UNIVERSIDAD POLITÉCNICA DE YUCATÁN

BIS Universities

Teacher: Jorge Javier Pedrozo

Student: Noh Uc Juan Said

Canul Gio Olga Nataly

Pérez Larrache Mario Alberto

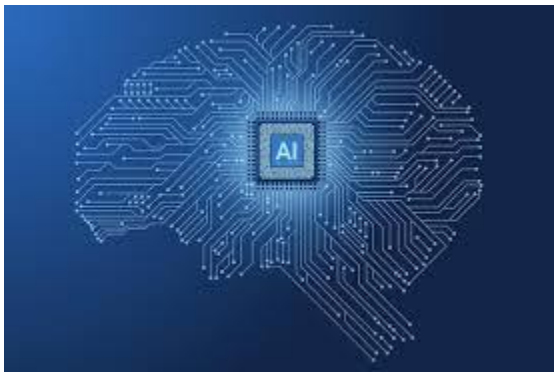
Sánchez Villanueva Jorge Emmanuel

Cámara Sierra Julián Jesús

Group : 3-A

Embedded system

Programm: Advanced Programmin



Technical Architecture Document

The Smart Home Automation System is a project that integrates a Flutter mobile application with an ESP32 microcontroller through Bluetooth communication. Its purpose is to allow users to control home devices, monitor environmental data, and automate everyday actions. The system combines mobile development, embedded programming, and real-time communication to create a complete smart home solution.

The general architecture of the system is composed of two main parts: the mobile application and the embedded hardware system. The Flutter app provides the interface and logic for device management, automation rules, and data visualization. The ESP32 acts as the central controller that communicates with the app, executes commands, and sends sensor data. Both components communicate through Bluetooth using a custom JSON-based protocol that ensures efficient data transfer.

In the hardware setup, the ESP32 is connected to several components: a DHT22 sensor to measure temperature and humidity, an LDR sensor for light intensity, LEDs to simulate lighting control, and two servo motors that represent mechanical systems such as doors or windows. Power is supplied through a breadboard setup with resistors for current control. The system was designed with modularity in mind to make it easy to add more sensors or actuators in the future.

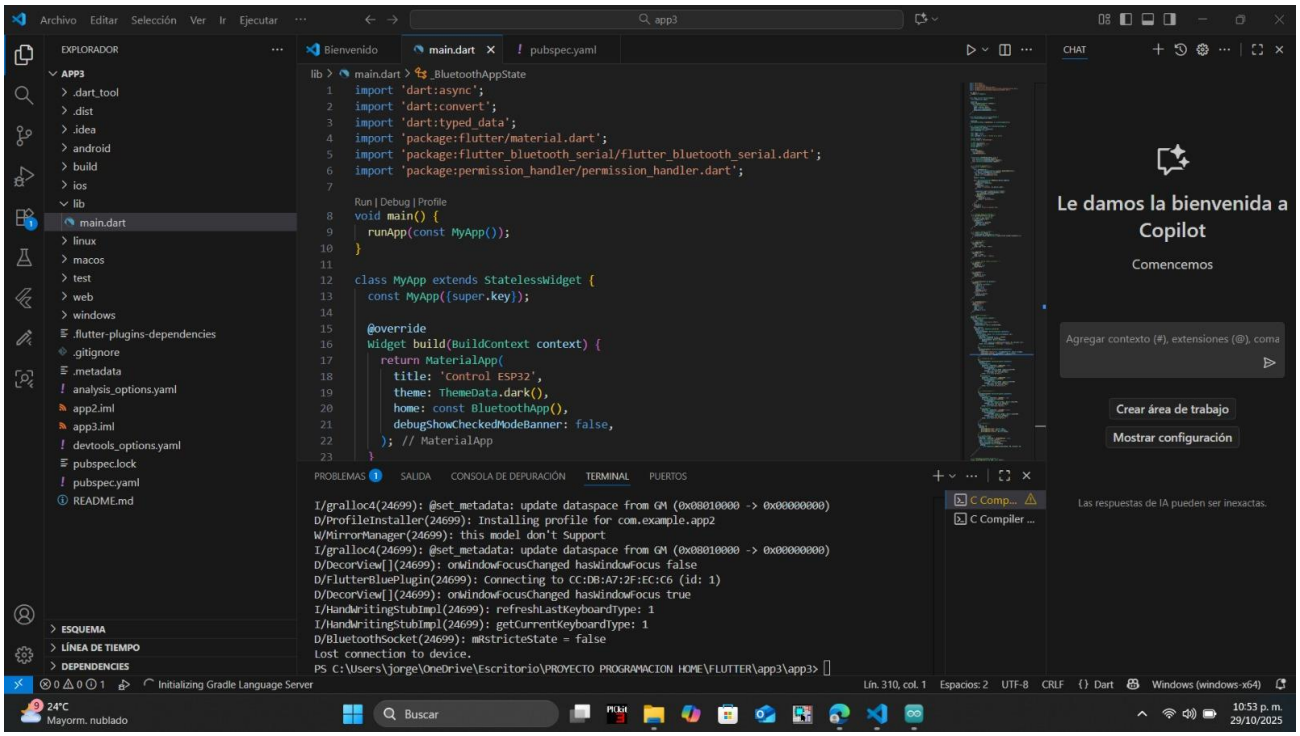
On the software side, the mobile application was developed using Flutter and Dart. The Provider pattern was used for state management to ensure clean separation between the user interface and the logic. SQLite serves as the local database, allowing the app to store user preferences, schedules, and sensor readings even when offline. The flutter_bluetooth_serial package enables communication between the phone and the ESP32, and shared_preferences is used for quick access to user configurations.

The overall architecture ensures scalability and maintainability by separating responsibilities: the UI handles interaction, the provider layer manages state, and the data and communication layers handle storage and device control.

API and Protocol Documentation

The system uses JSON-formatted messages sent over Bluetooth to communicate between the Flutter app and the ESP32. Each command includes a keyword that identifies the action and any additional parameters needed to execute it. The ESP32 parses the command, performs the requested operation, and sends back a JSON response.

This structure ensures that communication is clear, structured, and easy to expand if more devices are added in the future.



Code Documentation

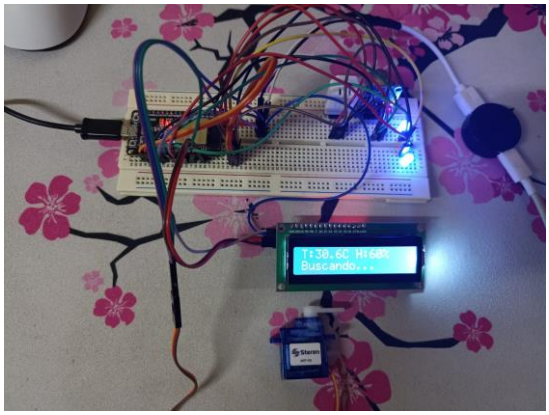
The Flutter project is organized into several folders for maintainability. The “models” folder defines data models for devices and sensor readings. The “providers” folder handles the logic and app state using the Provider pattern. The “services” folder includes the Bluetooth connection manager and the database helper classes. The “screens” folder contains the main user interface pages such as the dashboard, device control page, and settings. The “widgets” folder holds reusable UI components, and the “database” folder handles SQLite initialization and queries.

On the embedded side, the ESP32 code is written in Arduino C++. The BluetoothSerial library is used to establish the connection with the mobile app. The setup function initializes Bluetooth, sensors, and actuators, while the loop continuously listens for incoming messages. When a command is received, the code parses the JSON message and performs the corresponding action, such as turning an LED on, reading temperature, or moving a servo.

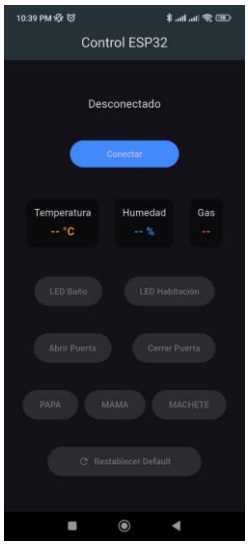
User Manual

To use the Smart Home Automation System, the user needs an Android smartphone with Bluetooth enabled and the Smart Home mobile app installed. The ESP32 circuit must be powered through USB or an external power supply.

- Installation and Setup
- Install the APK of the Smart Home app on the Android device.
- Turn on the ESP32 circuit.



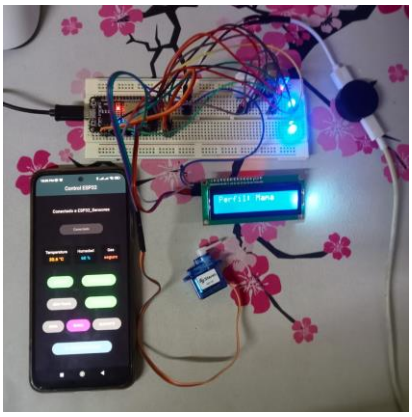
Open the app and pair with the device named “SmartHomeESP32.”



Once connected, the dashboard will display available devices and sensors.

How to Operate

From the main dashboard, the user can turn LEDs on or off, move servo motors, and view real-time sensor readings for temperature, humidity, and light levels. Automation rules can be set to trigger devices at specific times or conditions. The app also stores data locally, allowing users to view logs and analytics even when offline.



Troubleshooting

If the app cannot find the ESP32, ensure Bluetooth is enabled and the device is within range. If no sensor data appears, verify the wiring of the sensors. If the app crashes or shows connection errors, grant Bluetooth and location permissions and restart the app.

Testing and Results

The project was tested through unit and integration tests to verify stability and performance. Unit tests in Flutter confirmed that Bluetooth connections could be established, LEDs could be toggled, and sensor data was retrieved correctly. Database tests ensured that user data and automation schedules were stored and retrieved without issues.

Integration tests confirmed successful communication between the Flutter app and the ESP32. Commands were sent and executed reliably, and responses were received in real time. Automation schedules worked as expected, triggering devices at the defined times. The system also handled invalid commands gracefully without crashing.

User testing was conducted with several participants, who reported that the app interface was intuitive, responsive, and visually appealing. Bluetooth communication remained stable throughout the tests, and sensor readings updated accurately.

Conclusion

The Smart Home Automation System successfully demonstrates the integration of a Flutter mobile application with an ESP32 microcontroller using Bluetooth communication. The project applied clean architecture principles, modular design, and effective state management. Through the use of the Provider pattern, SQLite database, and JSON-based command protocol, the system achieved flexibility and scalability.

This project meets the technical and learning objectives by combining modern mobile development with embedded systems engineering. It provides a functional, user-friendly, and efficient example of how advanced programming can be applied to real-world automation problems.