**Smart Gesture-Based Communication System for Non-Verbal Patient Assistance**

**Aim:**

The aim of the project is to develop a **Smart Gesture-Based Communication System for Non-Verbal Patient Assistance** that utilizes **finger count-based AI-driven hand gesture recognition** to enable bedridden and non-verbal patients to communicate their needs efficiently. By integrating **real-time ESP32 camera-based detection** and **Firebase**, the solution ensures timely caregiver response, enhances patient comfort, and improves overall healthcare assistance. Each finger count (1 to 5) corresponds to a specific request, allowing patients to convey their needs intuitively and effectively.

**Hardware Components:**

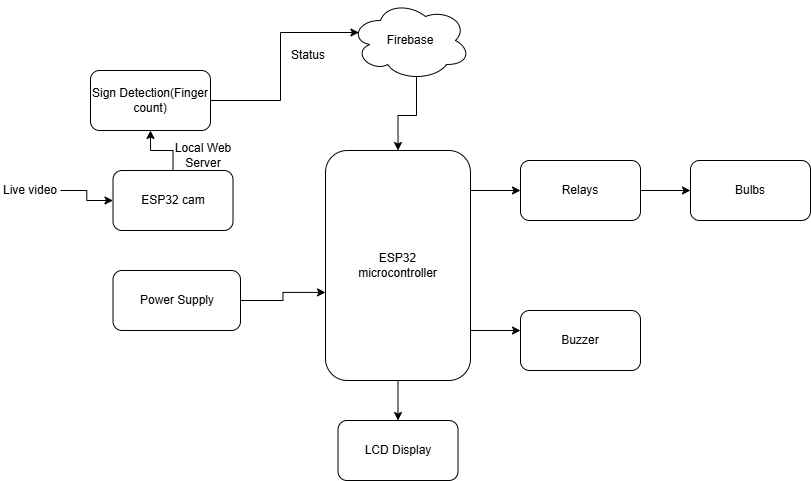
* **Source of Supply:** USB cable
* **ESP32 Camera**
* **Esp32 Microcontroller**
* **Lcd Display**
* **Buzzer**
* **Bulbs(Red,Green,Yellow,Blue,Pink)**
* **Relays**

**Software Tools:**

**Programming languages:** Python, Arduino.

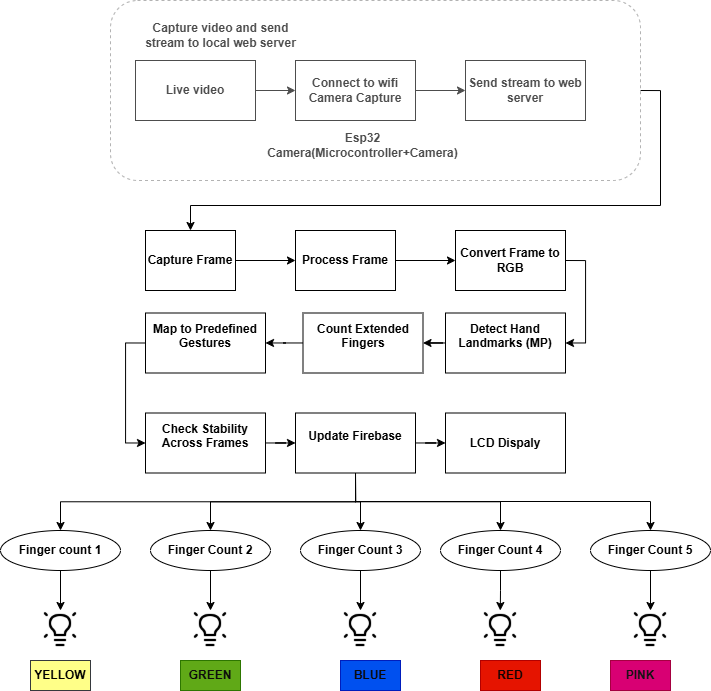
**Programming Tools:** Arduino IDE, Visual Studio.

**Block diagram:**



|  |  |
| --- | --- |
| **Component** | **Function** |
| Power Supply | Provides stable power to all components. |
| ESP32 Microcontroller | Acts as the central processing unit, managing data, gesture detection, and communication. |
| ESP32 Camera | Captures real-time video of hand gestures and sends it to the ESP32 for processing. |
| Gesture Recognition (Local Web Server) | Detects and processes hand gestures using MediaPipe Hand Tracking. |
| Firebase Realtime Database | Stores and updates the detected gestures for remote monitoring by caregivers. |
| Control Decisions (Based on Finger Count) | Determines the patient’s request based on detected finger count. |
| Relays and Bulbs | Activates specific bulbs to indicate the patient's needs:  • 1 Finger → Yellow (Water)  • 2 Fingers → Green (Food)  • 3 Fingers → Blue (Medicine)  • 4 Fingers → Red (Emergency)  • 5 Fingers → Pink (Happy) |
| Buzzer | Provides an alert sound for emergency requests. |
| LCD Display | Displays real-time status of the patient's request. |

**Work flow:**



**Implementation steps:**

**Capture Video and Send Stream to Local Web Server:**

This is the fundamental operation of the system. The system captures live video from a camera, which is then transmitted over a local WiFi network to a local web server. The flow looks like this:

* **Live Video Capture:** The camera continuously captures video frames, providing real-time visual input.
* **Connect to WiFi:** The device (likely an ESP32 microcontroller) connects to a WiFi network to enable wireless communication and streaming.
* **Send Stream to Web Server:** The captured video is encoded and sent as a continuous stream of data over the WiFi network to a local web server, which could be running on the same network.

**ESP32 Camera (Microcontroller + Camera):**

This section outlines the hardware involved:

* **ESP32 Microcontroller:** A low-cost, low-power device with built-in WiFi. It serves as the brain of the system, controlling the camera and managing the network transmission of the video feed.
* **Camera Module:** The ESP32 is connected to a camera module that captures the video frames, which are then processed and transmitted.

**1. Capture Frame:**

* The ESP32 captures individual video frames from the live video stream. A video is essentially a series of images (frames) displayed rapidly. Each frame is a captured that represents a moment in the video.

**2. Process Frame:**

* This is where the system performs the core operations of image processing:
  + **Convert Frame to RGB:** Each captured frame is converted into the RGB color space (Red, Green, and Blue). This is a standard format used for image processing, where each pixel is represented by the three color channels.
  + **Detect Hand Landmarks (MediaPipe):** The frame is passed through the **MediaPipe Hands API**, which is designed to detect and track hand landmarks in real time. The API identifies key hand points (like fingertips, palm center, knuckles) that are crucial for recognizing gestures.
  + **Count Extended Fingers:** Once the landmarks are detected, the system analyzes them to count how many fingers are extended. This typically involves comparing the relative positions of the detected hand landmarks, specifically looking at the tips of the fingers.
  + **Map to Predefined Gestures:** The system then maps the number of extended fingers to predefined gestures (e.g., 1 finger for water, 2 for food, etc.). Each gesture corresponds to a specific action or request.
  + **Check Stability Across Frames:** To prevent false or flickering gestures, the system verifies if the detected gesture remains consistent across multiple frames. If the same gesture is detected for a predefined number of frames (e.g., 5 consecutive frames), the gesture is considered stable and valid.

**3. Update Firebase:**

* After a stable gesture is detected, the system sends the gesture data (finger count) to the **Firebase Realtime Database**. Firebase acts as a remote storage and monitoring system where caregivers can access the latest detected gestures in real time.

**4. LCD Display:**

* The system provides visual feedback by displaying the representations like (water,food,medicine,emergency,happy) on an **LCD screen** connected to the ESP32.

**Representation:**

**1. Finger Count 1 to 5:**

* The system recognizes five distinct gestures based on the number of extended fingers:
  + **1 Finger → Water**
  + **2 Fingers → Food**
  + **3 Fingers → Medicine**
  + **4 Fingers → Emergency**
  + **5 Fingers → Happy**

Each number of fingers represents a different patient request.

**2. Colored Light Bulbs (Yellow to Pink):**

* Each finger count (gesture) is also mapped to a **colored light bulb** to provide another form of visual feedback. The color of the light bulb changes based on the gesture:
  + **1 Finger → Yellow Light**
  + **2 Fingers → Green Light**
  + **3 Fingers → Blue Light**
  + **4 Fingers → Red Light**
  + **5 Fingers → Pink Light**

The color change in the light bulbs makes it easy to quickly identify the recognized gesture at a glance, providing an additional visual cue to caregivers or the patient.

**References:**

**1.ESP32 Microcontroller:** [**https://randomnerdtutorials.com/getting-started-with-esp32/**](https://randomnerdtutorials.com/getting-started-with-esp32/)

**2.ESP32- CAM tutorial:** [**https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/**](https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/)

**3.Upload Code in ESP32 CAM:** [**https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb/**](https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb/)

**4.Basics of Python:** [**https://www.geeksforgeeks.org/python-basics/**](https://www.geeksforgeeks.org/python-basics/)

**5.Machine learning:** <https://www.geeksforgeeks.org/machine-learning/>