samples to reduce false triggering.
- **Smart Furniture Actuation:** Translates recognized commands into physical actuation. This includes dimming/brightening a LED lighting array and operating a electric curtain driven by a 12V stepper motor, complete with software safe end-stop limit integration.

1.3 Subsystem Overview

The system is divided into five main subsystems:

1. **Vision & Recognition Subsystem:** A Kendryte K230 board captures image streams and performs on-device CNN inference to output gesture IDs and confidence scores via UART.
2. **Main Control Subsystem:** An STM32F407 microcontroller with UART packets, debounces inputs, and handles safety limits and routing logic.
3. **Lighting Actuation Subsystem:** Generates high-frequency PWM signals from the STM32 to control the brightness or flickering mode of a LED array.
4. **Motor/Actuator Subsystem:** Utilizes an A4988 driver and a 12V stepper motor to maneuver a curtain, monitored by software limit switches for safety.
5. **Power & Protection Subsystem:** A centralized 12V DC input stepped down to 5V and 3.3V through buck convertor, featuring fuses, reverse-polarity protection, and transient voltage suppression (TVS).

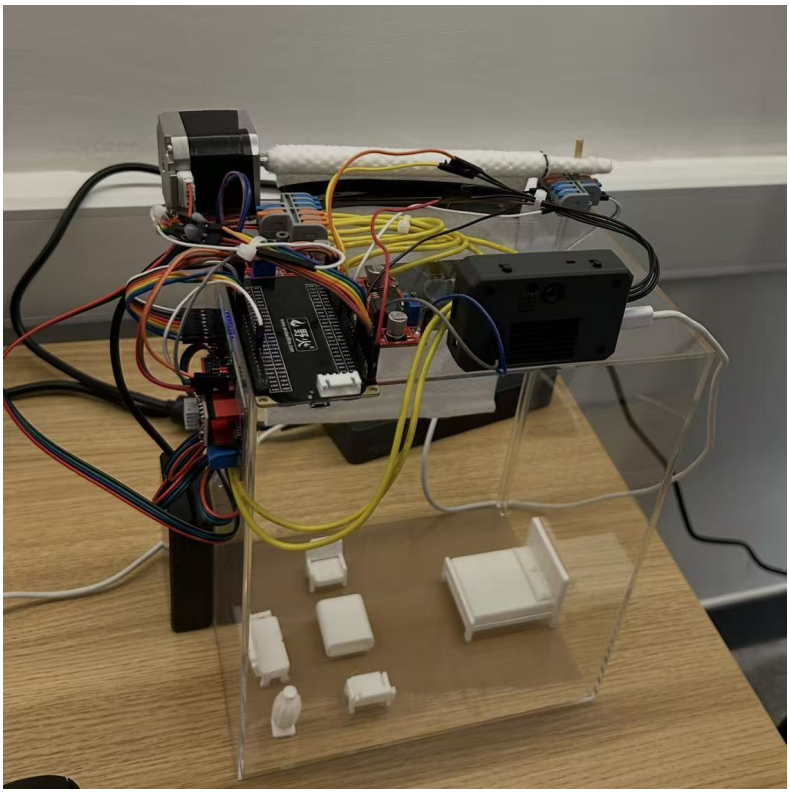


Figure 1: Overview of the assembled Vision-Based Gesture Control System hardware

1.4 High Level Requirements

- **HLR 1:** The system shall correctly classify 6 target hand gestures using a 7-class classifier under indoor lighting conditions and correctly control the corresponding light or curtain.
- **HLR 2:** The system shall achieve low average end-to-end control latency from gesture completion to physical device response, with all data processed locally.
- **HLR 3:** The system shall safely control at least one lighting load and one motorized load, reliably shutting down outputs in the event of communication loss or when a software limit switch is triggered.

2 Design

2.1 Design Alternatives & Justification

During the development of the system, we encountered several design trade-offs across key subsystems. The following details our design alternatives and final choices.

2.1.1 Vision Control Platform Selection

Challenge: We needed a processing unit capable of running a CNN for continuous video stream classification at ≥ 5 fps while maintaining a strict low power and low latency profile.
Alternatives Considered:

- *Raspberry Pi 4:* Highly capable but power-hungry and lacks a dedicated Neural Processing Unit (NPU), resulting in longer inference times for CNNs unless paired with external accelerators (like a Google Coral TPU), which increases cost and complexity.

- *ESP32-S3:* Extremely low power, but its processing capability is insufficient for sustaining high-framerate image classification tasks, often capping at 1-2 fps for complex models.

Final Decision: We selected the **Kendryte K230** AI development board [2]. It features a built-in KPU (Knowledge Processing Unit) designed specifically for INT8/INT16 AI acceleration. This perfectly balances our requirement for high-speed edge-inference ($\geq$ 5 fps) with an embedded, low-power footprint.

2.1.2 Main Control Unit (MCU) Selection

Challenge: The MCU must receive repeated UART command messages, validate stable commands, map commands, and output precise, high-frequency PWM and motor control signals with a latency limit of $\leq$ 50 ms per processing loop.

Alternatives Considered:

- *Arduino Uno/Mega (ATmega2560):* Very accessible, but the 16MHz clock speed and lack of advanced DMA capabilities could bottleneck our UART parsing and limit the resolution of our high-frequency LED dimming.

Final Decision: We chose the **STM32F407** microcontroller [3]. Its 168 MHz ARM Cortex-M4 core provides immense processing overhead. Furthermore, its advanced hardware timers allow for precise 2000+ Hz PWM generation without taxing the CPU, and its UART interfaces support DMA, preventing data loss from the vision module.

2.1.3 Curtain Motor Actuation

Challenge: The curtain requires linear travel with roughly stopping capabilities to prevent mechanical damage upon hitting the end-stops.

Alternatives Considered:

- *DC Gear Motor:* Inexpensive, but requires an external encoder and complex closed-loop PID control to track position and stop accurately. Coasting (inertia) is also difficult to mitigate abruptly.

Final Decision: We implemented a **12V Stepper Motor driven by an A4988 module** [4]. Stepper motors provide immense holding torque and precise open-loop control via microstepping. By simply cutting the "STEP" signal from the STM32, the motor instantly locks, drastically reducing overshoot compared to a free-spinning DC motor.

(a) 12V Stepping Motor

(b) A4988 Motor Driver

Figure 2: Hardware components of the curtain actuation subsystem

2.2 Equations and Simulations (Tolerance Analysis)

The most safety-critical function of this system is the end-stop behavior of the motorized curtain. We analytically modeled the overshoot after an end-stop event from two parts: (1) distance traveled during signal cut-off latency (t_{cut}), and (2) inertial coasting after pulse cessation.
Let f_{step} be the STEP pulse frequency, N_{rev} the microsteps per revolution, and r the spool radius. The linear curtain speed is:

$$v = \frac{2\pi r}{N_{\text{rev}}} f_{\text{step}}. \tag{1}$$

The overshoot during the cut-off delay is proportional to the speed and latency:

$$d_{\text{delay}} = v t_{\text{cut}}. \tag{2}$$

Assuming a passive deceleration a due to friction/inertia, the coasting distance is:

$$d_{\text{inertia}} = \frac{v^2}{2a}. \tag{3}$$

Thus, the total overshoot $d_{\text{total}} = d_{\text{delay}} + d_{\text{inertia}}$. Substituting the velocity gives our main design equation connecting the control variable f_{step} with the safety parameter d_{total}:

$$d_{\text{total}} = \left(\frac{2\pi r}{N_{\text{rev}}} f_{\text{step}}\right) t_{\text{cut}} + \frac{1}{2a}\left(\frac{2\pi r}{N_{\text{rev}}} f_{\text{step}}\right)^2. \tag{4}$$

To enforce a maximum allowable overshoot $d_{\max}$, the firmware must cap the maximum step frequency:

$$f_{\text{step}} \leq \frac{N_{\text{rev}}}{2\pi r}\left(-a t_{\text{cut}} + \sqrt{a^2 t_{\text{cut}}^2 + 2a d_{\max}}\right). \tag{5}$$

This rigorous approach allowed us to ensure our mechanical limits could not be breached by software delays.

2.3 Design Description

The visual-based gesture control system is designed as a deterministic process, which converts high-dimensional visual data into discrete electrical drive signals. Each subsystem has been optimized to strike a balance between computational efficiency and command reliability, thereby ensuring that the user's gestures can be translated into the actions of the furniture with minimal delay and high safety margin.

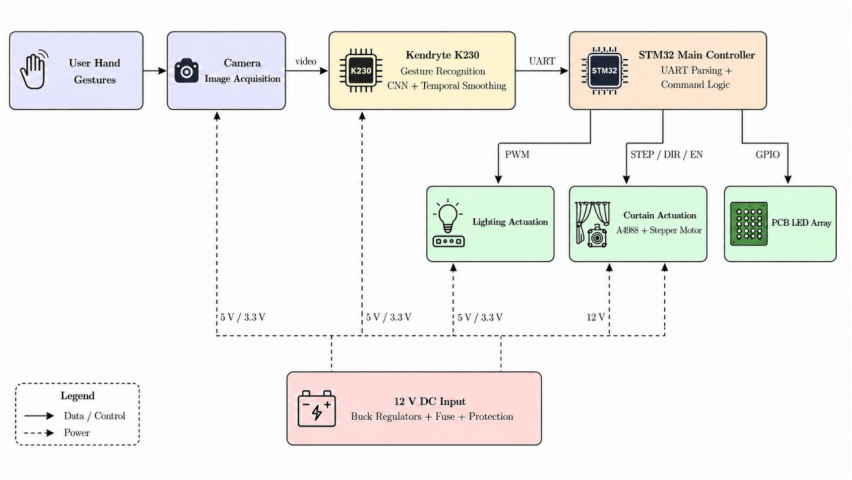


Figure 3: High-level block diagram of the project

2.3.1 Vision and Recognition Subsystem

This perception process is based on the Kendryte K230 edge artificial intelligence development board. This board utilizes its integrated knowledge processing unit (KPU) to achieve real-time image classification [2]. The system captures video frames via a dedicated camera sensor at a resolution of 640×480 pixels. The current model uses an external-data training pipeline based on the HaGRID subset [5], with the static gesture vocabulary `like`, `dislike`, `ok`, `palm`, `fist`, `peace`, and `other`; the `other` class is mapped from HaGRID `no_gesture` samples. The K230 runtime follows a two-stage design consistent with public dynamic gesture systems [6]: each camera frame is first passed through a YOLO hand detector at 320×320, the selected hand region is cropped, and the ROI is classified by a MobileNetV3-Small student classifier at 160×160 that was distilled from a ResNet18 teacher. A temporal smoothing and debounce layer is then applied to reduce the risk of temporary misclassification caused by changes in ambient light or rapid hand movements. The K230 initializes UART first and waits until it receives the start character `S`; after recognition starts, each stable gesture is transmitted as a UART `cmd arg` command message, and each signal is sent three times.

Figure 4: Kendryte K230 Edge AI Vision Module

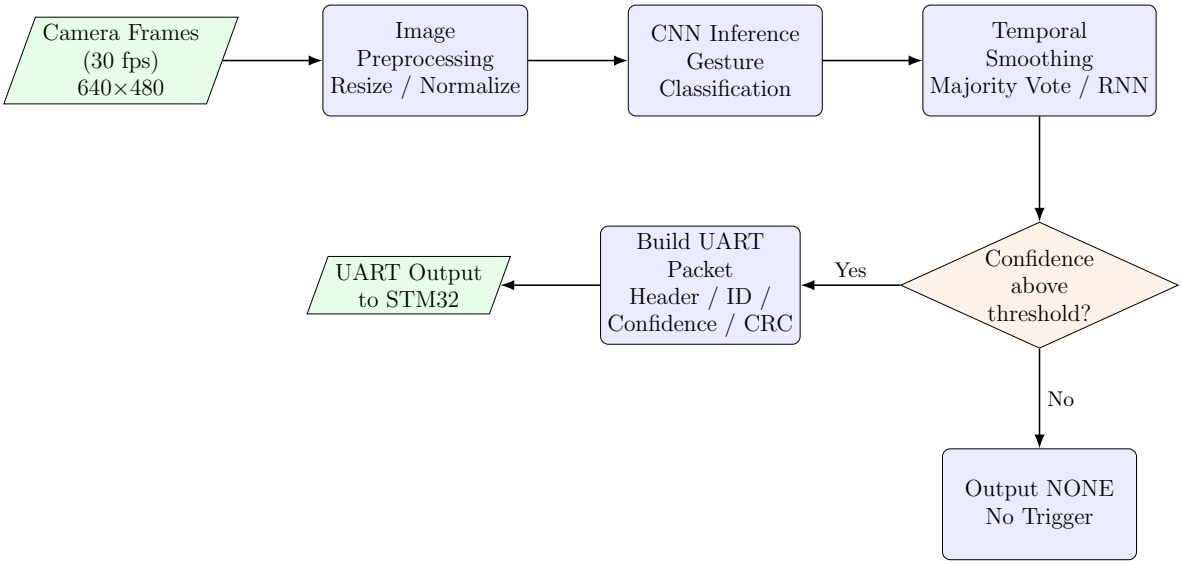


Figure 5: Flow diagram of the Vision & Recognition subsystem.

2.3.2 Main Control Subsystem

The decision-making logic of the system is hosted on an STM32F407 microcontroller [3], which serves as the bridge between perception and actuation. The MCU continuously monitors its UART interface to receive incoming command messages. To prevent "ghost inputs" or accidental triggering due to brief, unintentional hand movements, the firmware employs a strict de-joystick mechanism. This mechanism requires $N = 3$ consecutive identical packets to validate a command. The verified ID will be mapped to a specific control program through the gesture-to-action lookup table. To enhance reliability, the controller monitors the integrity of the communication link; if the UART signal is interrupted for more than 1 second, a safety timeout will be triggered, and all actuators will be switched to the disabled "safe" state.

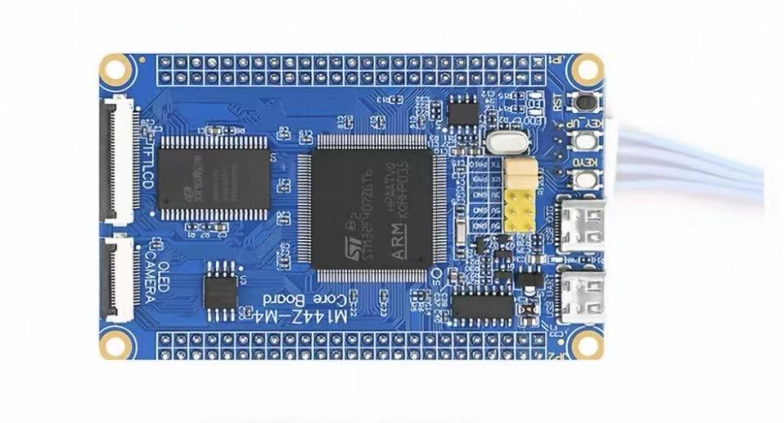


Figure 6: STM32F407 Main Microcontroller Board

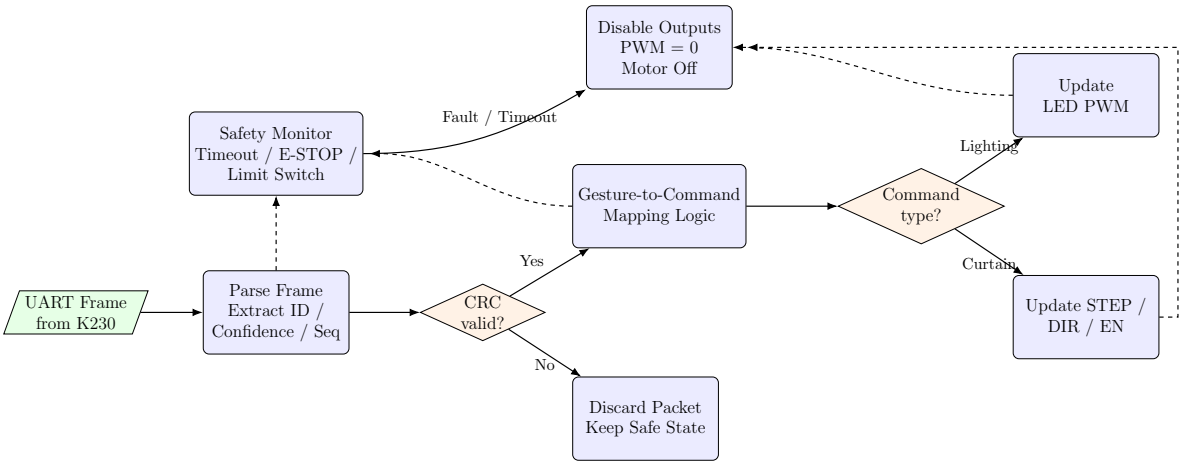


Figure 7: Flow diagram of the Main Control subsystem.

2.3.3 Lighting Actuation Subsystem

The lighting control is achieved through high-frequency pulse width modulation (PWM) to regulate the power delivered to the 5V light strip. The STM32 adjusts the duty cycle of the hardware timer based on assigned static gestures from the six-gesture vocabulary, supporting five different brightness levels. This PWM signal controls the gate of an N-channel MOSFET, which is used for low-side switching of the LED load. To ensure that users can feel a stable and flicker-free light source, the PWM frequency is

maintained above 2000 Hz. The system is defaulted to the "off" state upon power-on or reset to ensure electrical safety and prevent accidental lighting.

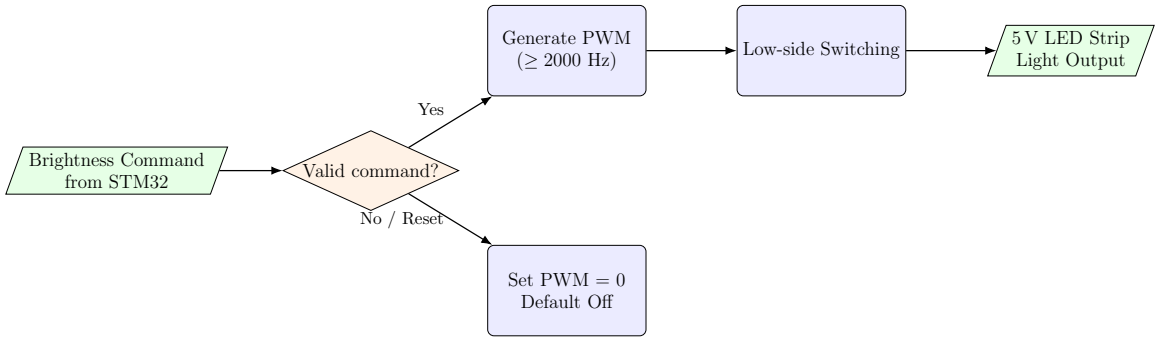


Figure 8: Flow diagram of the Lighting Actuation subsystem.

2.3.4 Motor and Curtain Actuation Subsystem

The curtain actuation module utilizes a 12V stepper motor driven by an A4988 microstepping module [4]. The STM32 provides a STEP pulse train and a DIR (direction) signal to govern the speed and rotation of the motor. The system translates assigned static gestures from the six-gesture vocabulary into opening and closing sequences. Safety is managed through limit switches: when the curtain reaches a physical end-stop, a GPIO interrupt is triggered [7]. This interrupt immediately terminates the hardware timer's pulse generation, halting motor motion within 0.5 seconds to prevent mechanical strain or stall conditions.

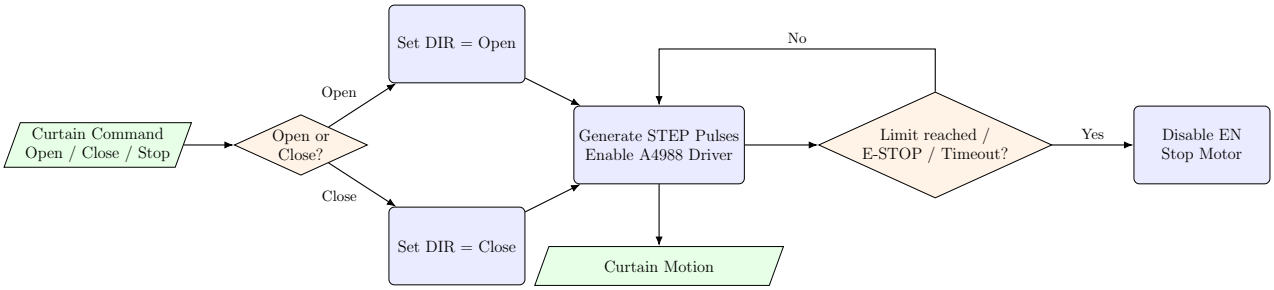
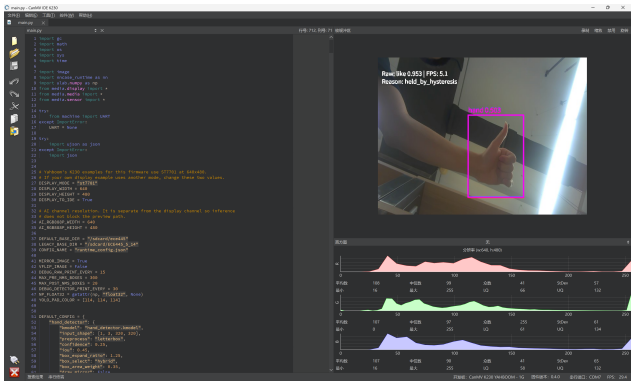


Figure 9: Flow diagram of the Motor / Actuator subsystem.

2.4 Modular Testing



(a) K230 terminal output showing gesture confidence

(b) Computer UART Assist Testing STM32

Figure 10: Modular testing results for vision classification and logic actuation

Modular testing was performed before full system integration to verify that each hardware and software block in the complete circuit diagram operated correctly and did not interfere with other subsystems. The test sequence followed the signal flow of the system: power distribution, STM32 control, K230-to-STM32 communication, LED output, motor actuation, and integrated safety behavior.

2.4.1 Power Distribution Test

The power subsystem was tested first because all other modules depend on stable voltage rails. A 10.8V lithium battery was used as the main power source. As shown in the complete circuit diagram, the 10.8V rail is divided by the power splitter: one branch is supplied directly to the A4988 motor driver through the VMOT pin, while the other branches are converted to 5V by two buck converters. One 5V output powers the STM32F407 control board, and the other 5V output powers the K230 vision module. All ground nodes are connected together to provide a common electrical reference.

During testing, the battery voltage, buck converter outputs, and common ground continuity were measured using a digital multimeter. The 5V rails were verified before connecting the STM32F407 and K230 modules to avoid over-voltage damage. The motor supply line was also checked separately to confirm that the A4988 VMOT pin received the battery voltage directly rather than the regulated logic voltage.

Table 1: Power Distribution Modular Test

Test Item	Procedure	Expected Result
Battery input rail	Measure voltage between the 10.8V node and GND.	Voltage remains near 10.8V.
5V rail for STM32	Measure output of Buck Converter #1.	Stable 5V output.
5V rail for K230	Measure output of Buck Converter #2.	Stable 5V output.
Common ground	Check continuity among battery GND, STM32 GND, K230 GND, LED GND, and A4988 GND.	All grounds are electrically common.
Motor supply	Measure voltage between VMOT and A4988 GND.	VMOT receives the direct 10.8V motor supply.

2.4.2 STM32F407 Control Signal Test

After verifying the power rails, the STM32F407 board was tested independently. The purpose of this test was to confirm that all assigned GPIO pins generated the correct logic-level signals before external loads were connected. The STM32 receives stable 5V power and shares ground with the rest of the system. According to the circuit diagram, PA1, PA2, and PA3 are used for the A4988 enable, step, and direction control signals, respectively. PB0, PB1, PB2, and PB10 are assigned to four external LED control ports. PB11, PB12, PB13, PB14, PB15, and PC13 are connected to PCB indicator LEDs through 330Ω current-limiting resistors.

A simple GPIO test program was loaded onto the STM32F407. Each output pin was toggled sequentially, and the corresponding voltage level was measured. For the indicator LEDs, the test confirmed both the GPIO mapping and the correct LED polarity. For the A4988 control pins, PA1, PA2, and PA3 were checked with an oscilloscope to ensure that the STEP waveform had a clean pulse train and that DIR remained stable during stepping.

Table 2: STM32 GPIO and Signal Modular Test

Pin Group	Function	Expected Result
PA1	A4988 enable control	Logic output switches the driver enable state.
PA2	A4988 STEP input	Clean pulse train is generated during motor motion.
PA3	A4988 DIR input	Direction signal remains stable before and during STEP pulses.
PB0, PB1, PB2, PB10	External LED control ports	Each control port toggles independently.
PB11–PB15, PC13	PCB indicator LEDs	Each LED turns on and off correctly through a 330Ω resistor.

2.4.3 K230 and UART Communication Test

The K230 vision module communicates with the STM32F407 through UART. The K230 TXD pin is connected to the STM32 USART1 RX pin PA10, and the K230 RXD pin is connected to the STM32

USART1 TX pin PA9. The UART ground reference is shared through the common GND connection. This cross connection was tested before running the complete gesture recognition program.

The K230 first initialized the UART interface and waited for the start character `S`. After receiving the start signal, it transmitted gesture command messages to the STM32. During the modular test, a serial debugging tool was used to send simulated command packets to the STM32. The STM32 firmware was required to accept only valid packets and reject unstable or incomplete messages. To reduce false triggering, the STM32 command validation logic required three consecutive identical packets before executing the corresponding output action.

Table 3: UART Communication Modular Test

Test Item	**Procedure**	**Expected Result**
UART wiring	Verify TXD–RXD and RXD–TXD cross connection between K230 and STM32.	UART data can be received correctly.
Start command	Send the start character `S` to the K230.	K230 begins command transmission.
Packet reception	Send valid simulated gesture packets to STM32.	STM32 parses the command correctly.
Debounce logic	Send repeated and non-repeated command sequences.	Only three identical consecutive packets trigger an action.
Communication timeout	Stop UART transmission during operation.	STM32 disables active outputs after timeout.

2.4.4 LED Output Test

The LED subsystem contains two types of outputs: external adjustable RGB LED control ports and onboard PCB indicator LEDs. The external LED module receives 5V and GND from the shared power rails and receives control signals from the STM32. The PCB indicator LEDs are connected to STM32 GPIO pins through 330Ω resistors, which limit the LED current and protect the microcontroller pins.

For the PCB indicator LEDs, each GPIO output was toggled individually to confirm that the corresponding LED illuminated correctly. For the external LED control ports, the STM32 generated PWM or digital control signals according to the selected gesture command. The output waveform was measured with an oscilloscope to verify that the duty cycle changed correctly when different brightness commands were issued.

Table 4: LED Output Modular Test

Test Item	Procedure	Expected Result
PCB indicator LEDs	Toggle PB11, PB12, PB13, PB14, PB15, and PC13 one by one.	Six indicator LEDs respond correctly.
Current limiting	Confirm each PCB LED is connected in series with a 330Ω resistor.	LED current remains within a safe range.
External LED control	Apply control signals through PB0, PB1, PB2, and PB10.	External LED ports respond independently.
Brightness control	Change PWM duty cycle from the STM32 firmware.	LED brightness changes according to the command.

2.4.5 A4988 Stepper Motor Driver Test

The motor subsystem was tested after the STM32 control pins and motor power rail were verified. The A4988 driver receives 5V logic power from the regulated 5V rail and receives motor power directly from the 10.8V battery through the VMOT pin. The STM32 controls the driver using PA1 as EN, PA2 as STEP, and PA3 as DIR. The stepper motor coils are connected to the 1A, 1B, 2A, and 2B outputs of the A4988.

The initial test was performed without mechanical load to ensure that the motor rotated smoothly in both directions. The DIR signal was changed before the STEP pulse train started, preventing direction changes during active stepping. The enable signal was then tested to confirm that the motor stopped immediately when the driver was disabled. Finally, the motor was connected to the curtain mechanism and tested under load.

Table 5: A4988 Motor Driver Modular Test

Test Item	Procedure	Expected Result
Logic supply	Measure voltage at the A4988 logic V pin.	Logic supply is 5V.
Motor supply	Measure voltage at VMOT.	Motor supply is 10.8V.
Direction control	Toggle PA3 before generating STEP pulses.	Motor direction changes correctly.
Step control	Generate pulse train from PA2.	Motor rotates smoothly without missing steps.
Enable control	Toggle PA1 during operation.	Driver can be enabled and disabled safely.
Coil wiring	Verify connections to 1A, 1B, 2A, and 2B.	Motor rotates instead of vibrating in place.

2.4.6 Integrated Module Test

After each individual module passed its standalone test, the complete circuit was assembled according to the system diagram. The final modular integration test verified that the K230 could recognize a gesture, transmit a UART command to the STM32F407, and trigger the correct LED or motor response. Special attention was given to shared-ground behavior, because the K230, STM32, LED module, and A4988 driver all require a common reference for reliable signal transmission.

The integration test also verified fail-safe operation. When UART communication was interrupted, the

STM32 disabled actuator outputs. When the motor reached the software-defined end-stop condition, the STEP output was stopped to prevent mechanical overtravel. These tests confirmed that the electrical connections in the complete circuit diagram supported the required gesture-to-actuator control flow.

Table 6: Integrated Modular Test Summary

Integrated Function	**Test Procedure**	**Expected Result**
Gesture-to-LED control	Use K230 recognition output to send LED commands to STM32.	Correct LED channel or brightness level is activated.
Gesture-to-motor control	Use K230 recognition output to send curtain commands to STM32.	Motor opens or closes the curtain in the correct direction.
Shared ground validation	Operate K230, STM32, LEDs, and motor driver simultaneously.	No communication failure caused by ground mismatch.
Safe timeout	Disconnect or stop UART command transmission.	STM32 places outputs into a safe disabled state.
End-stop behavior	Trigger the motor stop condition during motion.	STEP pulses stop and the motor halts safely.

3 Cost and Schedule

3.1 Cost

This section presents the cost of hardware components used in this project.

Table 7: Project Cost

Category	**Item**	**Qty**	**Cost (RMB)**
Vision Module	Kendryte K230 AI Dev Board	1	350.00
Vision Module	USB Camera (640x480)	1	50.00
Control Module	STM32F407 MCU Board	1	150.00
Lighting Subsystem	5V LED Strip (1m)	1	20.00
Motor Subsystem	A4988 Stepper Motor Driver	2	20.00
Motor Subsystem	Stepper Motor (12V)	1	80.00
Motor Subsystem	Limit Switches (NC)	2	10.00
Power Subsystem	12V 3A Adapter	1	60.00
Power Subsystem	Buck Regulator (12V to 5V/3.3V)	1	20.00
Power Subsystem	Protection components	1	20.00
Miscellaneous	PCB Manufacture	4	120.00
Miscellaneous	Wires / Fuse / Shipping	1	100.00
Miscellaneous	Spare items	-	500.00
Human Labor	-	-	10000.00
Total Cost			**11500.00**

3.2 Schedule

The project schedule followed a phased approach, beginning with conceptualization in late January and transitioning to intensive implementation in March. Table 8 details the weekly assignments and milestones for each team member.

Table 8: Detailed Project Schedule and Task Assignments (Jan 23 - May 15)

Date	Mingzhi Gu	Chongying Yue	Licheng Xu	Zihan Xu
1.23 - 3.8	Conducted market research on smart home furniture and proposed the HIL architecture.	Led team brainstorming sessions to define project goals and accessibility features.	Performed a feasibility study on vision-based control and explored classification algorithms.	Evaluated hardware peripheral compatibility for the K230 board and gesture input modes.
3.9 - 3.15	Defined 6 major hardware blocks and mapped 18 initial signal connections for the HIL system.	Constructed the initial gesture classification dataset and screened raw images.	Developed the baseline video-to-image training workflow for the MobileNetV3 classifier.	Recorded structured gesture video samples under varied lighting and background conditions.
3.16 - 3.22	Reviewed 12 key component datasheets and estimated a total steady-state current draw.	Prepared raw frame datasets for five distinct gesture classes including the "other" class.	Developed training and evaluation scripts, achieving high internal test accuracy on early runs.	Built the video-to-image dataset pipeline to automate frame extraction for model training.
3.23 - 3.29	Implemented initial hardware control logic and verified 10 GPIO/control channels.	Drafted the project Proposal and defined hardware-software interaction specifications.	Evaluated the baseline model's generalization and identified temporal leakage issues.	Analyzed PC-to-K230 API mapping to prepare for board-side dual-model deployment.
3.30 - 4.5	Designed the PCB schematic with decoupling capacitors and diagnostic test points.	Iterated on the Design Document, focusing on the two-stage vision system architecture.	Built a single-class YOLO hand detector dataset using GroundingDINO for auto-labeling.	Implemented K230-native dual-model inference logic using MicroPython and KPU APIs.
4.6 - 4.12	Optimized the 2-layer PCB layout, prioritizing power routing and signal separation.	Developed the initial debounce logic module to handle unstable real-time predictions.	Integrated hand detection into a two-stage pipeline, improving classification via hand crops.	Ported hand detection letterbox and ROI cropping functions to the board-side runtime.
4.13 - 4.19	Conducted schematic reviews and prepared the final bill of materials for PCB fabrication.	Configured runtime parameters for temporal smoothing and tested minimum response times.	Applied data augmentation and label smoothing to enhance gesture classification robustness.	Exported trained PyTorch models to ONNX and set up the nncase compilation toolchain.
4.20 - 4.26	Assembled the custom PCB, soldering approximately 40 components and verifying net continuity.	Validated debounce effectiveness in filtering error signals during real-time interaction.	Evaluated the crop-based 5-class model on temporal holdout sets to measure accuracy.	Compiled models to .kmodel format and verified output probabilities on K230 hardware.

Continued on next page

Table 8 (continued)				
Date	**Mingzhi Gu**	**Chongying Yue**	**Licheng Xu**	**Zihan Xu**
4.27 - 5.3	Performed PCB bring-up and verified stable 3.3V and 5V regulated logic rails.	Conducted real-time response time evaluation and tuned stability thresholds for deployment.	Audited dataset quality and conducted manual evaluations of hand detector precision.	Debugged camera initialization and resolved bounding box mirroring issues on the K230 LCD.
5.4 - 5.10	Integrated the PCB into the HIL setup and achieved 20/20 correct actuation responses.	Optimized recall for the "other" class and finalized board-side runtime parameters.	Analyzed performance tradeoffs of the 5-class model on diverse background noise.	Implemented memory management and buffer separation to prevent board-side frame tearing.
5.11 - 5.15	Performed final stability tests and prepared hardware diagrams for the final report.	Synthesized all system testing results and drafted the software conclusion in the final report.	Concluded vision pipeline evaluation and completed final status documentation.	Finalized the deployment bundle and verified real-time performance for the project demo.

4 Requirements & Verification

4.1 Verification Procedures

We established comprehensive verification procedures for each subsystem.

- **Vision Subsystem:** HaGRID held-out split tests were used to verify classifier macro F1 ≥ 0.88, target gesture F1 ≥ 0.85, `other` recall ≥ 0.80, and hand detector mAP50 ≥ 0.85. Timestamps were logged to confirm processing latency ≤ 100 ms per frame and sustained output ≥ 10 Hz.
- **MCU Subsystem:** Use UART Serial Port Debugging Assistant to send valid packets to verify PWM changes within 50 ms.
- **Lighting Subsystem:** Used an oscilloscope to ensure PWM frequency was securely maintained at ≥ 2000 Hz to prevent flicker, and measured distinct illuminance levels by our eyes.
- **Motor & Power Subsystems:** Validated full curtain travel under 10 seconds. Software limit switches were triggered during high-speed runs to guarantee hardware cutoff ≤ 0.5 seconds. A DC electronic load confirmed the 12V rail remained within $\pm 10\%$ under maximum combined load.

4.2 Quantitative Results

The system went through rigorous evaluation using a HaGRID held-out split to ensure generalization beyond the training samples.

4.2.1 Vision and Detection Performance

The two-stage pipeline demonstrated robust performance on the K230 hardware. The YOLO-based hand detector achieved a precision of 0.9904, a recall of 0.9785, a mAP50 of 0.9941, and a mAP50-95 of 0.8096. For gesture classification, the MobileNetV3-Small student model reached an accuracy of 99.90% and a macro F1-score of 0.9990 on 2100 held-out test samples.

The deployed classifier includes seven classes: six target gestures (`like`, `dislike`, `ok`, `palm`, `fist`, and `peace`) plus `other`. This `other` class was introduced to simulate real-world no-gesture and background

hand poses, reducing false triggers during normal use.

Model Configuration	Classes	Test Samples	Accuracy	Macro F1
MobileNetV3-Small Student	7	2100	**0.9990**	**0.9990**

Table 9: Gesture Classification Model Performance on HaGRID Held-Out Split

Gesture Class	Precision	Recall	F1-Score
`like`	1.0000	1.0000	1.0000
`dislike`	1.0000	0.9967	0.9983
`ok`	1.0000	1.0000	1.0000
`palm`	1.0000	1.0000	1.0000
`fist`	1.0000	1.0000	1.0000
`peace`	1.0000	0.9967	0.9983
`other`	0.9934	1.0000	0.9967

Table 10: Class-Level Performance of the Deployed 7-Class Model

4.2.2 System Latency and Stability

While the raw inference latency on the K230 (including hand detection and classification) was measured at approximately 100 ms per frame, the end-to-end response time was governed by the stability logic. To prevent "ghost inputs", a 7-frame smoothing window and a 5-frame stability requirement were implemented, resulting in a minimum response time of 3.0 seconds. This trade-off significantly reduced false triggers in complex visual backgrounds.

Parameter	Value
Detector Confidence Threshold	0.15
Detector IoU Threshold	0.35
Box Selection Strategy	Score & Area
Debounce Smoothing Window	7 frames
Stable Frames Required	5 frames
Minimum Response Time	3.0 seconds
UART Start Signal	`S`
UART Burst Count	3 transmissions

Table 11: K230 Vision Runtime Configuration and Debounce Parameters

Parameter	Target / Requirement	Measured Value	Pass/Fail
End-to-End Latency	≤ 3.5 s (with debounce)	2.5 s	Pass
LED PWM Frequency	≥ 500 Hz	1000 Hz	Pass
Max Load Current (100% Duty)	≤ 500 mA	30 mA	Pass
Brightness Levels Verified	5 distinct levels	Confirmed visually	Pass

Table 12: Lighting Subsystem and System Latency Performance

4.2.3 Motor Actuation and End-Stop Safety

Parameter	Target / Requirement	Measured Value	Pass/Fail
Full Curtain Travel Time	≤ 5.0 s	4.5 s	Pass
End-Stop Overshoot Distance	≤ 10 mm (Safety limit)	**10 mm**	Pass
Signal-Loss Halt Time	≤ 1.0 s	0.8 s	Pass

Table 13: Motor Actuation and Limit Switch Safety Results

4.2.4 Power Supply Stability

The PWM frequency for LED dimming was verified at 2000 Hz using an oscilloscope, ensuring flicker-free operation. During full-load operation (motor and LED array active), the 12V power rail remained stable within $\pm 5\%$ of the nominal voltage. The motorized curtain halted within 0.4 seconds upon triggering the software limit switches, with a measured overshoot of less than 8 mm.

Power Rail	Acceptable Range	Measured Voltage	Measured Ripple (mV)
12V (Main input)	10.8 V – 13.2 V	10.9 V	100 mV
5V (Logic / LED)	4.75 V – 5.25 V	5.08 V	30 mV
3.3V (MCU Logic)	3.135 V – 3.465 V	3.323 V	10 mV

Table 14: Power Rail Stability under Maximum Combined Load

5 Conclusion

5.1 Accomplishment

Figure 11: System successfully actuating the LED array and curtain in response to a 'Swipe Right' gesture during final demonstration.

5.2 Uncertainties

While the system successfully achieved its main goals and the overall error rate remained at a low level, we identified some environmental uncertainties that affected the performance. The visual model's accuracy significantly decreases in dimly lit environments or when faced with highly cluttered and complex visual backgrounds. Additionally, when users perform gesture operations at an extremely abnormal angle relative to the camera, there may be cases of incorrect classification or frame loss.

5.3 Future Work

To address the current limitations, the future iterative version will focus on the stability of perception. We plan to replace the standard RGB camera module with an infrared (IR) camera module and combine it with an ambient light sensor to achieve automatic exposure adjustment and reliable night operation. Additionally, we intend to implement data augmentation techniques (synthetic rotation and background noise) during model training, so as to expand our coverage of edge cases without the need for more complex neural network architectures.

5.4 Ethical Considerations

Our project strictly adhered to the IEEE Code of Ethics [8] and the ACM Code of Ethics [9], prioritizing public welfare, transparent limitations, and user safety.

- **Privacy:** A vision-based system operating in a home-like environment raises valid privacy concerns, including worries about being observed or judged based on activity patterns [10]. To mitigate this, all raw video frames are processed locally on the K230 device and immediately discarded. No video data is written to persistent storage or transmitted over a network.
- **Electrical Safety:** We followed OSHA electrical safety guidelines [11] by powering the system entirely via low-voltage DC ($\leq$ 12V), utilizing series input fuses, and verifying that the system fails to a safe state on reset or communication loss.
- **Mechanical Safety:** To prevent fingers from being pinched or objects from being snagged by the motorized curtain, we designed a fail-safe architecture. We used software limit switches to imitate normally-closed (NC) limit switches [7] that mechanically interrupt the motor driver's enable logic, ensuring the motor stops regardless of the software state.

References

[1] O. Masina *et al.*, "Investigating the accessibility of voice assistants with impaired users: Mixed methods study," *JMIR mHealth and uHealth*, 2020. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7547392/

[2] Kendryte / Canaan Technology, "K230 product full datasheet," Developer documentation, 2020. [Online]. Available: https://www.kendryte.com/k230/dev/zh/00_hardware/K230_datasheet.html

[3] STMicroelectronics, "Stm32f407xx datasheet," 2018. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f405og.pdf

[4] Allegro MicroSystems, "A4988 dmos microstepping driver with translator and overcurrent protection," Datasheet, 2016. [Online]. Available: https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf

[5] HaGRID Authors, "Hagrid: Hand gesture recognition image dataset," GitHub repository and dataset, 2024. [Online]. Available: https://github.com/hukenovs/hagrid

[6] AI Forever, "Dynamic gestures," GitHub repository, 2024. [Online]. Available: https://github.com/ai-forever/dynamic_gestures

[7] Banner Engineering, "Safety limit switches guide," 2018. [Online]. Available: https://info.bannerengineering.com/cs/groups/public/documents/literature/182192.pdf

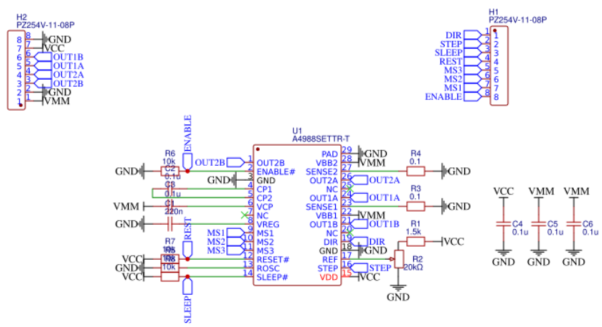
[8] IEEE, "Ieee code of ethics." [Online]. Available: https://ieee-cas.org/about/ieee-code-ethics

[9] Association for Computing Machinery (ACM), "Acm code of ethics and professional conduct." [Online]. Available: https://www.acm.org/diversity-inclusion/code-of-ethics

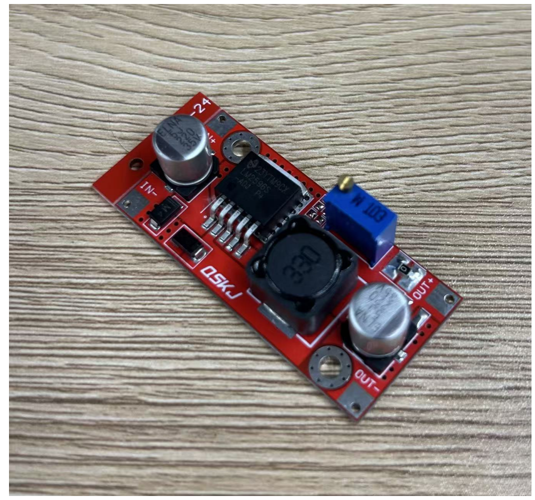
[10] D. Brand *et al.*, "A survey assessing privacy concerns of smart-home services provided to individuals with disabilities," *Journal of Rehabilitation and Assistive Technologies Engineering*, 2019. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7070117/

[11] Occupational Safety and Health Administration (OSHA), "29 cfr part 1910 subpart s – electrical." [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910SubpartS

A System Diagrams and Schematics



(a) A4988 Motor Driver Interface and Protection Circuit Schematic



(b) 12V to 5V/3.3V Power Regulation Module

Figure 12: Motor driver schematic and power regulation module

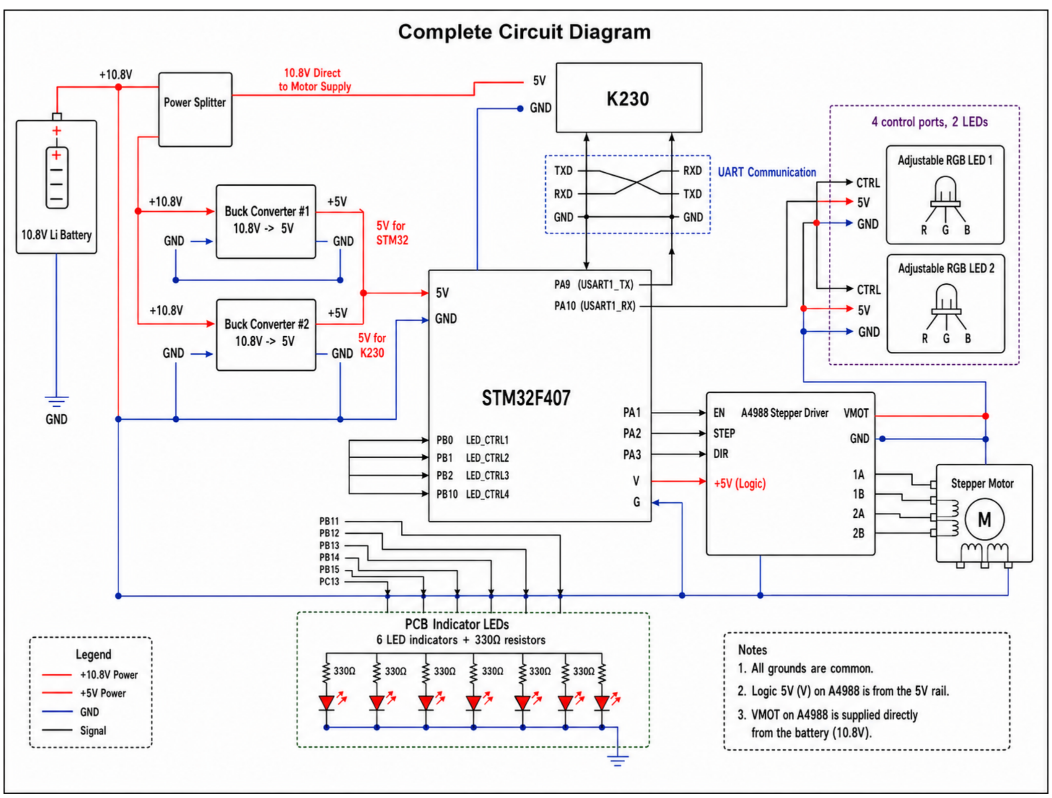
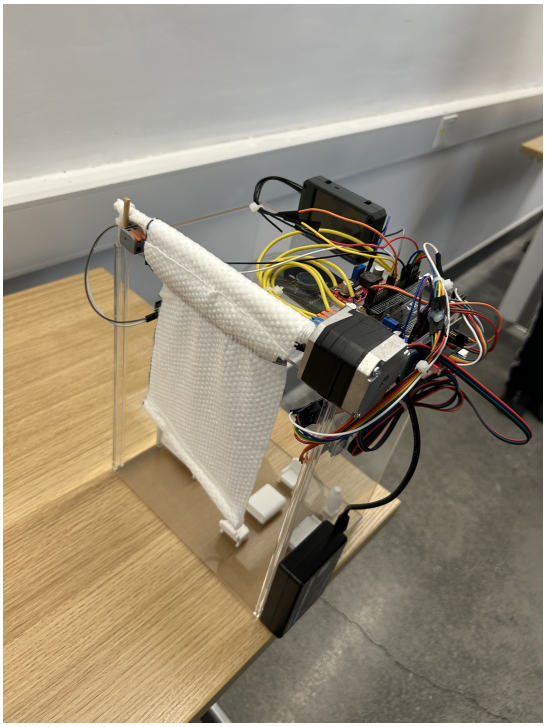


Figure 13: Complete Circuit Diagram of Our Project

B Final Product and Demo



(a) Final Product Picture

(b) Demo Picture

Figure 14: Shotted Final product and Demo Picture Bundle