SIMPLE HOME AUTOMATION SYSTEM

S: Design Lab

21 STYCZNIA 2025

AUTORZY: MATEUSZ SZYCH, NIKODEM SZAFRAN

Table of contents:

1) Initial project assumptions

a) Functionalities:
   a. Communication implemented via the MQTT protocol
   b. Intuitive GUI system that a new user can master in approximately 15 minutes
   c. Control of system components using a computer or smartphone
   d. Fast system responsiveness (<200ms)

b) Power supply:
   a. The ESP8266 board will be powered through a computer USB port or a 15V DC adapter
   b. Power pins on the board will operate at the 3.3V standard
   c. A 5V power output, normally unavailable on the board, will be added through an adapter module extension

c) Interfaces:
   a. Multiple lighting sources, implemented in the project using LEDs

    b. Temperature sensor LM35
    c. Alarm system with arming capability, consisting of:
        i. PIR motion sensor HC-SR501, which will detect movement when the alarm is armed
        ii. HXD buzzer, emitting an alarm sound when motion is detected by the sensor

d) Other assumptions:
    a. None

2) Text introduction to the Project

The main goal of the project was to create a solution that would serve as a foundation or base for a home automation system using the open-source OpenHAB software.
OpenHAB is an excellent choice for home automation systems as it offers flexibility, wide compatibility, and extensive scalability. This software supports many communication protocols and standards, making it suitable for both simple and complex installations. OpenHAB allows the integration of various devices into one system, significantly simplifying home management and control. The system even supports voice control solutions, such as Alexa. OpenHAB also provides an intuitive graphical user interface, enabling quick changes and system configuration according to needs without requiring much coding.

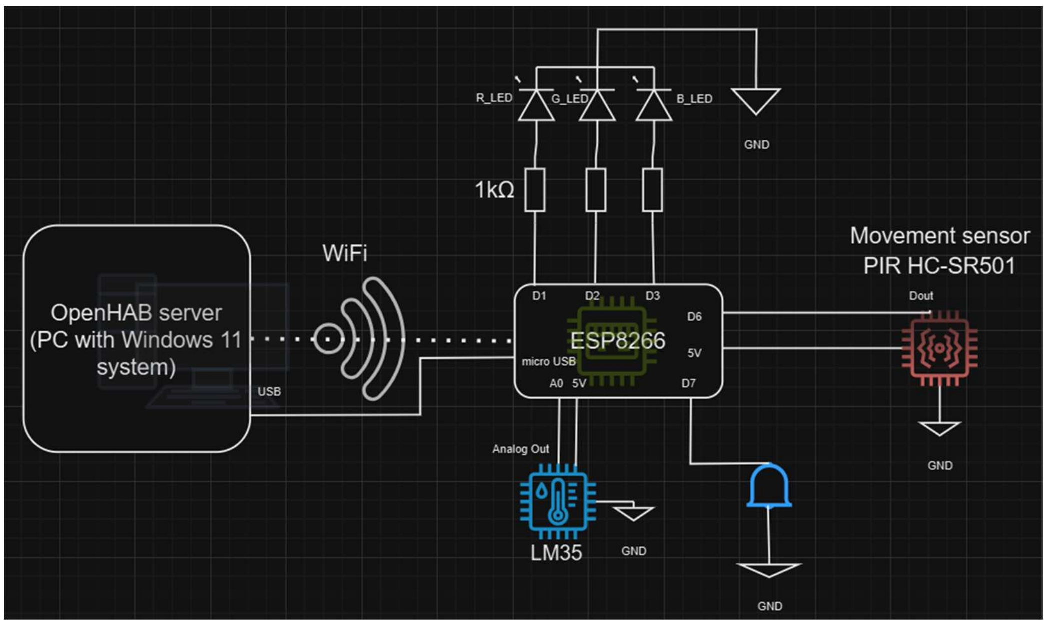
In our project, communication between all components and OpenHAB is achieved using the MQTT protocol, with the ESP8266 board, within a local WiFi network. This ensures smooth operation and easy scalability for future expansion. MQTT is implemented using Mosquitto Eclipse on a personal computer running the Windows operating system.

The system allows control via a mobile application or web interface. In our system, we can control three different light sources. We also use a temperature sensor to monitor environmental conditions. The integrated alarm system features arming and disarming functions and includes a motion detector to identify potential breaches of the protected zone. The system is designed to be scalable, enabling the addition of new features without removing existing ones.

3) Work schedule

1. **Creation of a GitHub Repository**
   - Set up a central repository to store and manage the project.
2. **Preliminary Project Sketch**
   - Develop an initial draft outlining the system architecture and design.
3. **Task Distribution and Collaboration Rules**
   - Assign responsibilities among team members and establish guidelines for collaboration.
4. **Launching OpenHAB on the PC**
   - Install and configure the OpenHAB software on a personal computer to serve as the control hub.
5. **Programming the ESP8266 Board**
   - Write and upload the firmware necessary for the ESP8266 to communicate with the system.
6. **Integrating OpenHAB with the ESP8266**
   - Establish communication between OpenHAB and the ESP8266 board using the MQTT protocol.
7. **Adding Components to the System**
   - Connect and integrate various devices, such as sensors and actuators, into the system.
8. **Testing and Improvements**
   - Conduct thorough testing to identify and fix any issues, ensuring a stable and functional system.

4) Block diagram



5) Component specifications

1. **LM35 (Temperature Sensor):**
    - Output Voltage: 10 mV/°C
    - Accuracy: ±0.5°C (at 25°C)
    - Operating Voltage: 4V to 30V
    - Operating Temperature Range: -55°C to +150°C
2. **LED DIODES:**
    - Forward Voltage: Typically 1.8V to 3.3V (depending on color)

- Forward Current: ~20 mA

3. **PIR Sensor HC-SR501:**
   - Operating Voltage: 4.5V to 20V
   - Detection Range: Up to 7 meters
   - Detection Angle: 120 degrees
   - Adjustable Time Delay: 5 seconds to 5 minutes
4. **ESP8266 (Wi-Fi Module):**
   - Operating Voltage: 4.5V – 10V
   - Using Wi-Fi standard
   - Power Consumption: ~80 mA during transmission, ~10 μA in deep sleep
   - Clock speed 80 MHz
   - 11 digital pins
   - microUSB
   - Temperature range: -40°C – 125°C

6) Practical Implementation of the Project

1) Preparation of the Development Environment:

a. **Install Azul JDK version 17** – This version ensures OpenHAB runs most efficiently and with minimal issues.
b. **Install OpenHAB** – Use the Windows version of the software.
c. **Install Mosquitto Eclipse** – This will be used for communication between OpenHAB and the ESP8266 board.
d. **Install Arduino IDE** – Along with the required libraries, especially those necessary for MQTT communication.

2) Configuring Mosquitto:

a. Configure the Mosquitto configuration file to allow connections from any address on the WiFi network without requiring login credentials.

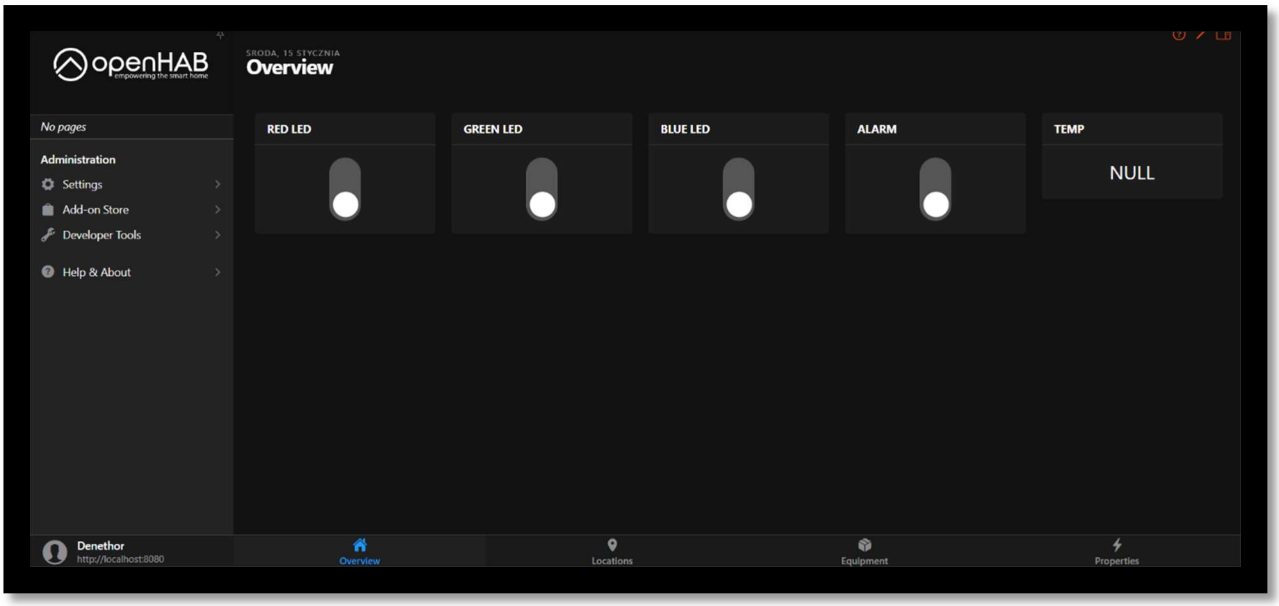
3) Configuring OpenHAB:

a. Create an admin account with a username and password.
b. Download the "MQTT" extension necessary for communication using this protocol.
c. Create an MQTT Broker "Thing" – This will enable communication.
d. Link it to Mosquitto by entering the computer's IP address.
e. Create representations of physical components as "Things."
f. Create virtual switches to control the components.
g. Connect the switches to the components using "Channels" and send appropriate messages through specified topics.
h. Add a page in the OpenHAB interface to display the switches created.

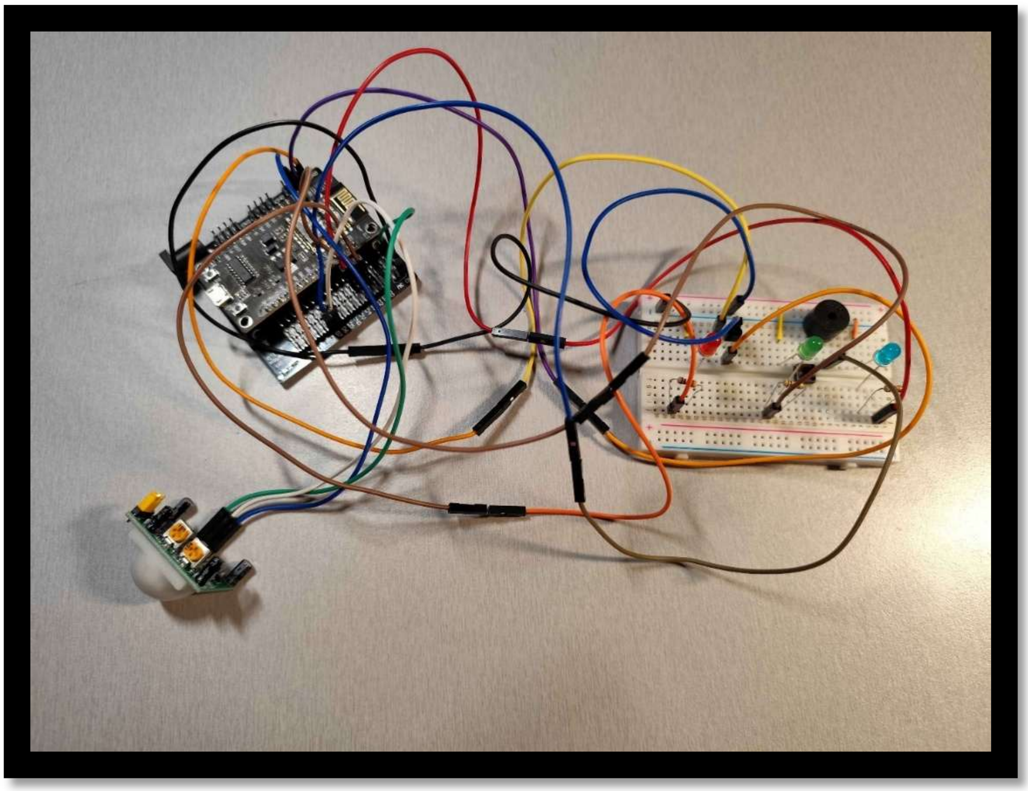
4) Programming the ESP8266 Board:

a. Install the PubSubClient library.
b. Write functions for handling message reception and transmission using the MQTT protocol.
c. Implement code to control LEDs from OpenHAB (message reception only) – The user can toggle LEDs on and off via the web or mobile app.
d. Implement code for reading temperature from the sensor (message transmission only) – The ESP8266 will send temperature readings to OpenHAB, where they will be displayed in the GUI every 5 seconds.

e. Implement code for arming/disarming the alarm system (message reception and transmission).



OpenHab GUI after finishing



Physical implementation of the project

7) Duties

A. Creation of GitHub repository – **Mateusz Szych**
B. Implementation of Preliminary Project Assumptions – **Mateusz Szych & Nikodem Szafran**
C. Creation of Block Diagram – **Mateusz Szych**
D. Initializing of OpenHab server and connecting it with MQTT Broker – **Nikodem Szafran**
E. Physical implantation of scheme on board – **Mateusz Szych & Nikodem Szafran**
F. Configuration of ESP8266 board, adding basic functionalities – **Nikodem Szafran**
G. Implementation of addidional functionality – **Mateusz Szych**

8) Sources

https://mosquitto.org/

https://www.openhab.org/

https://www.arduino.cc/en/software

https://www.azul.com/downloads/?package=jdk#zulu