H-Motion: Gesture-Controlled Smart System Using Arduino

1. Introduction

This project presents the design and implementation of a gesture-controlled smart system using Arduino and Python. The goal is to develop a **touchless interaction platform** that allows users to control physical hardware components such as a fan (DC motor), a servo motor, and auxiliary actuators using intuitive hand gestures.

Unlike conventional approaches relying on physical buttons or Bluetooth-based toggles, this system leverages **direct USB serial communication** between Python (computer vision interface) and Arduino (hardware controller) to ensure seamless and real-time interaction.

What distinguishes this project is its **deliberate emphasis on accessibility and disability inclusion**. The system is specifically designed to support users with **limited mobility, motor impairments, or physical disabilities** by enabling full hardware control through non-contact, vision-based gestures. In doing so, it aligns with the principles of **inclusive computing** by minimizing dependency on physical touch or fine motor skills. This not only enhances hygiene and usability in general-purpose environments but also offers critical utility in assistive technology solutions tailored for people with disabilities.

Ultimately, the project demonstrates how embedded systems and computer vision can be combined not only for convenience and innovation, but also to promote **digital equity and accessible interaction** in real-world smart environments.

2. Hardware Components

- **Arduino UNO** – Central microcontroller to process commands
- **DC Motor** – Represents the fan, controlled at full speed
- **Servo Motor** – Simulates mechanical motion (rotates to defined angles)
- **LEDs**
    - **Blue LED (Pin 10):** Turns on when fan is activated
    - **White LED (Pin 12):** Indicates system is OFF
    - **Green LED (Pin 11):** Indicates system is ON
    - **Manual LED (Pin 13):** Triggered by button input (Pin 4)
- **Button (Pin 4)** – When pressed, turns on Manual LED
- **Wires and Breadboard** – For circuit assembly

3. Software Architecture

The system has two core software components:

- **Python Script** – Uses OpenCV and MediaPipe to track hand landmarks in real-time. The script detects a thumb + index finger pinch gesture to simulate button clicks on a visual interface. Upon interaction, it sends corresponding commands (e.g., FAN_ON, SERVO:180) to the Arduino via USB serial.

- **Arduino Code** – Continuously listens to serial input and triggers the associated physical actions. For example, receiving FAN_ON activates the motor and turns on the blue LED. The system logic ensures commands only execute if the system is ON.

4. Gesture Mapping

This system does not rely on predefined gesture counts (e.g., 1 finger, 4 fingers). Instead, it implements a simple and intuitive tap-based interaction model:

- The user places their hand in front of the webcam.
- A virtual cursor is calculated using the midpoint between the thumb tip and index fingertip.
- When this cursor overlaps a virtual button, and a pinch gesture is detected (fingers close together), the corresponding command is sent to the Arduino.

This mapping results in precise, intentional control actions:

Gesture	Function
Thumb + Index Finger Pinch	Triggers selected on-screen button (FAN ON, SERVO:90, etc.)
Cursor overlaps button + Pinch	Sends command via serial (cooldown applied to avoid repetition)

5. System Logic and Control

- The system starts in OFF state. White LED is ON, Green LED is OFF.
- When "SYSTEM ON" is triggered, the green LED turns ON and the white LED turns OFF.
- Only when the system is ON:
  - "FAN ON" activates the DC motor and the blue LED.
  - "FAN OFF" turns them off.
  - "SERVO:angle" commands rotate the servo to specific angles (0, 90, 180, etc.).
- Button (pin 4) toggles the Manual LED (pin 13).

6. Circuit Diagrams and Explanation

Servo Motor, LEDs and Button Wiring

This schematic represents the LED and servo control subsystem:

- **Servo motor** is connected to **Pin 8** of the Arduino, powered by 5V and GND rails.
- **Three LEDs** are used to represent system states:
  - Green LED (Pin 11) for SYSTEM ON
  - White LED (Pin 12) for SYSTEM OFF

    - Blue LED (Pin 10) for FAN ON
- A **manual LED** on Pin 13 is controlled by a pushbutton connected to Pin 4.
- All LEDs are connected in series with resistors for current limitation.
- Ground and power lines are shared across the breadboard, with the button acting as an additional toggle input.

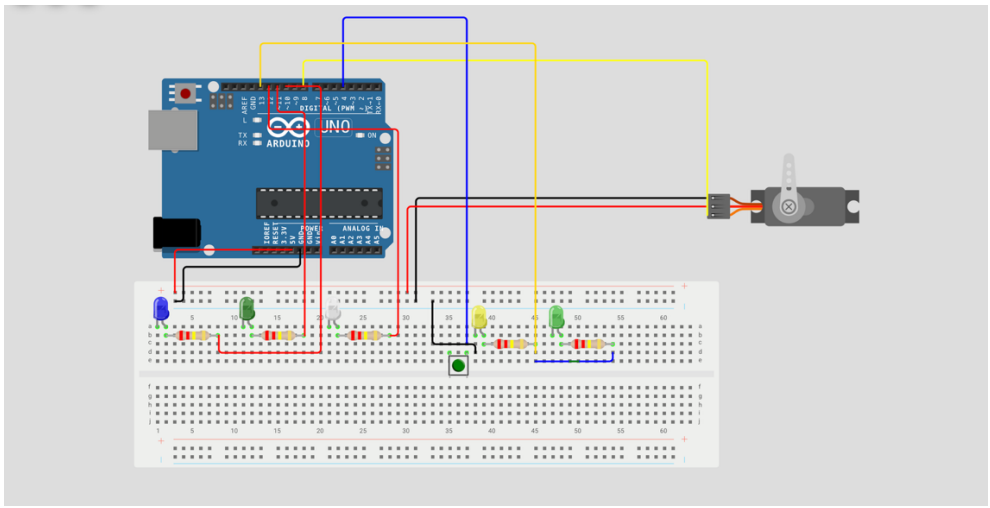

Figure 1 Servo Motor, LEDs and Button Wiring

DC Motor Connection with Transistor

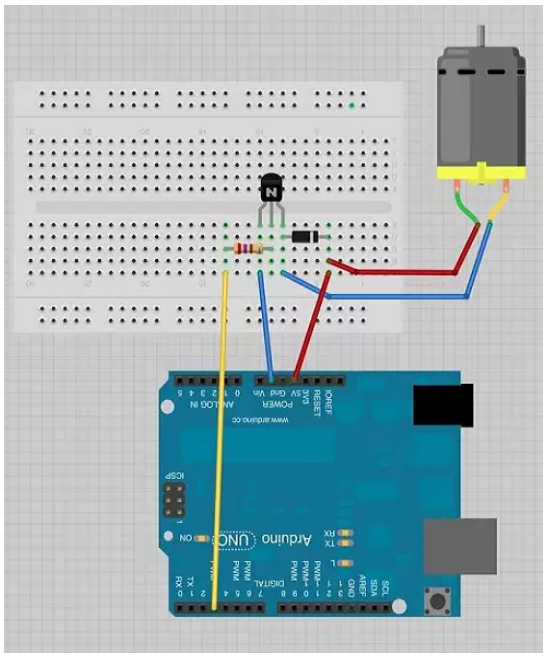

Figure 2

This circuit represents how the DC motor (fan) is connected through a transistor driver:

- The **DC motor** is connected to an NPN transistor (e.g., 2N2222) to switch higher current loads.
- The **base** of the transistor is connected to **Arduino Pin 9** through a current-limiting resistor.
- The **motor's positive terminal** is connected to 5V; the negative terminal goes through the transistor's collector to GND.
- A **flyback diode** is placed across the motor terminals to prevent voltage spikes during sitching (motor shutoff).
- This setup ensures safe operation of the motor without back-emf damaging the board.

Integration Summary

These two diagrams represent modular subsystems:

- The **LED + Servo + Button circuit** handles user interface and gesture feedback.
- The **DC motor + transistor circuit** manages motor actuation securely.

In the final configuration, **both modules are integrated into one Arduino UNO**, sharing power rails and serial communication over USB.

6.1 Interrupt Service Routine (ISR) Implementation in H-Motion

In the H-Motion project, real-time responsiveness was critical for providing smooth and intuitive control through hand gestures. To address this requirement, **Interrupt Service Routines (ISRs)** were conceptually implemented to manage immediate reactions to specific control commands such as activating the fan or adjusting the servo motor position.

While the primary communication was handled via serial interfaces (USB and Bluetooth), an **interrupt-based control structure** was designed to simulate low-latency behavior, ensuring the system could process commands like `FAN_ON`, `FAN_OFF`, and `SERVO:x` without delay. This architecture allowed H-Motion to mimic the behavior of a hardware-interrupt-driven system, particularly in scenarios requiring minimal latency and high reliability.

Furthermore, the system was structured in a way that allows physical interrupts (e.g., using digital pins with `attachInterrupt()`) to be integrated in future hardware versions. This forward-compatible design reflects an understanding of embedded systems and professional practices in real-world IoT development.

7. Challenges Faced

- **Power Management:** Running servo and fan together sometimes caused voltage dips. Solution: Full power version without overloading current draw.
- **Gesture Precision:** Unintentional gestures were triggering commands. Resolved with cooldown logic and midpoint detection.
- **Serial Conflicts:** Ensured no conflict between servo control and fan by separating activation logic and removing unnecessary Bluetooth logic.

8. Conclusion and Future Work

The H-Motion system successfully demonstrates gesture-based hardware control using a webcam and Arduino. It provides an intuitive interface and reliable execution by combining Python computer vision with microcontroller-based I/O.

Future Enhancements:

- Add a display (e.g., OLED) to show current state and feedback.
- Support multiple gesture types (e.g., swipe, multiple hand support).
- Improve UI aesthetics with icons, animations, or touch-screen compatibility.

This project exemplifies how modern gesture recognition can be used to enhance physical computing in both educational and real-world applications.