ESP32-Based Smart Home Automation System

Han Guo

Email: hguo1217@gmail.com | Phone: (780) 394-9157

GitHub: https://github.com/PollybearG/ESP32-Based-Smart-Home-Automation-System.

January 2025 - April 2025

Team project to build a smart home system with ESP32 for temperature monitoring, light control, and break-in detection. My role: Handled all hardware connections, coded sensor functions, and assisted with Blynk and Telegram communication.

Abstract

Background

Smart homes are becoming common because they make life easier with things like remote control of lights and security alerts. Technologies like ESP32 are cheap and versatile, with built-in Wi-Fi and Bluetooth for connecting devices. This project drew from real-world systems like Ajax for security and Xiaomi for integration, but we aimed for a simple, customizable setup using open tools.

Project Overview

The system uses ESP32 to monitor temperature (DHT20 sensor), control garden lights (LDR sensor), and detect break-ins (motion and door sensors with camera alerts). We connected everything to Blynk for real-time app control and Telegram for notifications. I focused on wiring the hardware and writing code for the sensors, while helping with Blynk and Telegram parts since the team required everyone's input.

Project Design

Hardware

- Microcontroller: ESP32 for Wi-Fi/Bluetooth and low power.
- Sensors: DHT20 for temperature/humidity, LDR for light, Ultrasonic Sensor for motion, and door switch for entry detection.
- Other: Servo for locks, LEDs for lights, ESP32-CAM for photos (switched to laptop camera due to pin conflicts).

GitHub repository (Doc/Project_Overall_Circuit.pdf).

Software

Coded in C with Arduino IDE, handling sensor readings and device control. Full code available in GitHub repository (Code/ESP32-Based_Smart_Home_Automation_System).
Used Blynk for the app interface, and Telegram records the video and uses the videos for alerts.
GitHub repository (Doc/Blynk_and_Telegram_Interfaces.pdf).

Implementation

- Wired all hardware connections, including I2C and SPI for sensors, ensuring stable integration on the ESP32 board.
- Wrote code to read DHT20/LDR/HC-sr04 data and trigger actions like LED on/off or servo movement.
- Assisted with Blynk dashboard setup for remote control and Telegram bot for photo alerts, as the team shared tasks.
- Tested subsystems for real-time response, achieving 50ms delays in monitoring.

Results

- Temperature and light control worked reliably, with automatic adjustments based on sensor data.
  GitHub repository (Doc/Project_Sensor_Test_Working_Results.pdf).
- Break-in detection captured photos and sent Telegram alerts within seconds, improving security.
- Overall system achieved stable performance in a simulated home setup.

Challenges and Solutions

- GPIO pin conflicts with ESP32-CAM: Switched to laptop camera to free pins, keeping the system functional.
- Sensor accuracy issues: Calibrated DHT20 and LDR through code tweaks for better readings.
- Team coordination: Divided tasks but ensured everyone contributed, like sharing Blynk/Telegram code.

Conclusion

This project gave me hands-on experience with ESP32 hardware wiring and coding for sensors, which helped me understand how to build reliable IoT systems. It shows my ability to handle real-time data and integrations like Blynk, useful for embedded roles.

Improvement

- Add more sensors like smoke detectors for full home coverage.
- Switch to MQTT for faster communication.
- Integrate AI for smarter alerts, like face recognition on captured photos.

References

- Espressif Systems. (2023). ESP32 Technical Reference Manual. espressif.com

- Instructables. (2025). Smart Home Automation Using Blynk & ESP32 IoT Projects. instructables.com
- Random Nerd Tutorials. (n.d.). ESP32/ESP8266 Thermostat Web Server. randomnerdtutorials.com
- Taiwo, O. et al. (2022). Enhanced Intelligent Smart Home Control and Security System. Journal of Intelligent Systems.

(Note: Report reconstructed and optimized based on original Sheridan College project to highlight my contributions in hardware connections and sensor coding.)