Industrial Attachment – GizanTech

Final Report

Team – 03

Team Members-

Shah Ahmed Raad_2003042

Zabed Iqbal Chowdhury_2003011

Contents

Final Report

Smart Fingerprint Attendance System

Introduction

This project focuses on developing a biometric attendance system that leverages an ESP32 microcontroller and a fingerprint sensor to register and authenticate employees. It incorporates a database system for storing fingerprint records (Offline when WIFI is not available or connected, online when WiFi is available or connected) and enables Over-the-Air (OTA) updates to ensure easy maintenance and scalability. The system is designed to enhance security and streamline attendance tracking with minimal manual intervention.

Project Objectives

- Develop an efficient fingerprint recognition system using an ESP32 and a compatible sensor.
- Store and manage employee data using SPIFFS and MySQL for secure and persistent record-keeping.
- Implement OTA update functionality to facilitate remote firmware upgrades without physical access.

Hardware Components

- ESP32 Microcontroller: The central unit responsible for processing and communication (Wi-Fi/Bluetooth).
- R307 Fingerprint Sensor: Captures and verifies employee fingerprints using a UART connection.

Pin Configuration:

- RXD2 (Pin 16) → TX (Fingerprint Sensor)
- TXD2 (Pin 17) → RX (Fingerprint Sensor)
- Vin & GND → Power Supply (5V & Ground)

Hardware Specifications

R307 Fingerprint Sensor

- Supply Voltage: 3.6V - 6.0V
- Current Consumption: 120mA max
- Fingerprint Capture Time: <1 second
- Storage Capacity: 162 templates
- Interface: TTL Serial (Default Baud Rate: 57600)
- Environmental Range: -20°C to +50°C, 40%-85% RH

ESP32 Development Board

- Model: ESP32 CP2102
- Flash Memory: 32Mbit
- GPIO Ports: 22
- Wi-Fi: 2.4GHz (802.11 b/g/n)
- Bluetooth: BLE 4.2
- Security: WPA/WPA2 Encryption

Software & Development Tools

- Development Environment: Arduino IDE
- Languages: C, PHP, HTML
- Key Libraries:
  - ✓ Adafruit_Fingerprint (Fingerprint sensor interface)
  - ✓ ArduinoHttpClient (Handles OTA and MySQL connectivity)
  - ✓ Arduino_JSON (Parses and structures JSON data)
  - ✓ WebSockets (Supports real-time communication)

Testing & Validation

Test Case	Feature	Scenario	Result	Status	Feedback
TC-01	Sensor Detection	Ensure sensor is recognized by ESP32	Detected successfully	Pass	Initialization ~1 sec
TC-02	Fingerprint Enrollment	Register and store three fingerprints	Successful registration	Pass	Smooth operation
TC-03	Data Retrieval	Match fingerprint with stored records	Data retrieved in <2 sec	Pass	Working fine
TC-04	OTA Update	Perform firmware update over Wi-Fi	Update completed without errors	Pass	Minimal disruption
TC-05	Power Stability	Verify data persistence after reboot	Data remained intact	Pass	No data loss
TC-06	MySQL Integration	Attempt database connection via XAMPP	Failed due to network issue	Fail	Needs further debugging
TC-07	Stress Testing	Handle multiple scans and updates	Stable under load	Pass	No crashes detected

Guide to Setting Up the Fingerprint Sensor System Development Environment

1. ### Installing Arduino IDE

   The Arduino IDE is essential for programming the ESP32 and uploading firmware to the Fingerprint Sensor System.

   - ✓ Step 1: Visit the official website and download the Arduino IDE: [Arduino IDE Download].
   - ✓ Step 2: Run the installer (arduino-{version}-windows.exe) and follow the setup wizard to install the IDE.
   - ✓ Step 3: Open the Arduino IDE and configure the ESP32 board:
     - Go to File > Preferences.
     - In the “Additional Boards Manager URLs” field, paste this URL: https://dl.espressif.com/dl/package_esp32_index.json.

- Navigate to Tools > Board > Boards Manager, search for “ESP32,” and install the package by Espressif Systems.

✓ Step 4: Choose the correct board:

- Go to Tools > Board > ESP32 Arduino > ESP32 Dev Module.

✓ Step 5: To verify setup, upload a simple sketch (such as the Blink example) to the ESP32 over USB.

2. Installing XAMPP

XAMPP is used to host a MySQL database for integration with the ESP32, as planned for Day 4.

✓ Step 1: Download XAMPP from the official website: [XAMPP Download].

✓ Step 2: Run the installer (xampp-windows-x64-{version}-installer.exe) and proceed with the default settings, ensuring Apache and MySQL are selected.

✓ Step 3: Open the XAMPP Control Panel and start both Apache and MySQL modules.

✓ Step 4: To access phpMyAdmin, open a web browser and go to http://localhost/phpmyadmin.

✓ Step 5: Create a new database for employee records:

- In phpMyAdmin, click New, name the database (e.g., employee_db), and create a table with fields like id, name, and fingerprint_id.

✓ Step 6: Record MySQL server details (host: localhost, user: root, password: [default empty]) for use when connecting with ESP32.

3. Setting Up Libraries and Boards in Arduino IDE

To develop the Fingerprint Sensor System firmware, certain libraries and boards need to be installed.

- ESP32 Board Setup: This was completed in Step 2 (Arduino IDE setup). Ensure you have selected the right board (ESP32 Dev Module) under Tools > Board.

- Installing Libraries:
    - Open Sketch > Include Library > Manage Libraries to access the Library Manager.
    - Install the following libraries:
        - Adafruit_Fingerprint by Adafruit: For communication with the fingerprint sensor.
        - ArduinoHttpClient by Arduino: Supports HTTP requests for OTA updates and MySQL integration.
        - Arduino_JSON by Arduino: For parsing and serializing JSON data for employee records.
        - WebSockets by Markus Sattler: Enables real-time communication for OTA updates.
    - After installation, test each library by adding it to a sample sketch (e.g., #include <Adafruit_Fingerprint.h>).

Fingerprint Sensor System User Manual

1. **Setting Up the Arduino IDE on Your Computer Arduino IDE controls the ESP32 and the fingerprint sensor for the system.**

   Open the Arduino IDE (with its blue icon and white infinity symbol). You'll see a code editor window.

2. **Selecting the ESP32 Board in Arduino IDE**
   The ESP32 is the core device running the system. To get Arduino IDE to work with it, follow these steps:

   ✓ Step 1: In the Arduino IDE, go to File (top left) > Preferences.

   ✓ Step 2: In the "Additional Boards Manager URLs" field, paste: https://dl.espressif.com/dl/package_esp32_index.json, then click "OK."

   ✓ Step 3: Go to Tools > Board > Boards Manager, search for "ESP32," and click "Install" for the "esp32 by Espressif Systems" package.

- ✓ Step 4: To select your ESP32 board, navigate to Tools > Board > ESP32 Arduino > ESP32 Dev Module.

3. **Connecting the ESP32 and Fingerprint Sensor**
   Now, connect the hardware so the ESP32 can communicate with the fingerprint sensor.

   - ✓ Step 1: Gather the necessary items: ESP32 board, fingerprint sensor, USB cable, and connecting wires.
   - ✓ Step 2: Wire the fingerprint sensor to the ESP32 as follows:
     - TX (sensor) → Pin 16 (RX on ESP32).
     - RX (sensor) → Pin 17 (TX on ESP32).
     - VCC (sensor) → 3.3V (ESP32).
     - GND (sensor) → GND (ESP32).
   - ✓ Step 3: Plug the ESP32 into your computer using the USB cable. A light should turn on, indicating power.

4. **Enrolling a Fingerprint**
   Enroll your fingerprint in the system for recognition.

   - ✓ Step 1: Open the Arduino IDE and paste the project code (provided by your instructor or team).
   - ✓ Step 2: Click the green “Upload” button (right arrow), and wait until “Done uploading” appears.
   - ✓ Step 3: Open the Serial Monitor: Go to Tools > Serial Monitor, and set the baud rate to “115200” (bottom right).
   - ✓ Step 4: Look for the “Found fingerprint sensor!” message to ensure the sensor is operational.
   - ✓ Step 5: Enter an ID number (1 to 127) when prompted.
   - ✓ Step 6: Follow the on-screen instructions to place your finger on the sensor twice. The sensor will blink or light up.

- ✓ Step 7: When you see “Fingerprint enrolled successfully!” your fingerprint has been saved with the chosen ID (e.g., ID 1).

5. **Storing Fingerprint Data in Local Memory (SPIFFS)**
   The system saves your fingerprint ID and details (e.g., name) to the ESP32’s using SPIFFS.

   - ✓ Step 1: After enrolling, the Serial Monitor will prompt for your name. Enter your name (e.g., “John”) and press Enter.
   - ✓ Step 2: The system automatically saves your fingerprint ID and name to memory (SPIFFS). A message will appear concerning the operation being successful or not.
   - ✓ Step 3: To check the enrollment, scan your fingerprint again. The Serial Monitor should display “Found ID 1: John” if the fingerprint matches.

6. **Performing an Over-the-Air (OTA) Update**
   OTA allows you to update the system without unplugging any hardware, using Wi-Fi.

   - ✓ Step 1: Ensure the ESP32 is powered on and connected to your computer.
   - ✓ Step 2: Connect the ESP32 to your Wi-Fi by entering the SSID and password when prompted in the Serial Monitor.
   - ✓ Step 3: The Serial Monitor will show the ESP32’s IP address (e.g., “192.168.1.100”). Note this IP address.
   - ✓ Step 4: In a web browser, enter the IP address (e.g., http://192.168.1.100) and press Enter.
   - ✓ Step 5: A webpage with an “Update” button will appear. Click “Choose File,” select the new firmware (provided by your instructor), and click “Update.”
   - ✓ Step 6: Wait for the update to complete. The Serial Monitor will display “Update successful!” and the ESP32 will restart.

7. **Troubleshooting Tips**

   - ✓ Sensor Not Found: Double-check the wiring to ensure everything is connected correctly.
   - ✓ Wi-Fi Not Connecting: Verify that your Wi-Fi credentials are accurate and that both the ESP32 and computer are on the same network.
   - ✓ Fingerprint Not Enrolling: Ensure your finger is placed firmly on the sensor. If the sensor is dirty, clean it with a soft cloth.
   - ✓ OTA Fails: Make sure the Wi-Fi signal is stable. If issues persist, restart the ESP32 and try again.

Conclusion

The Fingerprint Sensor System project was an enriching experience that demonstrated the impact of hardware development in a practical context. Starting from the ground up on Day 1, we successfully developed a working prototype by Day 6, combining a fingerprint sensor with an ESP32, managing data storage, and enabling OTA updates. The support from the CTO, CEO, and CRM, alongside practical troubleshooting, helped us navigate obstacles like sensor compatibility and network issues.

This journey not only improved my skills in microcontroller programming and hardware integration but also taught me the value of collaboration and persistence. I'm proud of our accomplishments and excited to apply these insights to future hardware projects, potentially delving deeper into IoT or biometric systems. A huge thank you to GizanTech for this amazing opportunity!