0.1 System Monitor

0.1.1 Objective and Functionality

The System Monitor is a pivotal component in a biometric data-based framework for user access and verification. It serves two main functions: position detection using user location data and user identification. The System Monitor's functionalities include:

- Subscribing to and processing location data from two ESP-01 modules with ultrasonic sensors via MQTT.
- Displaying user location and verification data on an LCD screen.
- Visually representing user proximity using an LED bar.

0.1.2 Project Definition and Milestones

The development of the System Monitor involved the following milestones:

1. Establishing MQTT communication to receive data from ultrasonic sensors.
2. Integrating and configuring the LED bar and LCD with the ESP32.
3. Developing and testing the software for proximity detection and data display.
4. Efficient multitasking and task synchronization within FreeRTOS.

0.1.3 Achieved milestones, execution order, priority, and dependencies

1. **Milestone 1: Establishing MQTT Communication** - *Priority:* High. Fundamental for data transmission. - *Dependencies:* Basic WiFi setup. - *Execution Order:* First, as it is crucial for data reception.
2. **Milestone 2: LED Bar and LCD Integration** - *Priority:* Medium. Important for user interface. - *Dependencies:* Successful MQTT setup. - *Execution Order:* Second, building upon established communication.
3. **Milestone 3: Software Development** - *Priority:* High. Essential for functionality. - *Dependencies:* Functional hardware setup. - *Execution Order:* Third, focusing on processing and display.
4. **Milestone 4: Multitasking and Task Synchronization** - *Priority:* High. Critical for system reliability. - *Dependencies:* Completion of initial software development. - *Execution Order:* Fourth, finalizing the system integration.

0.1.4 Hardware Development

The hardware setup for the System Monitor comprises:

- An ESP32 microcontroller connected to a breadboard.
- An LED bar and an LCD screen interfaced with the ESP32 for display.
- Two ESP-01 modules, each with an ultrasonic sensor, for distance measurement.

0.1.5 Software Implementation

The software, written in C++, leverages FreeRTOS for effective multitasking. Key functions include:

- `WiFiTask` for managing WiFi connectivity.
- `MQTTTask` for handling MQTT communication.
- `LCDDisplayTask` and `checkForPresenceAndDirection` for controlling the display based on sensor data.
- Semaphore (`xSemaphoreLCD`) to ensure safe LCD access.

0.1.6 Interaction with Other Components

The System Monitor interacts with the ESP-01 modules to determine the user's location based on distance data, which is then displayed on the LED bar and LCD screen.

0.1.7 Testing

Testing ensured accurate sensor data transmission via MQTT and responsive displays on the LED bar and LCD. An iterative approach was applied to refine the system's performance.

0.1.8 Dedication Time

Approximately 20 hours were dedicated to developing the System Monitor, focusing on both hardware assembly and software programming.

0.1.9 User Story

As a user approaches, the System Monitor detects their presence, lighting up the LED bar and displaying their approximate location (left, right, or center) on the LCD screen, providing real-time feedback.

0.1.10 Challenges and Solutions

- **Hardware Challenges:** Complex wiring and space constraints on the breadboard, along with troubleshooting LED issues.
- **Software Challenges:** Establishing reliable communication and processing sensor data accurately. Achieved through careful coding and testing.

0.1.11 Hardware and Software Integration

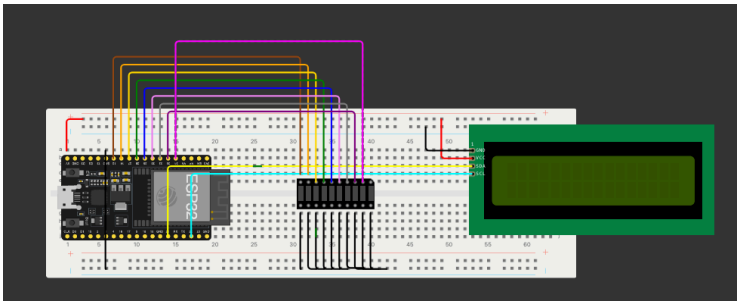

Figure 1: ESP32 with LED bar and LCD display connected on a breadboard.

0.1.12 Microcontroller Interaction Protocol

The System Monitor project involves coordinated communication between multiple microcontrollers, primarily the ESP32 and two ESP-01 modules. The communication is structured as follows:

Communication Protocol The project utilizes the MQTT (Message Queuing Telemetry Transport) protocol, a lightweight and efficient messaging protocol ideal for IoT applications. This protocol is chosen for its low bandwidth usage and its ability to provide reliable communication over WiFi.

WiFi Configuration Each microcontroller in the system is configured to connect to a WiFi network, enabling them to send and receive data over the network. The network credentials (SSID and password) are programmed into the microcontrollers.

Data Flow and Configuration The ESP-01 modules, each equipped with an ultrasonic sensor, collect distance data and publish this information to specific MQTT topics. The ESP32, acting as the central unit, subscribes to these topics to receive the data.

0.1.13 MQTT Tree Structure

The MQTT protocol in the System Monitor project uses a structured approach to manage the data flow. Below is the outline of the MQTT topics and their functions:

MQTT Topics

- **sensor1/distance:** This topic is used by the first ESP-01 module. It publishes the distance data measured by its connected ultrasonic sensor. The ESP32 subscribes to this topic to receive updates on the user's distance from this sensor.
- **sensor2/distance:** Similar to the first, this topic is for the second ESP-01 module. It provides distance data from its sensor, allowing the ESP32 to determine the user's location relative to this second sensor.

Data Organization and Utilization The data sent to these topics include numerical values representing the measured distances. The ESP32, upon receiving this data, processes it to determine the user's proximity and location. Based on these calculations, the ESP32 then controls the LED bar and the LCD display to provide real-time feedback about the user's position.