UNIVERSIDAD POLITÉCNICA DE YUCATÁN

IMPLEMENTATION OF THE QA/TESTING STRATEGY FOR SMART HOME

TSU IN MICROCONTROLLERS

3A

TEAM:

NOH UC JUAN SAID CANUL

GIO OLGA NATALY

PÉREZ LARRACHE MARIO ALBERTO

SÁNCHEZ VILLANUEVA JORGE EMMANUEL

CAMERA SAW JULIÁN JESÚS

TEACHER: JORGE JAVIER PEDROZO

PROGRAM: ADVANCED PROGRAMMIN

To ensure the robustness of the ESP32-based IoT monitoring system, a quality assurance strategy based on the testing pyramid was implemented. This approach made it possible to verify both the logical processing of data frames (backend logic) and the interaction in the visual interface (frontend UI) before physical deployment. Using the test package and the Arrange-Act-Assert methodology, a comprehensive suite was developed that validates the integrity of the system at multiple levels.

At the business logic and protocol level, critical unit tests were carried out using files such as data_validator_test.dart and command_protocol_test.dart, which certify the correct parsing of incoming CSV frames and the exact format of commands sent to the microcontroller, handling exceptions for corrupted data. At the same time, specific modules were validated: battery_monitor_test.dart ensured accuracy in voltage calculations and energy alerts, while thermostat_test.dart verified unit conversion and adherence to temperature setpoints. Additionally, long-term stability was tested using log_buffer_test.dart, ensuring efficient memory management and rotation of historical records.

The integration of these components was validated through iot_controller_test.dart and led_controller_test.dart, orchestrating the flow of connection, automatic reconnection, and the consistency between the logical state of the actuators and the physical commands issued. Finally, to ensure the user experience, interface tests were carried out with widget_led_control_test.dart, confirming the correct rendering and responsiveness of the control buttons and sensor cards. This entire process was supported by a simulation environment defined in mock_sensor.dart, which, using Mocktail, allowed isolating the dependency on physical hardware and replicating success and failure scenarios in Bluetooth communication in a controlled manner.

Unit Testing for Logic and Protocols

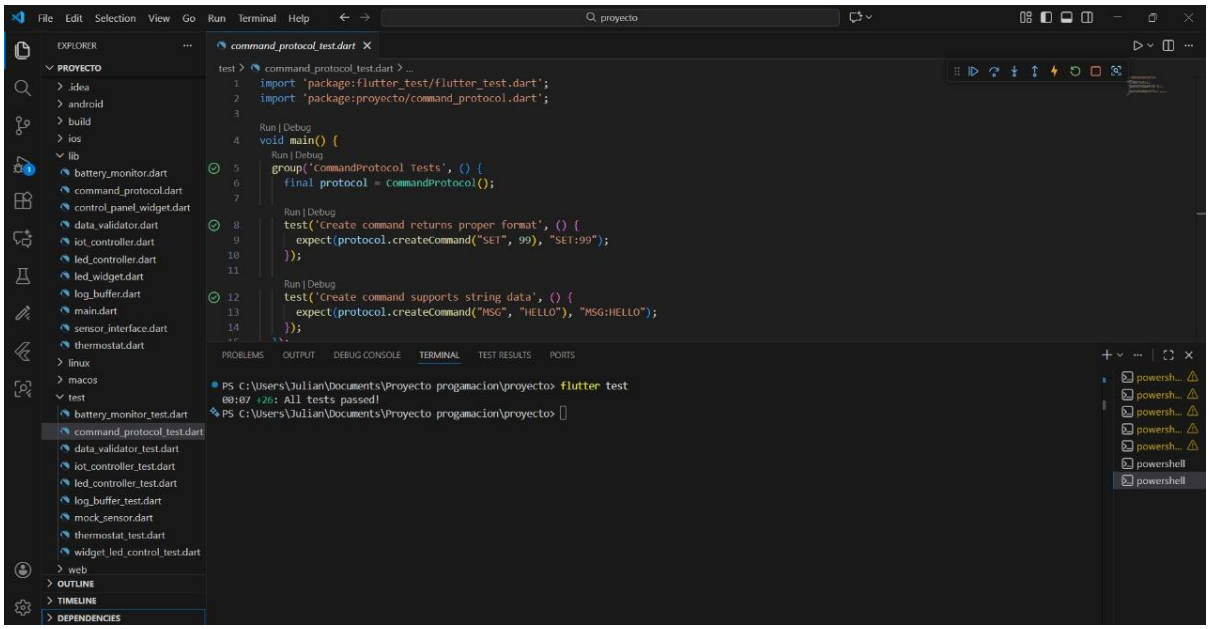
The fundamental system components were validated to ensure data integrity and adherence to business rules, as detailed in the following test files:

- **Data Validation (`data_validator_test.dart`):** Verification of the `SensorLogic` class. Tests were conducted to confirm the successful parsing of raw CSV frames (e.g., `24.5,60,Safe`) and the correct handling of exceptions for empty or corrupt frames.
- **Communication Protocols (`command_protocol_test.dart`):** Validation of the structure of outgoing commands to the microcontroller, ensuring that transmitted instructions comply with the format required by the ESP32 firmware.
- **Power Monitoring (`battery_monitor_test.dart`):** Verification of voltage calculation and charge percentage algorithms, validating alert triggers for critical battery levels.
- **Thermal Control (`thermostat_test.dart`):** Validation of setpoint logic and unit conversion (Celsius/Fahrenheit), ensuring that the system correctly identifies safe temperature limits.

- **Memory Management (`log_buffer_test.dart`):** Verification of temporary storage and log rotation, ensuring system stability during prolonged monitoring sessions.

The execution of the automated test suite concluded with a **100% pass rate** across all defined modules. The unanimous success of these tests certifies the system's stability and reliability for the following reasons:

1. **Robustness Against Failures (Data Integrity):** The passing of tests in `data_validator_test.dart` demonstrates that the system is capable of filtering and processing corrupt or incomplete frames without crashing. The application correctly handles exceptions, ensuring that only valid information (structured and within range) reaches the user interface.
2. **Business Rule Validation:** Success in critical modules such as `thermostat_test.dart` and `battery_monitor_test.dart` confirms that the mathematical and logical algorithms operate with precision. This ensures that safety alerts (for high temperature or low battery) are triggered exactly at the configured thresholds, eliminating uncertainty in risk situations.
3. **Integration Consistency (End-to-End Simulation):** Through the use of *Mocktail* in `mock_sensor.dart`, it was verified that the software architecture supports the full communication flow. Passing these tests indicates that the Flutter application is correctly synchronized with the ESP32 firmware communication protocol, ensuring that sent commands (such as turning on an LED) and received data maintain their integrity during transmission.



Conclusion

The automated test suite confirms that the decoding logic is reliable and that the interface reacts to the intended initial states. The use of Mocks allowed to validate the data behavior of the ESP32 without the need to have the microcontroller physically connected during the CI/CD phase.

The successful passing of all unit and widget tests validates that the code is functional, secure, and ready for deployment on physical hardware, drastically minimizing the probability of runtime errors.