Hand Gesture Home Automation System (for Asylum)

Department of Computer Science and Engineering

National Institute of Technology Rourkela ,Odisha 769008

Team Members

Sambit Kumar Rout (121CS0216)

Md Nafis Al Safayet (121CS0217)

Amit Kumar Sah: (121CS0218)

Priyanka Kumari Sah: (121CS0219)

Adya Ranjan Sahoo: (121CS0220)

Table of Contents

Introduction:

The IoT-Controlled Hand Gesture Recognition System integrates computer vision, hand gesture recognition, and the MQTT protocol to control IoT devices. Using hand gestures, users can toggle LEDs and RGB lights in real time, enhancing user experience in home automation. This report delves into the integration of an MQTT broker, hand gesture recognition, and the operation of a microcontroller as an IoT device.

Implementation Process:

1. Publisher Setup

- **Purpose:** The publisher sends messages to the MQTT broker based on the input provided via a file.
- **Steps:**
    1. Configure the MQTT client using the `paho-mqtt` library, setting up the broker address, port, and topic.
    2. Implement file monitoring:
        - Continuously check a text file (`sample.txt`) for new messages.
        - Clear the file after reading to avoid duplicate messages.
    3. Publish the read message to the MQTT topic using the `publish` method of the MQTT client.
    4. Handle errors such as file not found or broker disconnection gracefully.
    5. Ensure the loop is running asynchronously to support real-time updates.

2. Hand Gesture Detection

- **Purpose:** Detect specific hand gestures using a webcam and map them to corresponding commands.
- **Steps:**
    1. Use the **MediaPipe Hands** library to detect hand landmarks in real time.
    2. Capture live video feed from the webcam and process each frame:
        - Flip the frame for a mirrored view.
        - Convert the frame to RGB for accurate detection.
    3. Identify gestures:
        - Use fingertip coordinates to check which fingers are raised.
        - Recognize predefined gestures like **Thumb**, **V-sign**, or **Spider-Man** using custom logic.
    4. Map gestures to specific commands (e.g., "Disco On," "Bulb Off").
    5. Write the detected command to the `sample.txt` file for the publisher to pick up.
    6. Display feedback on the screen, such as the gesture name and finger count.
    7. Optimize for performance using FPS calculation and efficient frame handling.

3. Subscriber Configuration

- **Purpose:** The subscriber listens to messages from the MQTT topic and performs actions on connected devices.
- **Steps:**
    1. Connect to the MQTT broker using the `umqtt.simple` library.
    2. Subscribe to the specified topic (`chat/messages`) to receive updates.
    3. Implement a callback function to handle incoming messages:
        - Decode the message payload and map it to corresponding actions.
        - For example:
            - **"Disco On"**: Turn on the RGB LED.
            - **"Bulb Off"**: Turn off the simple LED.
    4. Control GPIO pins on the microcontroller:
        - Define separate pins for a simple LED and RGB LEDs.
        - Set the pin states (`HIGH` or `LOW`) based on the message.
    5. Ensure Wi-Fi connectivity for the device using the `network` module.
    6. Continuously listen for messages in a loop while handling potential interruptions or disconnections.
    7.

Integration of Components

1. **Publisher and Gesture Detection:**
    - The hand gesture detection module writes commands to a shared file (`sample.txt`).
    - The publisher reads these commands and sends them to the MQTT broker.
2. **Publisher and Subscriber:**
    - The subscriber listens to the MQTT topic and processes commands published by the publisher.
3. **Real-Time System:**
    - All components work in sync to enable gesture-based control of IoT devices. The overall latency is minimized through efficient MQTT communication and optimized gesture detection.

Background:

Home automation systems are increasingly incorporating gesture-based controls to improve convenience and accessibility. Using the MQTT protocol for this system allows efficient, low-latency data communication between devices, while hand gesture recognition leverages computer vision to enable intuitive interactions without physical contact.

Motivation:

This project aims to develop a non-contact home automation control system that allows users to control lights through hand gestures. By combining the MQTT protocol and IoT capabilities, this system can be remotely monitored and controlled, providing users with greater flexibility and functionality in managing home environments.

Objectives:

1. To implement an MQTT-based communication system for efficient message passing between devices.
2. To create a hand gesture recognition system using a webcam for identifying specific gestures.
3. To control RGB and LED lights on an IoT device based on recognized gestures.
4. To demonstrate the real-time, bi-directional communication between gesture recognition and an IoT-controlled environment.

Methodology

1. **MQTT Publisher (PC-Based)**
    - A Python script uses the Paho MQTT library to connect to the broker and send messages to a predefined topic.
    - Upon recognizing a hand gesture, the system writes a command to a text file, which is subsequently published to the MQTT broker.
2. **Hand Gesture Detection**
    - The Mediapipe library captures hand movements from a webcam feed and identifies distinct gestures by examining finger positions.
    - Each detected gesture corresponds to a command (e.g., turning on/off lights) that is published to the MQTT broker.
3. **MQTT Subscriber (IoT Device)**
    - The microcontroller (e.g., Raspberry Pi Pico or ESP32) subscribes to the topic and listens for commands.
    - Upon receiving a command, it activates/deactivates LEDs or RGB lights based on the published message.

Analysis and Findings

The system demonstrates successful integration of real-time gesture recognition with MQTT-based communication:

- **Real-Time Communication**: The MQTT broker facilitates seamless message exchange with minimal latency.
- **Gesture Recognition Accuracy**: The use of Mediapipe ensures reliable gesture recognition, with the system accurately identifying unique gestures for specific actions.
- **Device Response**: The microcontroller responds almost instantly to the commands, toggling the lights per the detected gestures.

Challenges

1. **Latency in Network Communication**: Occasionally, there was slight delay in message delivery due to network variability.
2. **Gesture Recognition Reliability**: Ambient lighting and hand positioning affected gesture detection accuracy, requiring fine-tuning.
3. **File Handling in Publisher**: Ensuring that the file was correctly updated and read without causing conflicts required careful management of read/write operations.

Conclusion and Future Work

The system successfully enables gesture-based control over IoT devices through MQTT. Future enhancements could include more gesture types, additional device support, and improved algorithms to handle varying lighting conditions.

Future Work:

- Expand to control additional home automation devices.
- Implement machine learning for gesture customization.
- Add voice control as an additional interaction method.

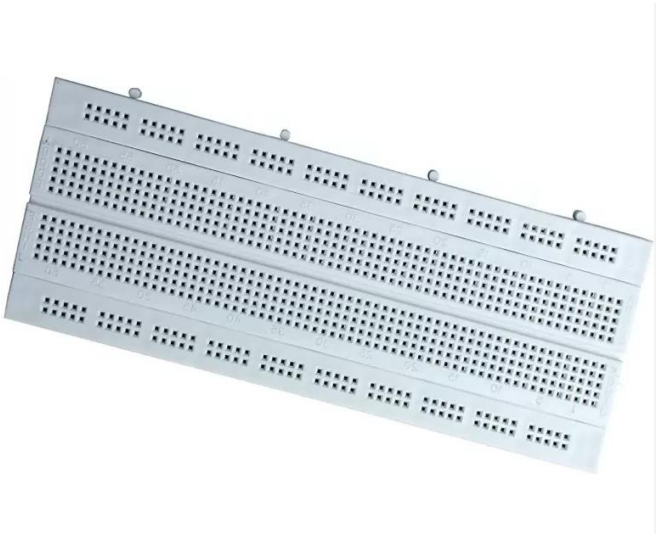
Additional Details

- Broker Details: Broker address – broker.hivemq.com, Port – 1883
- **Gesture Commands**: “Disco on,” “Disco off,” “Bulb on,” and “Bulb off” for light control.
- **Programming Language**: Python for publisher and gesture detection, MicroPython for subscriber.

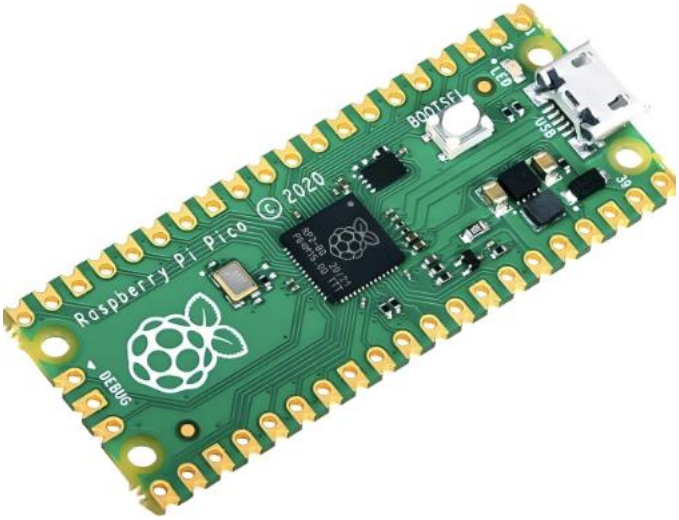
Devices used in the project:

1. Laptop with Webcam

2.Breadboard

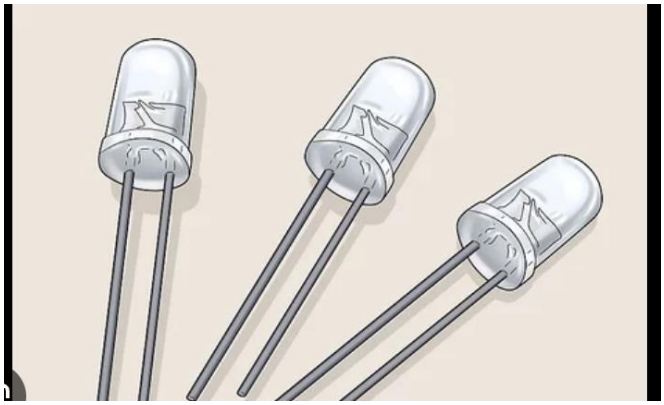


3.Rasperry pi pico

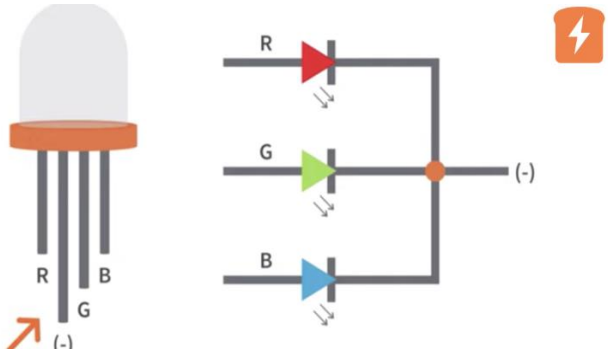


4.Wires

5.LED light

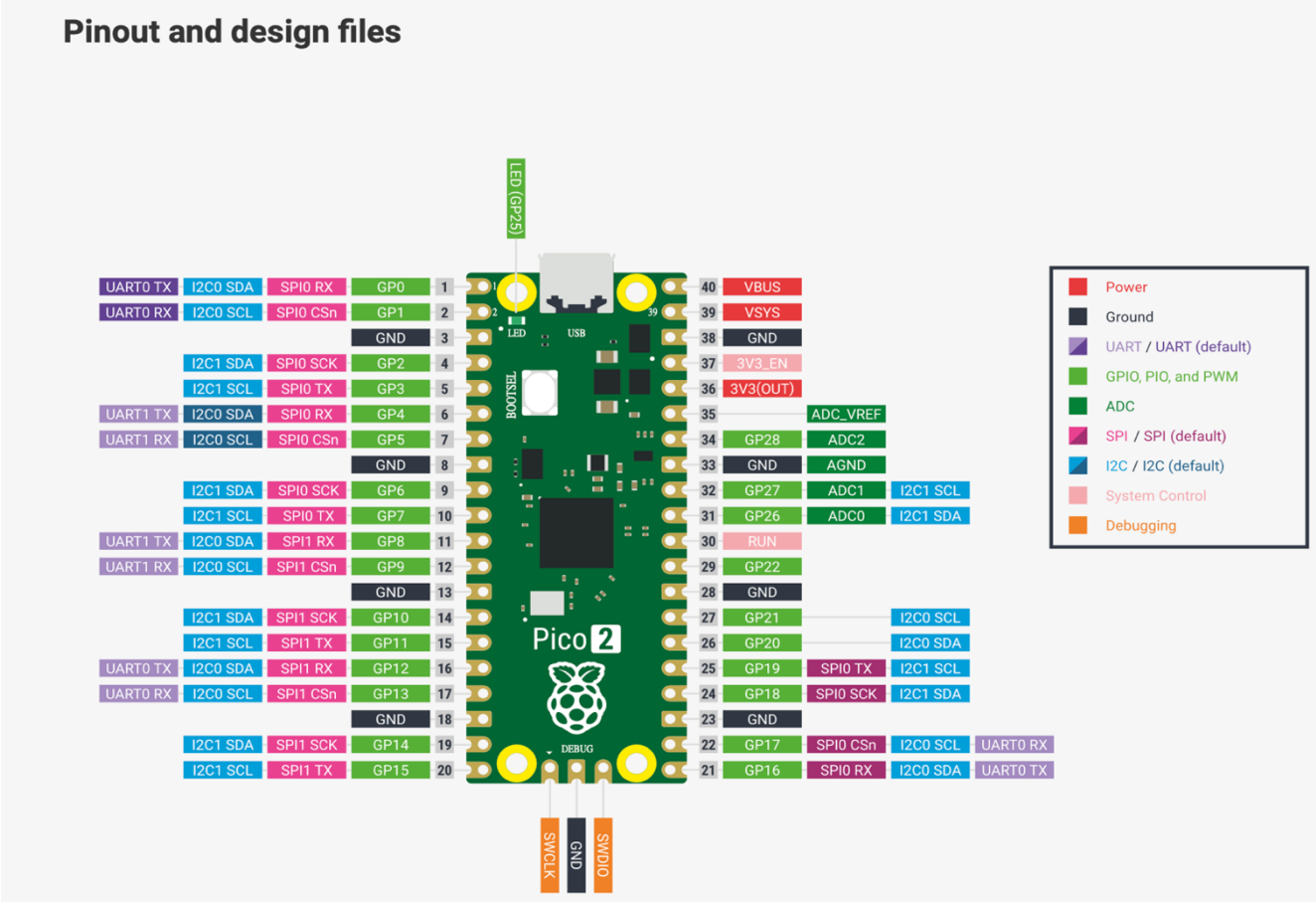


6.RGB light

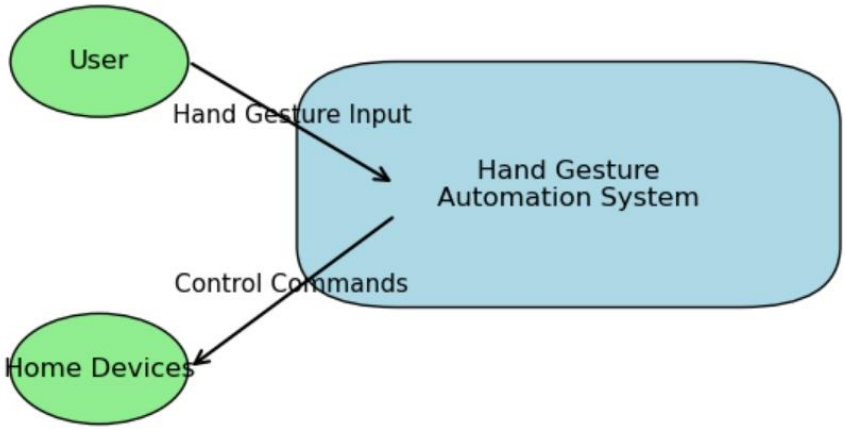


7.Wifi (Mobile)

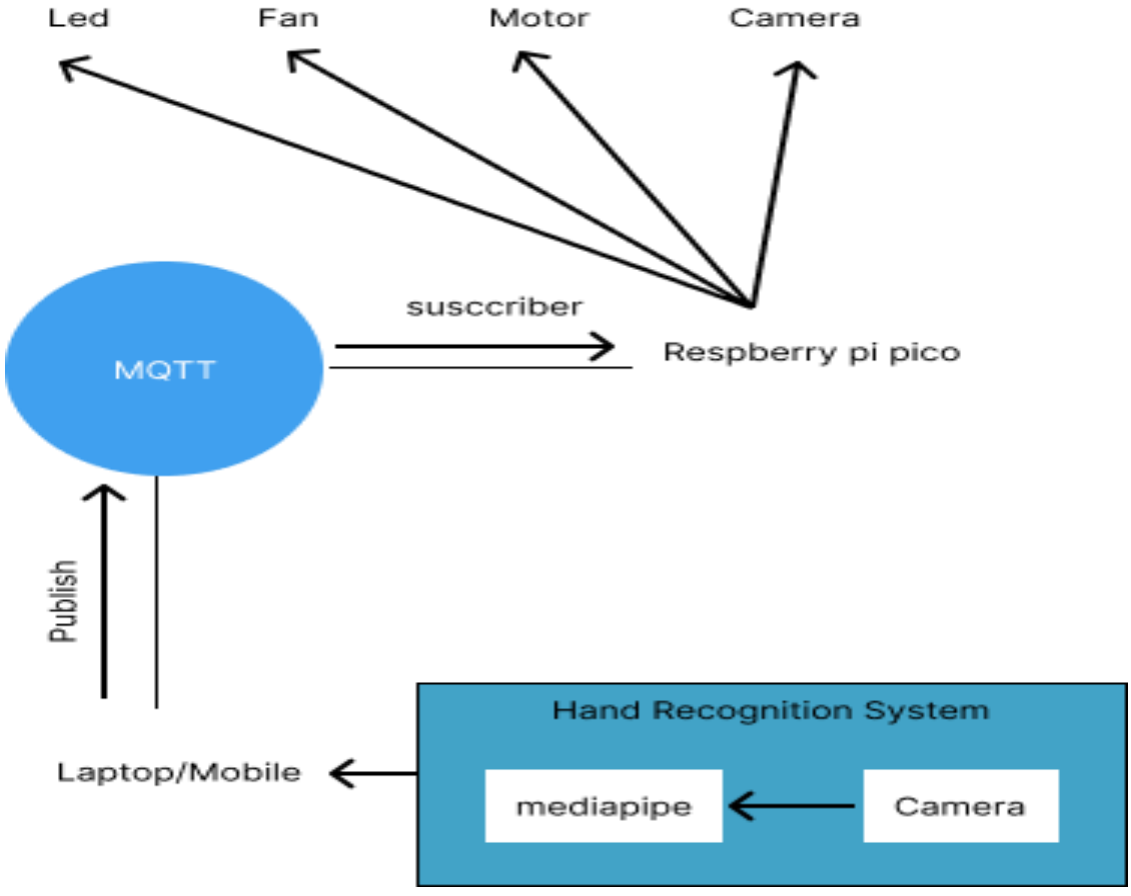
Circuit diagram of Rasperry pi pico



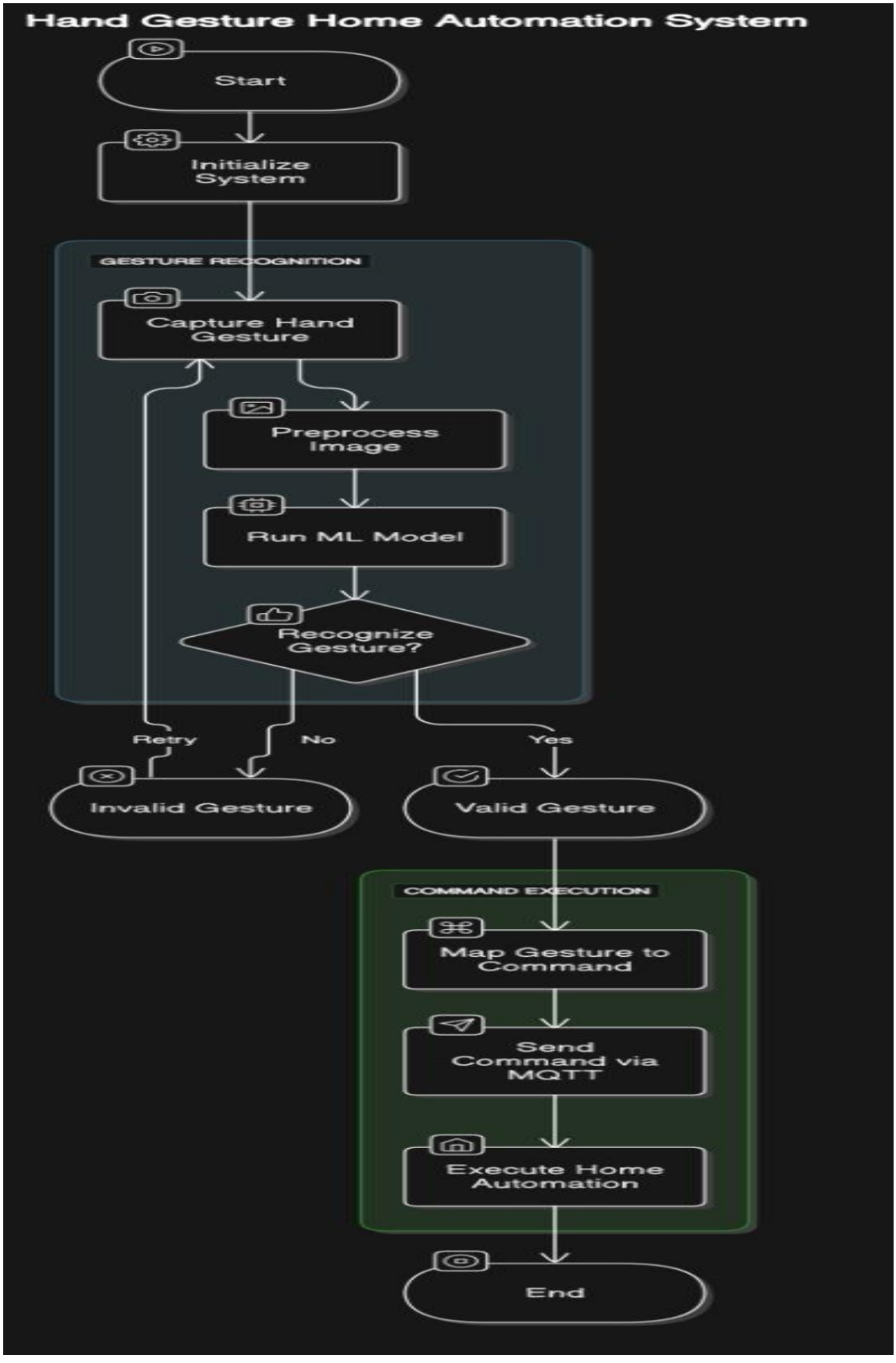
DFD Level 0: Hand Gesture Automation System



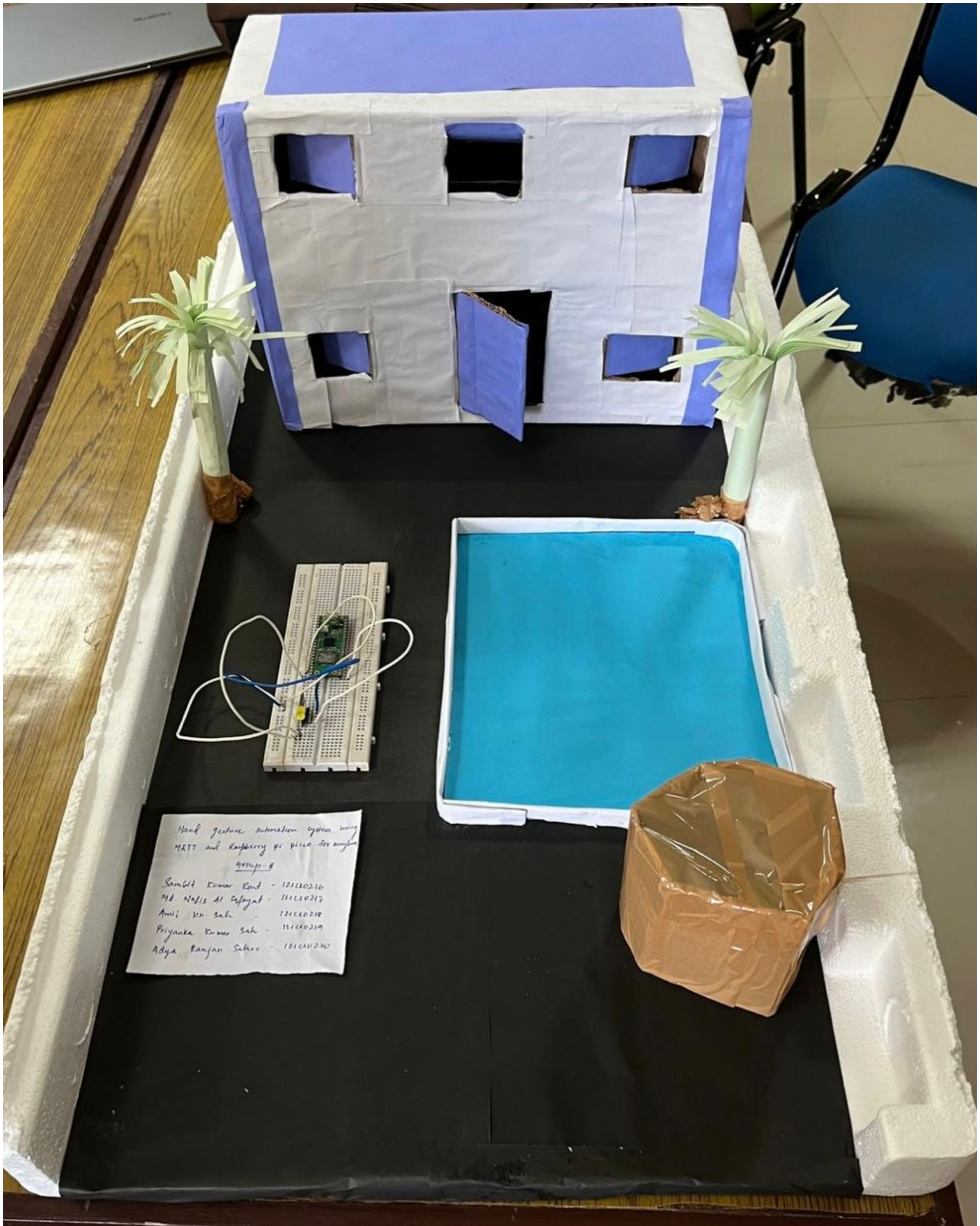
DFD Level 1:



Control Flow Diagram



Final Model


Hand gesture automation system using
Priyanka Kumar Sah
Adya Ranjan Sahoo