Project Plan Presentation

Zigbee Control Center with Raspberry Pi & ESP32

This project aims to create a local, self-hosted smart home control center using a Raspberry Pi and ESP32-C6. The system will leverage Zigbee protocol to provide reliable, cloud-independent home automation with unified control.

Critical Issues with Current Smart Home Devices

1. **Cloud Dependency**
2. **Poor Integration**
3. **App Dependencies**
4. **WiFi Bandwidth Saturation**

My Solution

A **local, unified Zigbee control center** that operates independently of cloud services, consolidates device control, and uses dedicated low-power mesh networking to preserve WiFi bandwidth.

Project Architecture

Core Technologies

- **Raspberry Pi OS** - Primary operating system
- **Python 3** - Application logic and sensor management
- **PyQt6** - Local graphical user interface
- **ESP-IDF** - ESP32 firmware development framework
- **Zigbee Protocol** - Low-power mesh networking
- **ESP32-C6** - Zigbee coordinator hardware
- **Raspberry Pi 4** - Main Hardware
- **Sensors** - Various sensors

Sprint 1: Raspberry Pi Sensor Interface Development

Duration: 5 Weeks

Goal: Create functional local control interface with integrated sensors

Deliverables:

- Python3 application running on Raspberry Pi OS
- PyQt6-based graphical user interface
- Integration of three sensor systems:
    - **DHT22** - Temperature and humidity monitoring
    - **LD2410C** - Presence detection
    - **Pi Camera Module v2** - Visual monitoring (WNYHRI 9132664)
- Real-time sensor data display
- Local data logging capabilities

Success Criteria:

- All sensors provide accurate real-time readings
- GUI displays all sensor data clearly
- Application runs stably on Raspberry Pi OS

Technical Requirements:

- Python libraries: Pysense, Adafruit DHT, PiCamera2, PyQt6s
- Proper sensor calibration
- Error handling for sensor disconnections
- Responsive UI updates with multi-threading

Sprint 2: Zigbee Network Integration

Duration: 6 Weeks
Goal: Establish Zigbee network communication between Pi and ESP32-C6 And End Device

Deliverables:

- ESP32-C6 firmware using ESP-IDF
- Zigbee coordinator configuration
- Communication protocol between Raspberry Pi and ESP32
- Network discovery and device pairing functionality
- Integration of Zigbee control into PyQt6s interface

Success Criteria:

- ESP32-C6 successfully acts as Zigbee coordinator
- Raspberry Pi can send commands to ESP32
- Device discovery works reliably
- Interface displays connected Zigbee devices
- Connects to end device
- Zigbee network formation and management
- Status monitoring and error reporting is accurate

Initial Product Backlog

High Priority (Sprint 1-3 Scope)

- [X] Set up Raspberry Pi development environment
- [X] Install and configure Python 3 with required libraries
- [X] Create Raspberry Pi GUI
- [X] Develop PyQt6 GUI framework
- [ ] DHT22 temperature/humidity sensor using adaFruit Library.
- [ ] LD2410C presence sensor
- [ ] Connect Pi Camera Module v2
- [ ] Add data logging functionality

- [X] Set up ESP-IDF development environment in VS Code
- [ ] Configure ESP32-C6 as Zigbee coordinator
- [ ] Implement Pi-to-ESP32 communication protocol
- [ ] Create device pairing workflow
- [ ] Integrate Zigbee controls into PyQt6 interface
- [ ] Write dynamic hashing function to create unique Zigbee PAN IDs
- [ ] Develop Zigbee end device firmware
- [ ] Implement vent servo/motor control
- [ ] Create temperature-based automation logic
- [ ] Add manual override functionality
- [ ] Test full system integration

Medium Priority (Future Enhancements)

- [ ] **Integrate into Home Assistant**
- [ ] Add historical data visualization (graphs/charts)
- [ ] Implement automation scheduling (time-based rules)
- [ ] Create scene/preset configurations
- [ ] Add notification system for alerts
- [ ] Implement backup and restore functionality
- [ ] Create device grouping capabilities
- [ ] Add energy usage monitoring
- [ ] Implement zone-based controls

Low Priority (Nice-to-Have Features)

- [ ] Voice control integration (local only)
- [ ] API for third-party integration
- [ ] Advanced analytics and reporting
- [ ] Multi-user access with permissions
- [ ] Integration with local weather data
- [ ] Geofencing capabilities (local network-based)
- [ ] Firmware OTA update system
- [ ] Custom automation scripting language

Technical Debt & Infrastructure

- [ ] Comprehensive error handling across all components
- [ ] Unit testing for critical functions
- [ ] Documentation for setup and configuration
- [ ] Code refactoring for maintainability
- [ ] Security hardening (authentication, encryption)
- [ ] Performance optimization
- [ ] Resource usage monitoring
- [ ] Automated testing framework

References and Documentation

- ESP32-C6 Datasheet
- Zigbee 3.0 Specification
- DHT22 Sensor Documentation
- LD2410C mmWave Radar Sensor Guide
- Raspberry Pi Camera Module v2 Documentation
- ESP-IDF Programming Guide

Question Time!