010123131

Software Development Practice I

Handout #9

<rawat.s@eng.kmutnb.ac.th>

Last Update: 2024-08-13

Open Source

Smart Home Platforms

Expected Learning Outcomes

- Understand the basic concepts and architecture of **smart home systems**.
- Identify and describe **various smart home devices** and their roles within a **smart home ecosystem**.
- Explain the **MQTT communication protocol** and its significance in IoT and smart home environments.
- Demonstrate the ability to integrate different smart home protocols and devices to create a unified smart home system.

Expected Learning Outcomes

- Understand the principles of **ZigBee wireless communication** and its applications in smart home networks.
- Understand the concept of **Tasmota firmware** for ESP-based devices.
- Compare features between Tasmota and other solutions such as ESPHome.
- Flash / configure Tasmota firmware on ESP-based devices, and customize Tasmota settings for various sensor modules and smart home applications.

Home Automation / Smart Home

- A **Smart Home**, or **Home Automation**, refers to the integration of hardware and software into the home environment to enhance convenience, security, and energy efficiency through automated control.
- Smart home devices usually use wireless technologies such as **Wi-Fi**, **Bluetooth / BLE**, **Zigbee**, **Z-Wave** or **Matter** to communicate and interact.
- Smart homes can automate tasks based on schedules or triggers according to specific conditions defined by users.

Home Automation / Smart Home

- Many smart home devices and platforms use **HTTPS** and **WebSocket** protocols.
  - **HTTPS** is a standard protocol used to secure communications by encrypting data between the client and the servers.
  - **WebSocket** is a protocol providing full-duplex communication channels over a single TCP connection. It is commonly used for real-time communication between devices and smart home platforms.

Commercial Smart Home Platforms

- Examples include:
  - **Apple HomeKit**
  - **Samsung SmartThings**
  - **Google Home**
  - **Xiaomi Mi Home**
  - **Tuya Smart**

Open-source Smart Home Platforms

- Examples include:
  - **Home Assistant (HA)**: https://www.home-assistant.io/
  - **OpenHAB**: https://github.com/openhab
  - **Domoticz**: https://github.com/domoticz/domoticz
  - **ioBroker**: https://github.com/ioBroker/ioBroker

Choosing Smart Home Platforms

- Key issues when choosing smart home platforms
  - Device Integration and Support
  - Communication Protocol Support
  - Ease of Use (Setup and Configuration)
  - GUI User Interface / Dashboard
  - Active Community and Support, Documentation

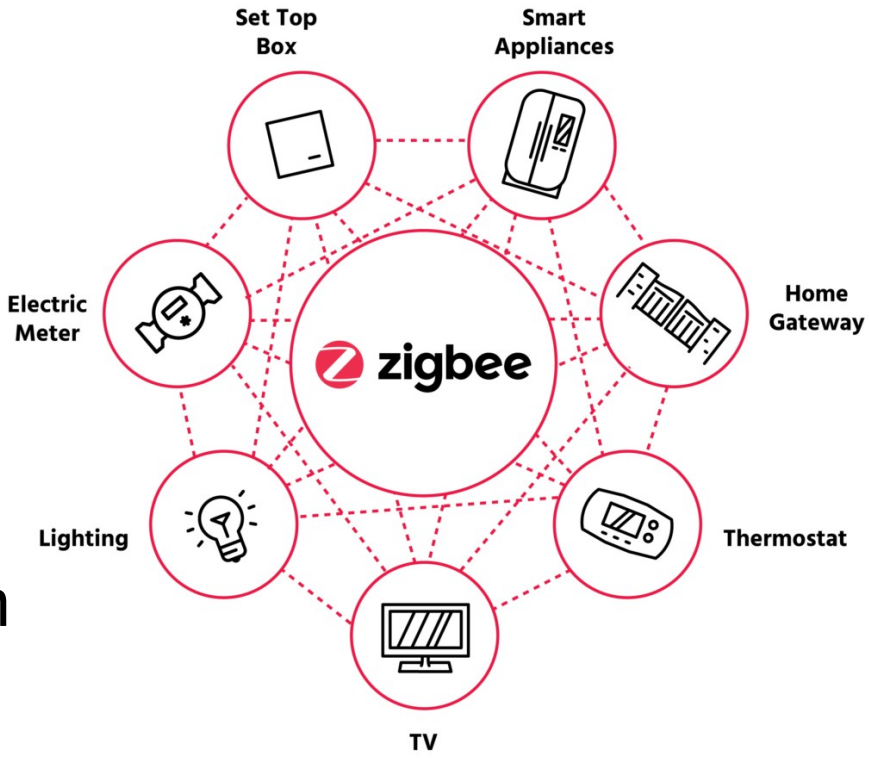
Choosing Smart Home Platforms

- Key issues when choosing smart home platforms
  - Active Development and Updates
  - Extensibility and Customization
  - Performance and Reliability
  - Security and Privacy
  - Scalability
  - Multi-vendor Device Support

Zigbee

- **Zigbee** is an open-standard **wireless communication protocol** specifically designed for **low-power, low-data-rate applications** in home automation and IoT.
- It supports **Mesh networking** with low power consumption of sensor devices.

Image Source: https://csa-iot.org/



Smart Home

Zigbee

- **Zigbee Standard**: The **Connectivity Standards Alliance (CSA)**, formerly known as the **Zigbee Alliance**, developed the Zigbee standards.
- Zigbee is based on **EEE 802.15.4**, which is a standard for low-rate wireless personal area networks (**LR-WPANs**) for the physical (**PHY**) and **MAC** layers.
- It also adds its own layers including the network (**NWK**), application (**APL**), and security (**APS**).

Zigbee

- **Zigbee 3.0**: The **Zigbee Pro specification** adds new features such as child device management, improved security, and new network topology options.
- Zigbee networks typically operate on the following **frequency bands:**
  - **2.4 GHz Band (world-wide)**: 16 channels (numbered 11 to 26) with a data rate of up to 250 kbps.
  - **868 MHz Band** (Europe)
  - **915 MHz Band** (North America)

Zigbee Stack Components

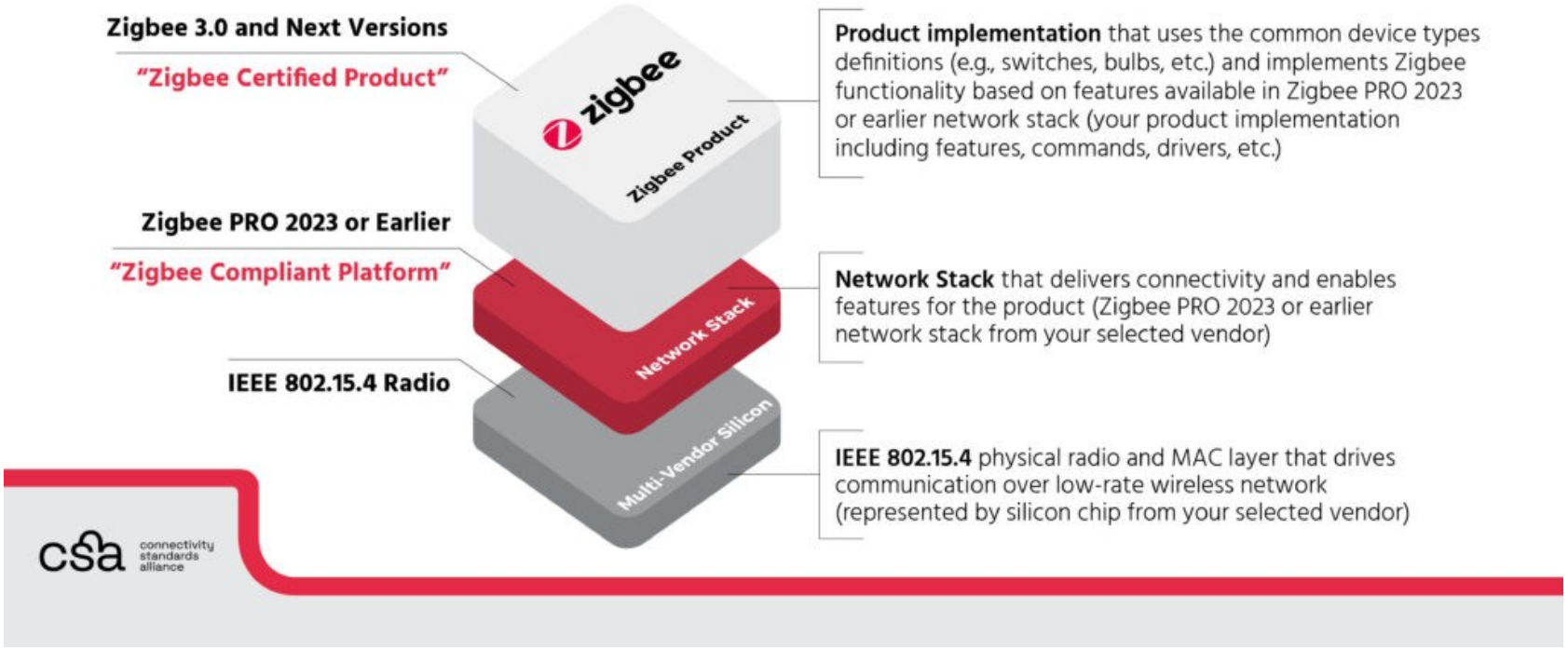


Image Source: https://csa-iot.org/newsroom/zigbee-stack-components/

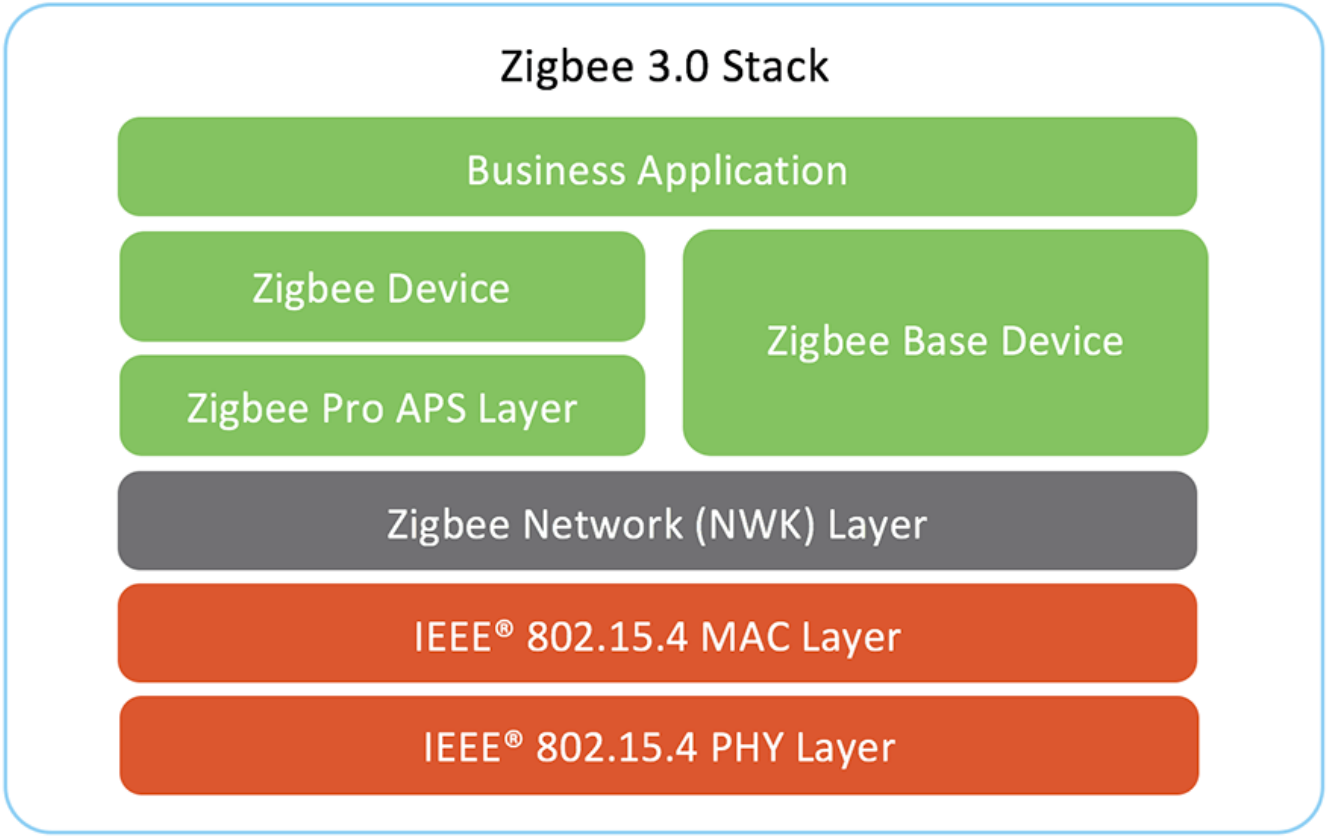


Image Source: https://www.digi.com/solutions/by-technology/zigbee-wireless-standard

Features of Zigbee

- **Flexibility** — Supports multiple network topologies such as point-to-point, point-to-multipoint and mesh networks
- **Low-duty cycle** — Provides long battery life
- **Low latency** — Can easily transport sensor data with minimal latency
- **Scalability** — Includes Direct Sequence Spread Spectrum (DSSS) up to 65,000 nodes per network
- **Robustness** — Employs collision avoidance, retries and acknowledgments
- **Low power consumption** — Zigbee devices can operate for several years on a single inexpensive battery thanks to its use of a power-saving feature called “sleep mode”
- **Low data rate** — With a data rate of up to 250 kbit/s, Zigbee is best suited for intermittent data sensor or device transmissions
- **Security** — Zigbee security uses 128-bit AES encryption as well as many additional security techniques.

Matter

- It is a unified standard for device compatibility and interoperability, bridging gaps between different communication protocols and ecosystems.
- It supports networking and Internet connectivity (based on IPv6) and is compatible with existing technologies, including **Zigbee** and **OpenThread**.
- Matter uses **Thread / OpenThread** as one of its networking layers to support Mesh networking and low-power communication between devices.

MQTT

- **MQTT** = *Message Queuing Telemetry Transport*
- It is a lightweight, **publish-subscribe network protocol** for efficient **Machine-to-Machine (M2M)** data transmission in IoT and smart home systems.
- MQTT operates over TCP/IP at the network transport layer, which ensures that MQTT messages are reliably transmitted over the network.

MQTT

- MQTT is based on a server-client architecture: MQTT clients and MQTT brokers (servers) communicate using TCP/IP sockets.
- Key functions of an MQTT broker include:
  - Topic-based message routing
  - Quality of Service (QoS) based message delivery
  - Session Management for MQTT subscribers and publishers
  - Security (Authentication and data encryption)

MQTT Brokers

- Many MQTT brokers do support **WebSocket**, allowing web-based clients (such as browsers) to connect to the MQTT broker.
- Examples of public MQTT brokers include:
  - **Mosquitto (open source)**
  - **HiveMQ**
  - **EMQX**

ZigBee Hub

- A **Zigbee** Hub is a device that serves as a central coordinator for a ZigBee network (called the **Zigbee network coordinator**).
- It also serves as a bridge between ZigBee devices (**Zigbee end devices** or **Zigbee routers**) and other systems, such as smart home platforms or cloud services.

Examples of Zigbee 3.0 Hubs

- **Philips Hue Bridge**
- **Samsung SmartThings Hub**
- **Amazon Echo Plus**
- **Tuya ZigBee 3.0 Smart Gateway Hub**
- **Xiaomi Gateway (Zigbee 3.0)**
- **Athom Zigbee 3.0 Gateway**

athom
Smart Home

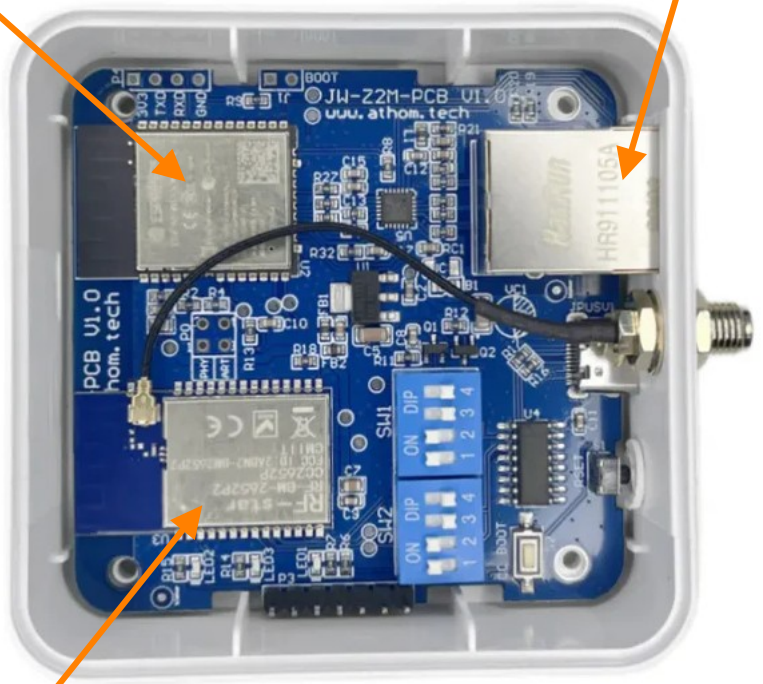
MQTT

ZigBee

Athom ZG01 Zigbee Gateway

Espressif ESP32-WROOM-32E Module

RJ45 port for Ethernet



Ti CC2652P Module

Image Source: https://athombridge.com/products/athom-zigbee-gateway

ZigBee2MQTT

- It is an **open-source project** that allows **Zigbee devices** to communicate with a smart home system via **MQTT**.
- It acts as a bridge or gateway between Zigbee networks and an MQTT broker.
- **Zigbee2MQTT** bridges the gap by converting Zigbee communication to MQTT, allowing ZigBee devices to work with MQTT-based smart home systems.
- Many smart home systems and platforms support MQTT for device integration.

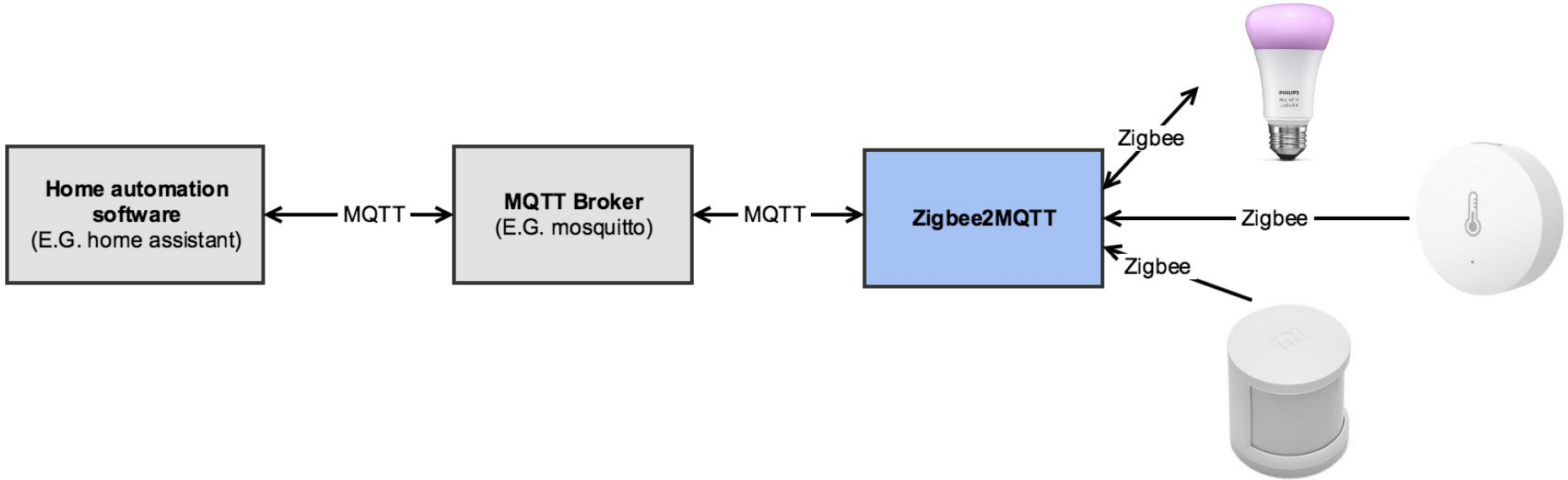


Image Source: ZigBee2MQTT

Home Assistant (HA)

- It is an **open-source home automation platform** that focuses on privacy and local control.
- It provides **a user-customizable dashboard** to control and monitor smart home devices.
- It can integrate with popular **voice assistants** like **Amazon Alexa** and **Google Assistant**, allowing for voice control of your smart home devices.

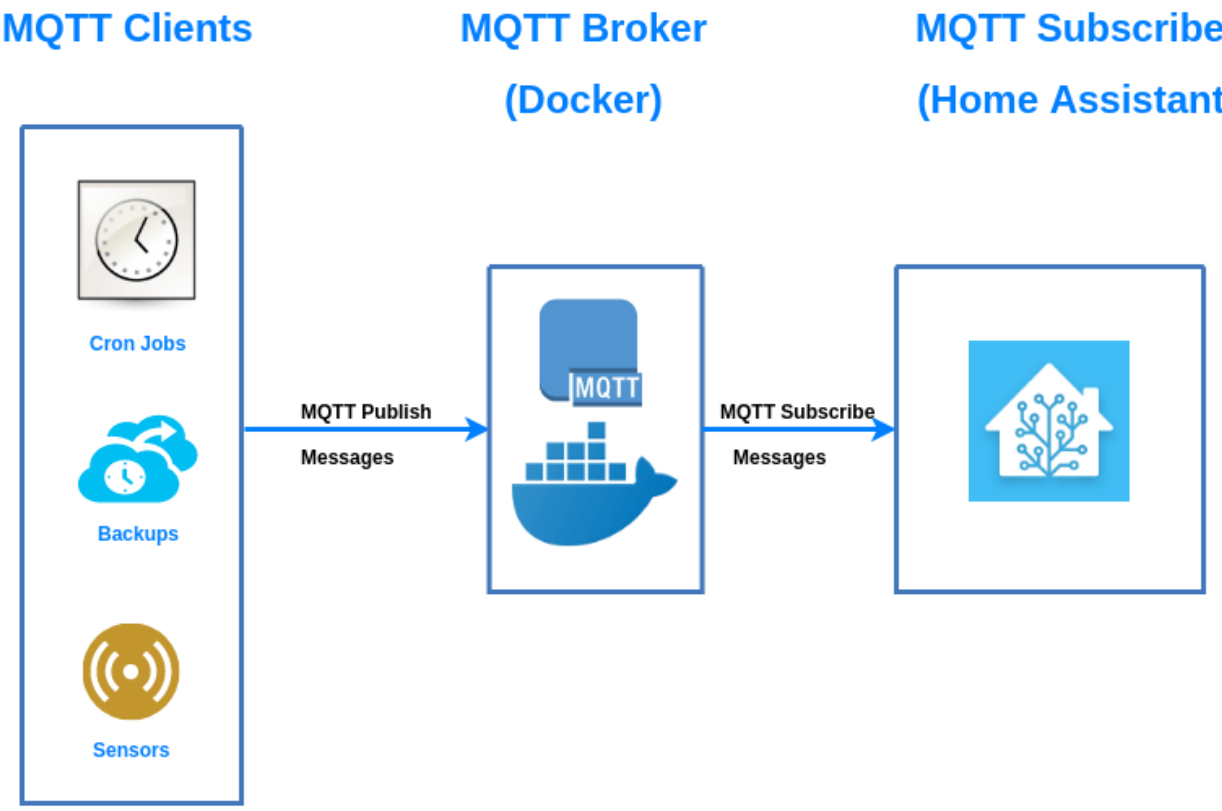


Image source: https://hometechhacker.com/mqtt-home-assistant-using-docker-eclipse-mosquitto/

Options for Deploying HA

- **Home Assistant OS**: *recommended installation method*
  - A dedicated OS for HA that runs on SBCs such as RPi.
- **Home Assistant Core + Supervised**:
  - A manual installation of HAt on any Python-supported system.
- **Home Assistant Container using Docker**:
  - A containerized installation of the HA using a Docker image from Docker Hub.

Node-RED

- **Node-RED** is an **open source, flow-based development tool for visual programming** of home automation tasks.
- It provides a graphical user interface (GUI) for creating and managing flows using a drag-and-drop approach.
- Node-RED can be installed as an **add-on in Home Assistant (for HA OS)**.

Node-RED

- Node-RED supports a variety of communication protocols and standards used in smart home applications such as HTTPS, WebSocket and MQTT.
- Node-RED can be used with Home Assistant (HA), which is an open-source smart home platform, using WebSockets for communication.

Firmware Options for Smart Home Devices

- **Tasmota** and **ESPHome** are both popular open-source firmware options for ESP8266 and ESP32-based devices, especially DIY projects.
  - **Tasmota**: https://tasmota.github.io/docs/, https://github.com/arendst/Tasmota
  - **ESPHome**: https://esphome.io/index.html, https://github.com/esphome/esphome
- Both software support over-the-air (OTA) updates of firmware for **ESP8266** and **ESP32**-based devices.

Tasmota vs. ESPHome

- Both **Tasmota** and **ESPHome** are not a standalone smart home platform but rather a tool for creating firmware for smart devices. It is often used in conjunction with platforms like **Home Assistan**t.
- **ESPHome** utilizes the **ESP-IDF framework** and **PlatformIO (PIO core)** for building its firmware and **YAML** is used to define various settings and parameters for the **ESPHome firmware**.
- In contrast, **Tasmota** is based on the **Arduino framework**. The Tasmota firmware utilizes Arduino core libraries for the **ESP8266 and ESP32 microcontrollers**.

Tasmota

- **Tasmota** is an **open-source firmware** designed primarily for **Espressif's ESP8266 and ESP32-based devices**, commonly used in smart home applications.
- Tasmota devices can be controlled via:
  - Web UI (https://tasmota.github.io/docs/WebUI/).
  - HTTP / WebSockets, MQTT (Documentation and Serial port

Tasmota

- Tasmota supports **Over-the-Air (OTA) firmware updates**, simplifying the process of keeping devices up to date.
- It supports integration with smart home or home automation platforms such as **Home Assistant (HA)**.
- Documentation:
  - https://tasmota.github.io/docs/

Tasmota

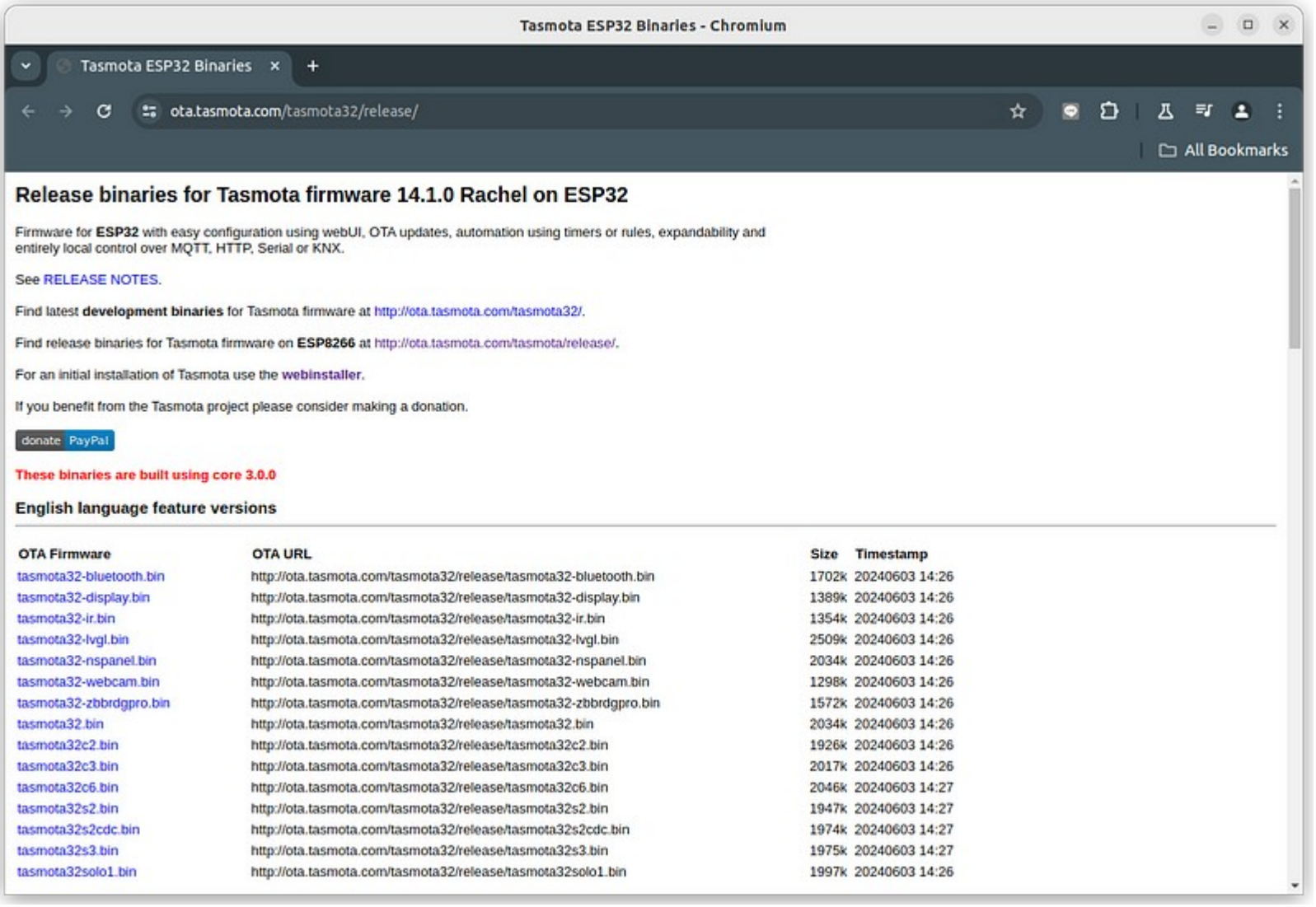
- For ESP32 devices, the Tasmota source code utilizes the **Arduino-ESP32 core v3.0.x+**.
- The following ESP32 device families are supported:
  - **Tensilica Xtensa based**: ESP32, ESP32-S2, ESP32-S3
  - **RISC-V based**: ESP32-C2, ESP32-C3, ESP32-C6, ESP32H2

Tasmota Firmware Options

- There are two categories of **pre-compiled firmware** files (.bin).
  - Initial firmware vs. OTA firmware (from the official OTA server).
  - ESP8266: https://ota.tasmota.com/tasmota/release/
  - ESP32: http://ota.tasmota.com/tasmota32/release/

Tasmota Firmware Options

- There are different Tasmota firmware variants (both initial and OTA firmware).
- Initial firmware .bin files for different ESP families:
  - **ESP32**: `tasmota32.factory.bin`
  - **ESP32C3**: `tasmota32c3.factory.bin`
  - **ESP32S3**: `tasmota32s3.factory.bin`



Release binaries for Tasmota firmware 14.1.0 Rachel on ESP32

Firmware for **ESP32** with easy configuration using webUI, OTA updates, automation using timers or rules, expandability and entirely local control over MQTT, HTTP, Serial or KNX.

See RELEASE NOTES.

Find latest **development binaries** for Tasmota firmware at http://ota.tasmota.com/tasmota32/.

Find release binaries for Tasmota firmware on **ESP8266** at http://ota.tasmota.com/tasmota/release/.

For an initial installation of Tasmota use the **webinstaller**.

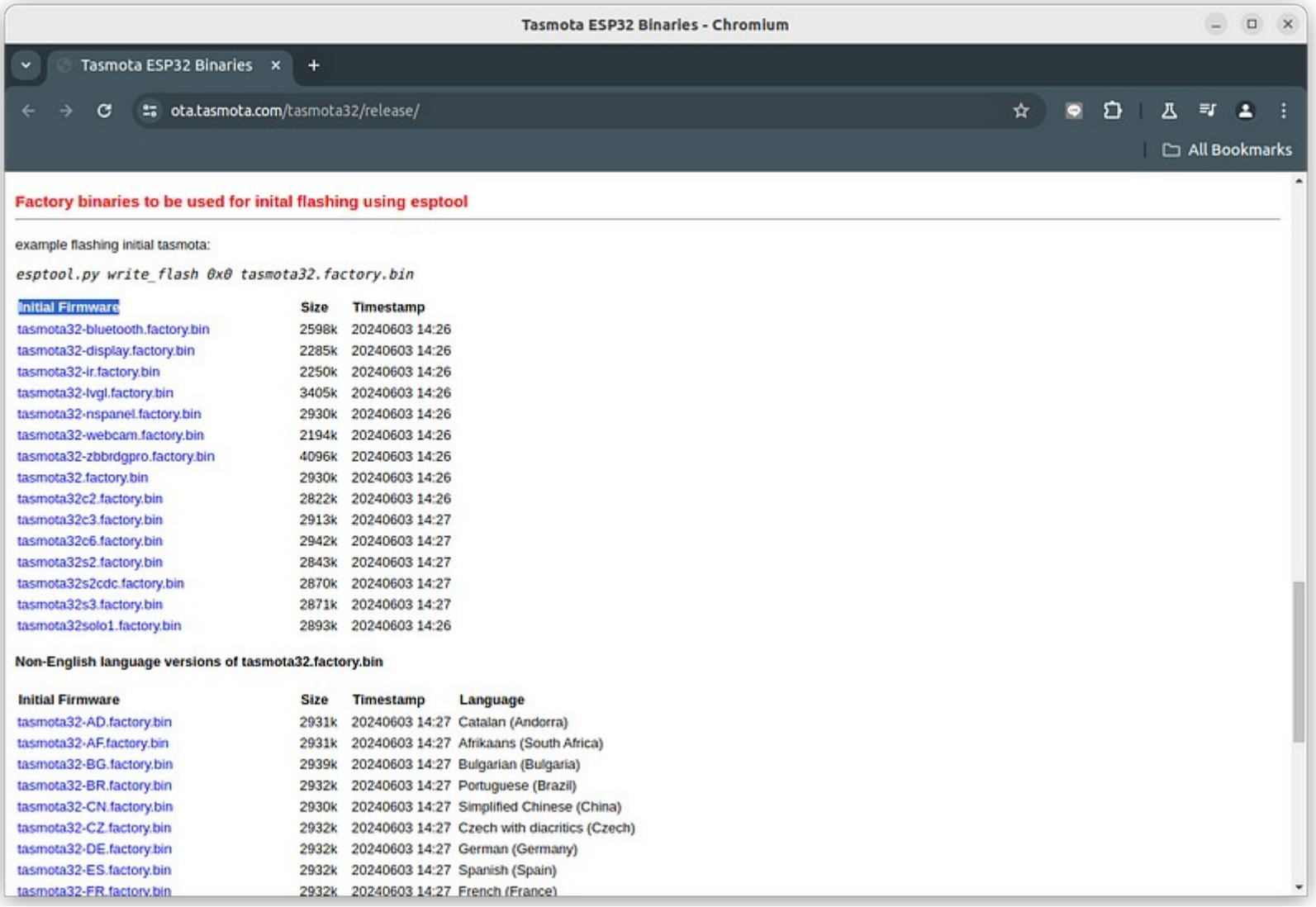
If you benefit from the Tasmota project please consider making a donation.

donate PayPal

These binaries are built using core 3.0.0

English language feature versions

OTA Firmware	OTA URL	Size	Timestamp
tasmota32-bluetooth.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-bluetooth.bin	1702k	20240603 14:26
tasmota32-display.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-display.bin	1389k	20240603 14:26
tasmota32-ir.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-ir.bin	1354k	20240603 14:26
tasmota32-lvgl.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-lvgl.bin	2509k	20240603 14:26
tasmota32-nspanel.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-nspanel.bin	2034k	20240603 14:26
tasmota32-webcam.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-webcam.bin	1298k	20240603 14:26
tasmota32-zbbrdgpro.bin	http://ota.tasmota.com/tasmota32/release/tasmota32-zbbrdgpro.bin	1572k	20240603 14:26
tasmota32.bin	http://ota.tasmota.com/tasmota32/release/tasmota32.bin	2034k	20240603 14:26
tasmota32c2.bin	http://ota.tasmota.com/tasmota32/release/tasmota32c2.bin	1926k	20240603 14:26
tasmota32c3.bin	http://ota.tasmota.com/tasmota32/release/tasmota32c3.bin	2017k	20240603 14:26
tasmota32c6.bin	http://ota.tasmota.com/tasmota32/release/tasmota32c6.bin	2046k	20240603 14:27
tasmota32s2.bin	http://ota.tasmota.com/tasmota32/release/tasmota32s2.bin	1947k	20240603 14:27
tasmota32s2cdc.bin	http://ota.tasmota.com/tasmota32/release/tasmota32s2cdc.bin	1974k	20240603 14:27
tasmota32s3.bin	http://ota.tasmota.com/tasmota32/release/tasmota32s3.bin	1975k	20240603 14:27
tasmota32solo1.bin	http://ota.tasmota.com/tasmota32/release/tasmota32solo1.bin	1997k	20240603 14:26



Factory binaries to be used for inital flashing using esptool

example flashing initial tasmota:

esptool.py write_flash 0x0 tasmota32.factory.bin

Initial Firmware	Size	Timestamp
tasmota32-bluetooth.factory.bin	2598k	20240603 14:26
tasmota32-display.factory.bin	2285k	20240603 14:26
tasmota32-ir.factory.bin	2250k	20240603 14:26
tasmota32-lvgl.factory.bin	3405k	20240603 14:26
tasmota32-nspanel.factory.bin	2930k	20240603 14:26
tasmota32-webcam.factory.bin	2194k	20240603 14:26
tasmota32-zbbrdgpro.factory.bin	4096k	20240603 14:26
tasmota32.factory.bin	2930k	20240603 14:26
tasmota32c2.factory.bin	2822k	20240603 14:26
tasmota32c3.factory.bin	2913k	20240603 14:27
tasmota32c6.factory.bin	2942k	20240603 14:27
tasmota32s2.factory.bin	2843k	20240603 14:27
tasmota32s2cdc.factory.bin	2870k	20240603 14:27
tasmota32s3.factory.bin	2871k	20240603 14:27
tasmota32solo1.factory.bin	2893k	20240603 14:26

Non-English language versions of tasmota32.factory.bin

Initial Firmware	Size	Timestamp	Language
tasmota32-AD.factory.bin	2931k	20240603 14:27	Catalan (Andorra)
tasmota32-AF.factory.bin	2931k	20240603 14:27	Afrikaans (South Africa)
tasmota32-BG.factory.bin	2939k	20240603 14:27	Bulgarian (Bulgaria)
tasmota32-BR.factory.bin	2932k	20240603 14:27	Portuguese (Brazil)
tasmota32-CN.factory.bin	2930k	20240603 14:27	Simplified Chinese (China)
tasmota32-CZ.factory.bin	2932k	20240603 14:27	Czech with diacritics (Czech)
tasmota32-DE.factory.bin	2932k	20240603 14:27	German (Germany)
tasmota32-ES.factory.bin	2932k	20240603 14:27	Spanish (Spain)
tasmota32-FR.factory.bin	2932k	20240603 14:27	French (France)

Tasmota Firmware Flashing

- The initial firmware can be flashed using **ESPTool** (a Python-based tool) developed by Espressif:

```
$ esptool.py write_flash 0x0 <tasmota32.factory.bin>
```

- Alternatively, Tasmota firmware can be flashed onto an ESP32 device via the **Tasmota Web Installer**.
  - URL https://tasmota.github.io/install/

Tasmota Firmware Flashing

- After the initial firmware has been flashed and the device is configured to connect to a Wi-Fi network, it can be updated with OTA firmware (over Wi-Fi).
- **Tasmota's Device Template**
  - A template defines an ESP8266 or ESP32 device and how its GPIOs are assigned.
  - Learn more about templates: https://tasmota.github.io/docs/Templates/

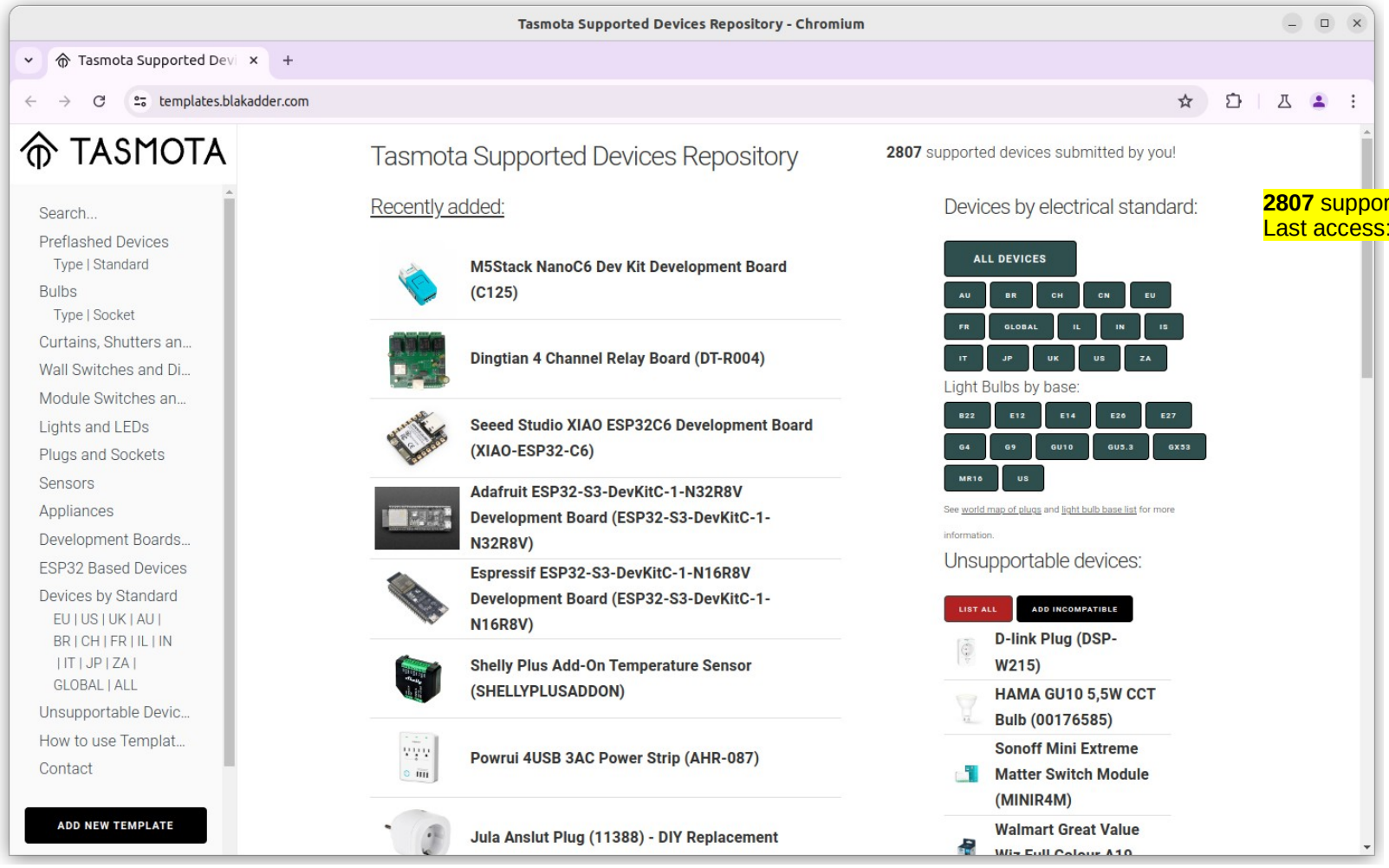
Tasmota Firmware Build

- Since Tasmota firmware is open source, users can configure and compile custom versions (for example, using a Docker container).
  - Firmware Options: Choose between `tasmota` (for ESP8266) or `tasmota32` (for ESP32).
  - Customization: Modify features by editing the `user_config_override.h` file.
- Documentation:
  - https://tasmota.github.io/docs/Compile-your-build/

Tasmota Rules

- Tasmota supports **rules** that are used to trigger events and send MQTT messages, or trigger other rules or actions, enabling complex automation sequences.
- Example:
  - Turn on an air-conditioner when a temperature sensor reading exceeds 25°C and send an MQTT message to the MQTT broker when this happens.

Tasmota Supported Devices Repository

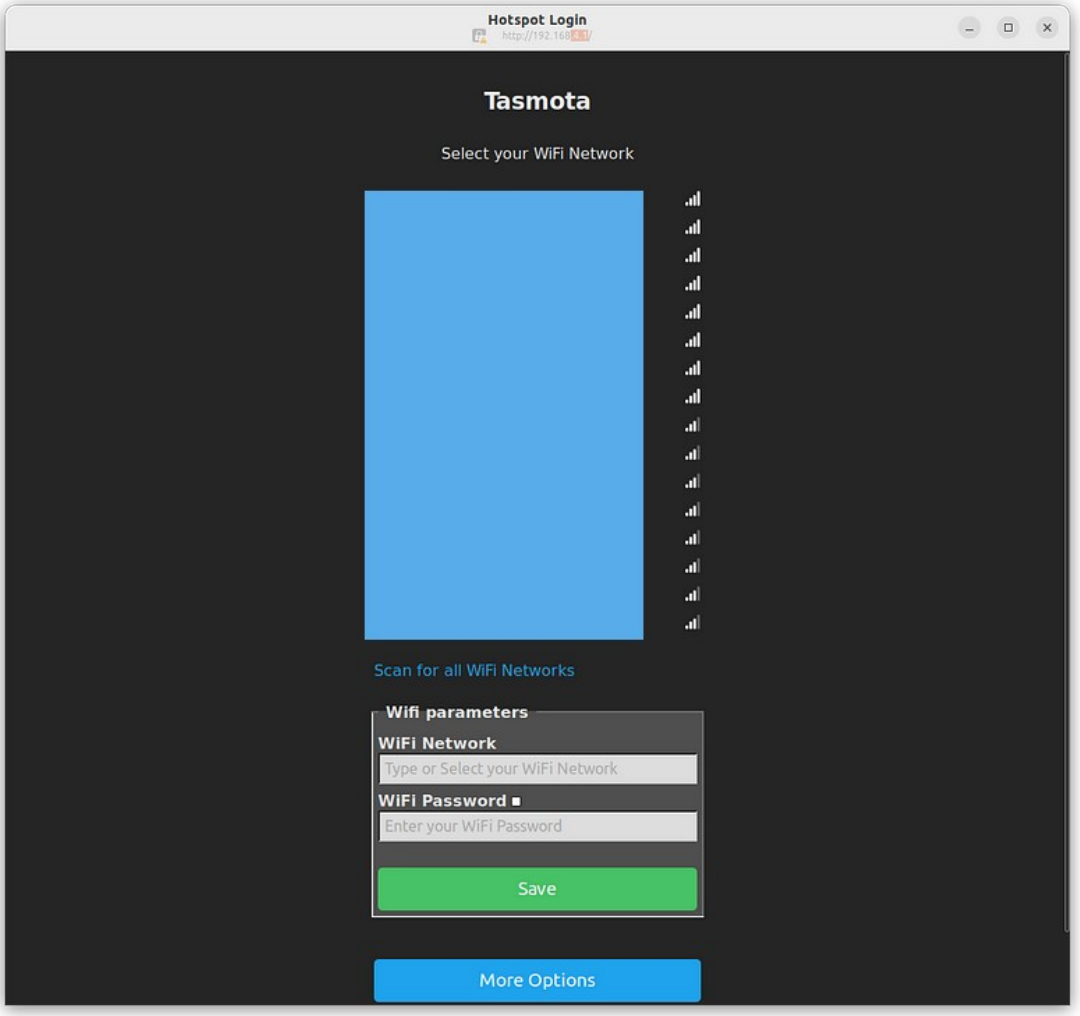


2807 supported devices
Last access: **2024-08-13**

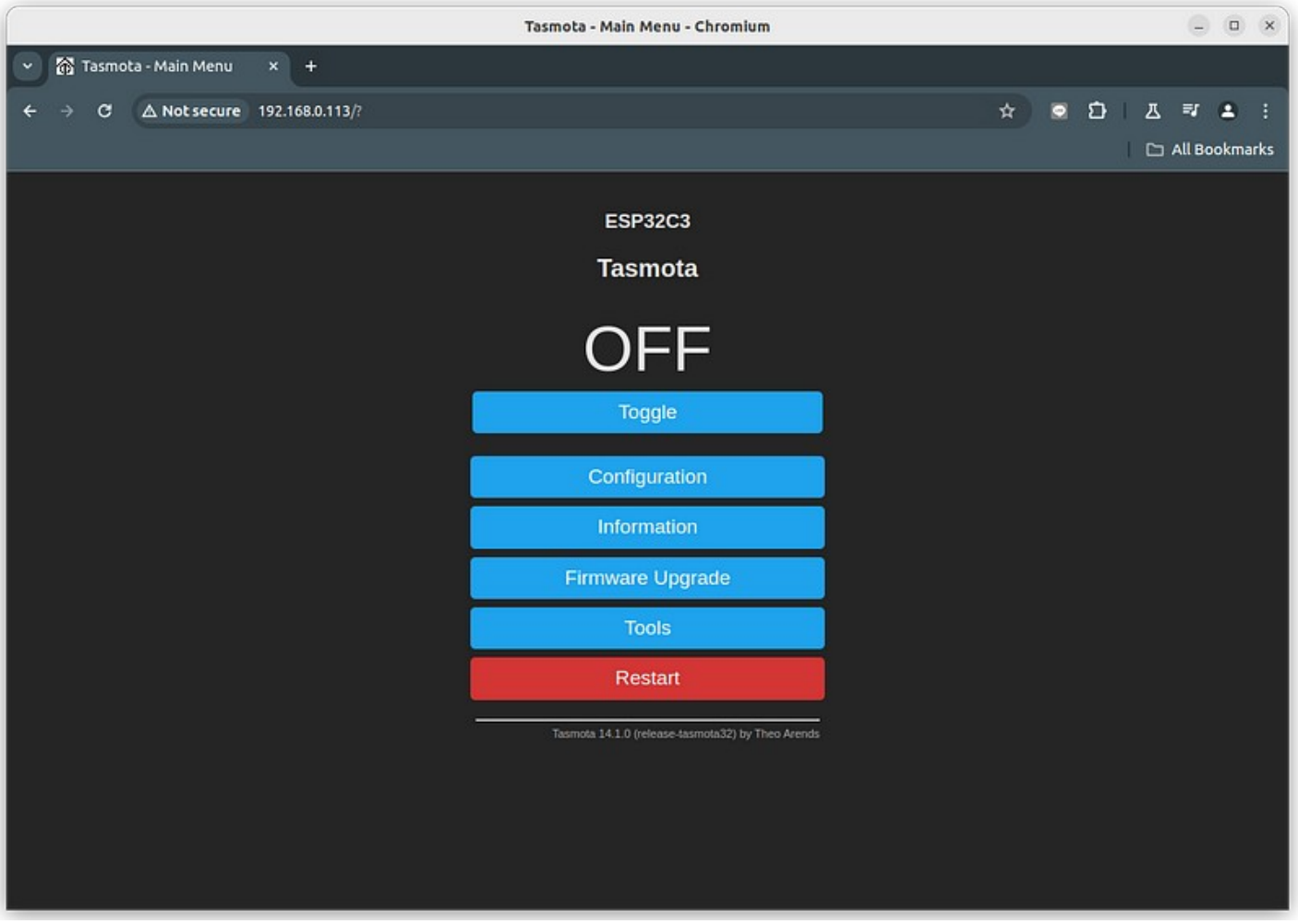
Tasmota Device Setup

- Steps to do after flashing the Tasmota firmware to an ESP8266 / ESP32 device.
  - **Initial Wi-Fi Setup**: Connect to the ESP32's Wi-Fi AP with the SSID: `tasmota-XXXX` where XXXX are hexadecimals.
    - The default IP address of the device: `192.168.4.1`.
  - **Wi-Fi Setting**: Enter the SSID and password for the Wi-Fi network to be used and reset the device.
  - Device Configuration: Set the device template and other configurations.

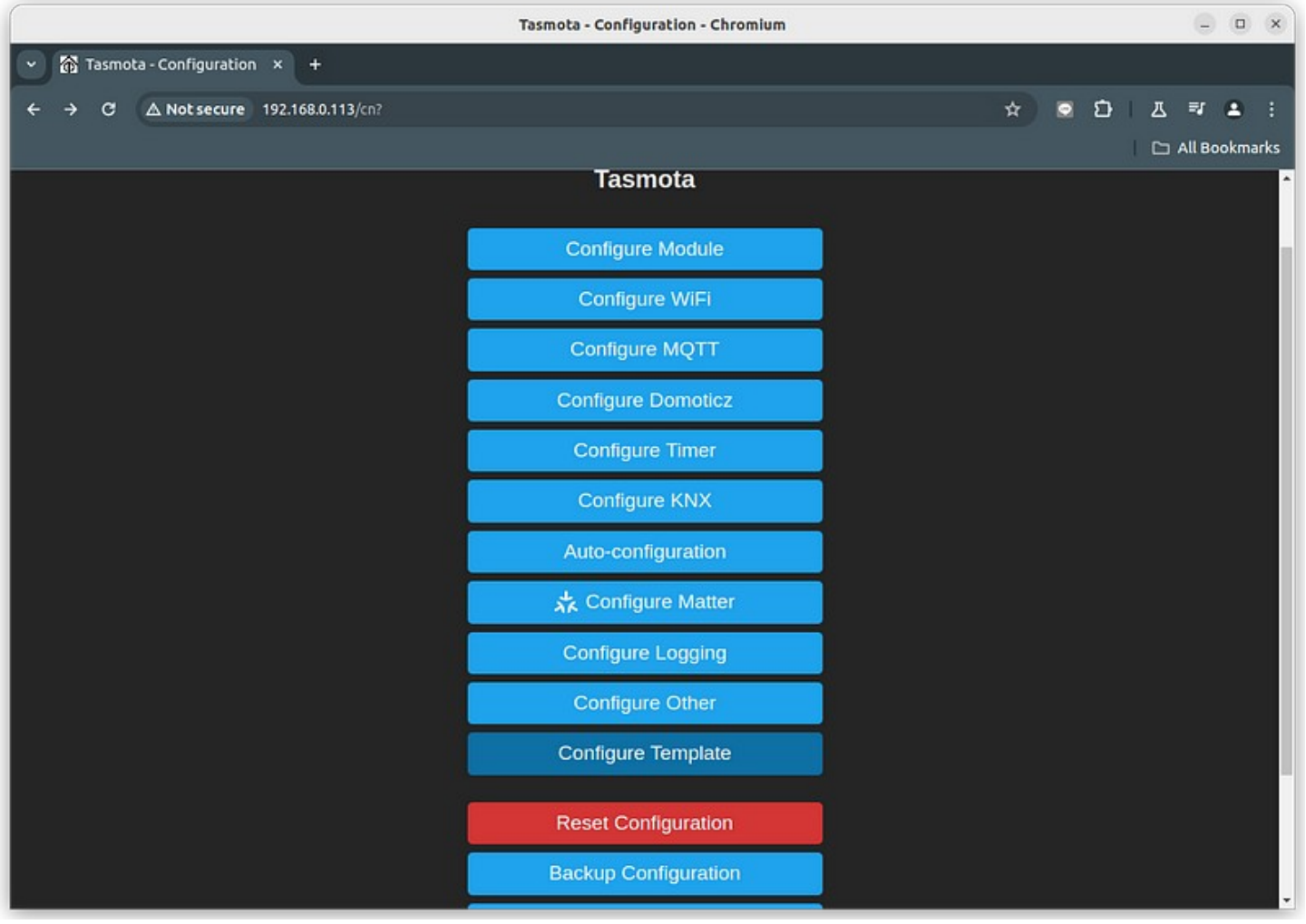
Initial Wi-Fi Setup



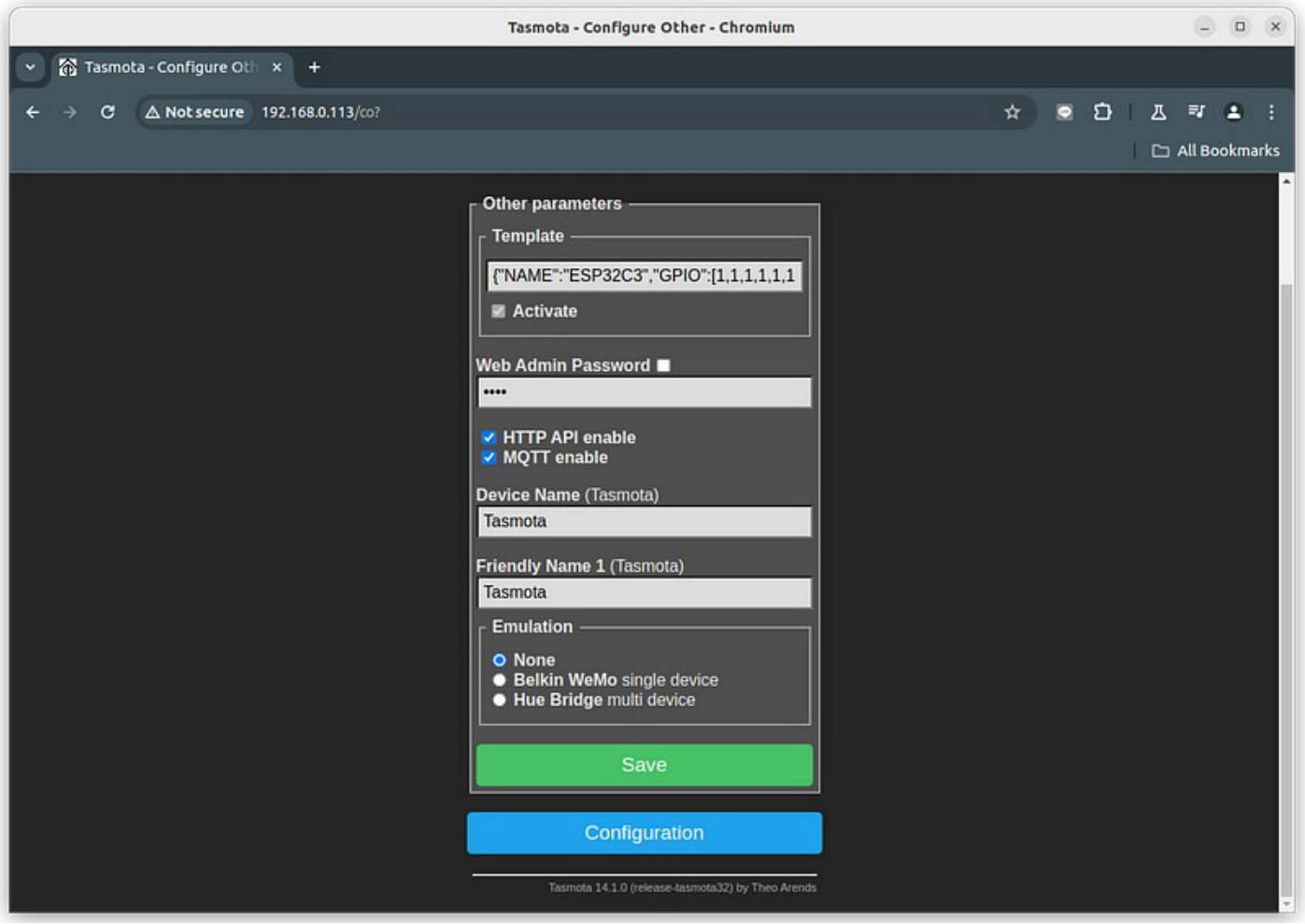
Tasmota Web Interface – the Main Menu page



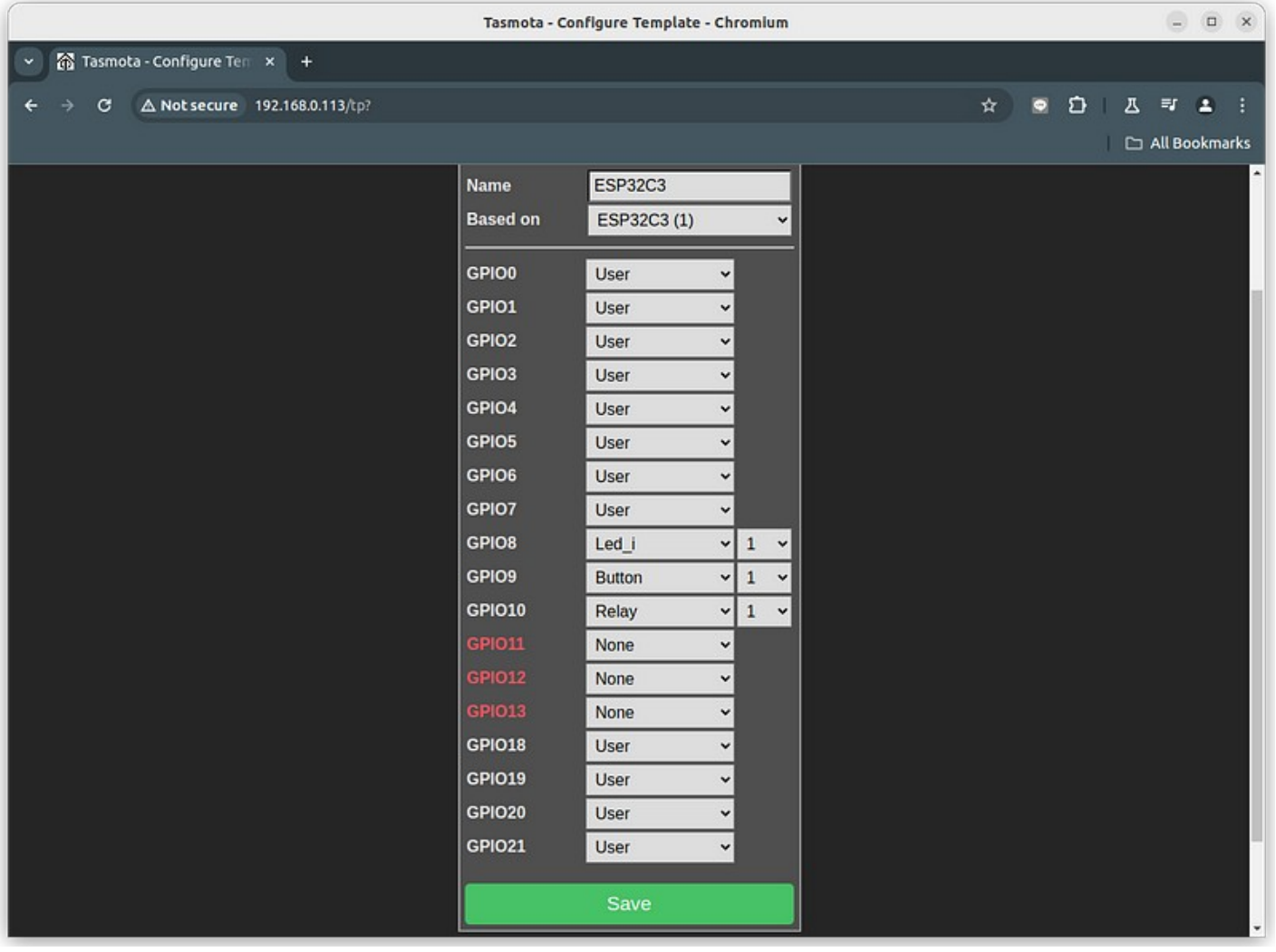
Tasmota Web Interface – Configuration Settings



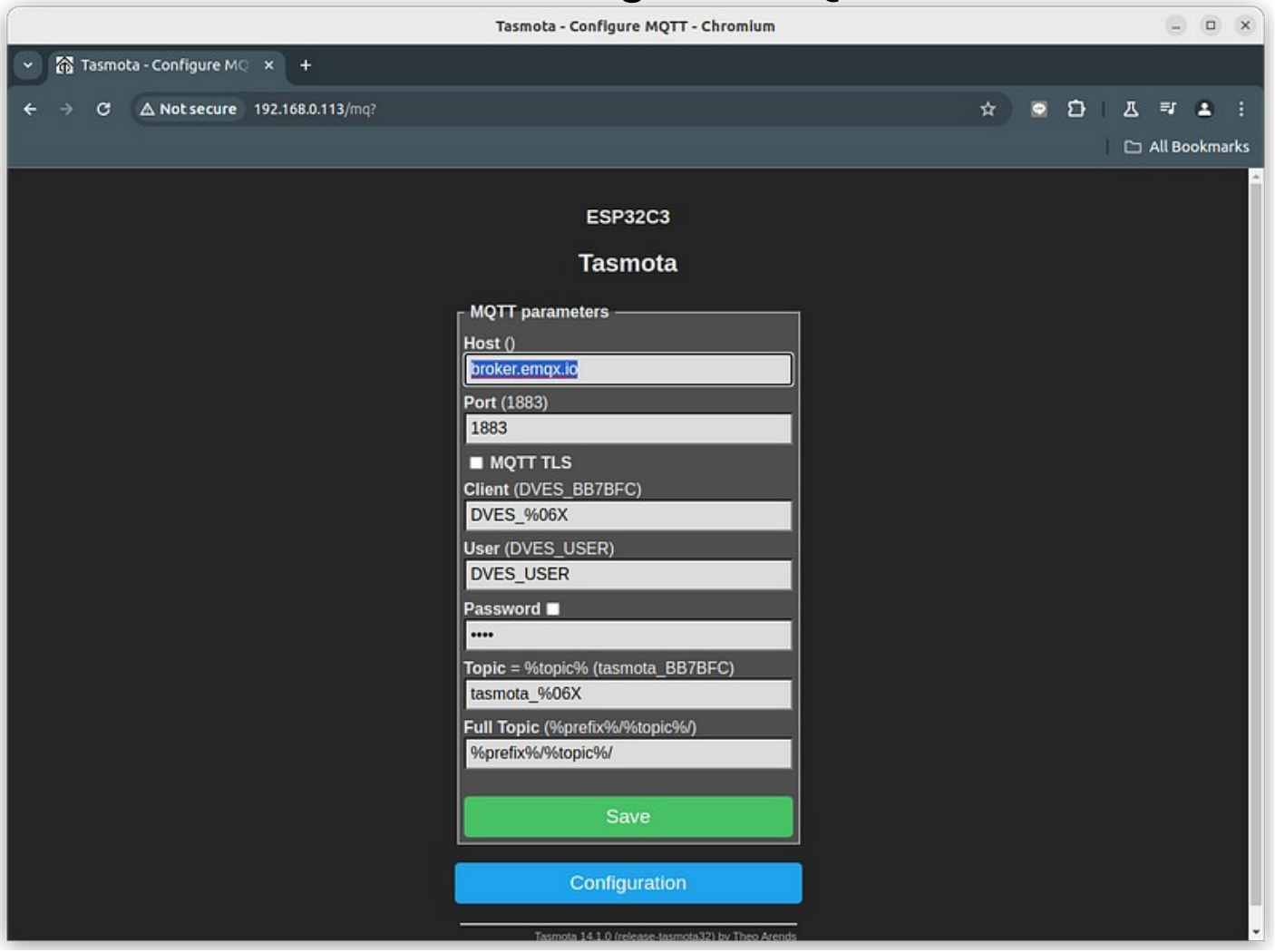
Tasmota Device Template and GPIO Configuration



User-defined GPIO Settings for ESP32C3 Device



User-defined Settings for MQTT Broker



Sonoff BASICR4 Switch Module (BASICR4)

GLOBAL

Available from:

Itead.cc

Aliexpress.com

Banggood.com

Install method:

USB to Serial

ESP32-C3

GPIO #	Component
GPIO00	None
GPIO01	None
GPIO02	None
GPIO03	None
GPIO04	Relay 1
GPIO05	None
GPIO06	LedLink
GPIO07	None
GPIO08	None
GPIO09	Button 1
GPIO10	None
GPIO12	None
GPIO13	None
GPIO18	None
GPIO19	None

EDIT ON GITHUB

Configuration for ESP32-C3

```
{"NAME":"Sonoff Basic R4","GPIO":[0,0,0,0,224,0,544,0,0,32,0,0,0,0,0,0,0,0,0,0,0,0],"FLAG":0,"BA
```

Image Source: https://templates.blakadder.com/sonoff_BASICR4.html

Tasmota – Matter Support

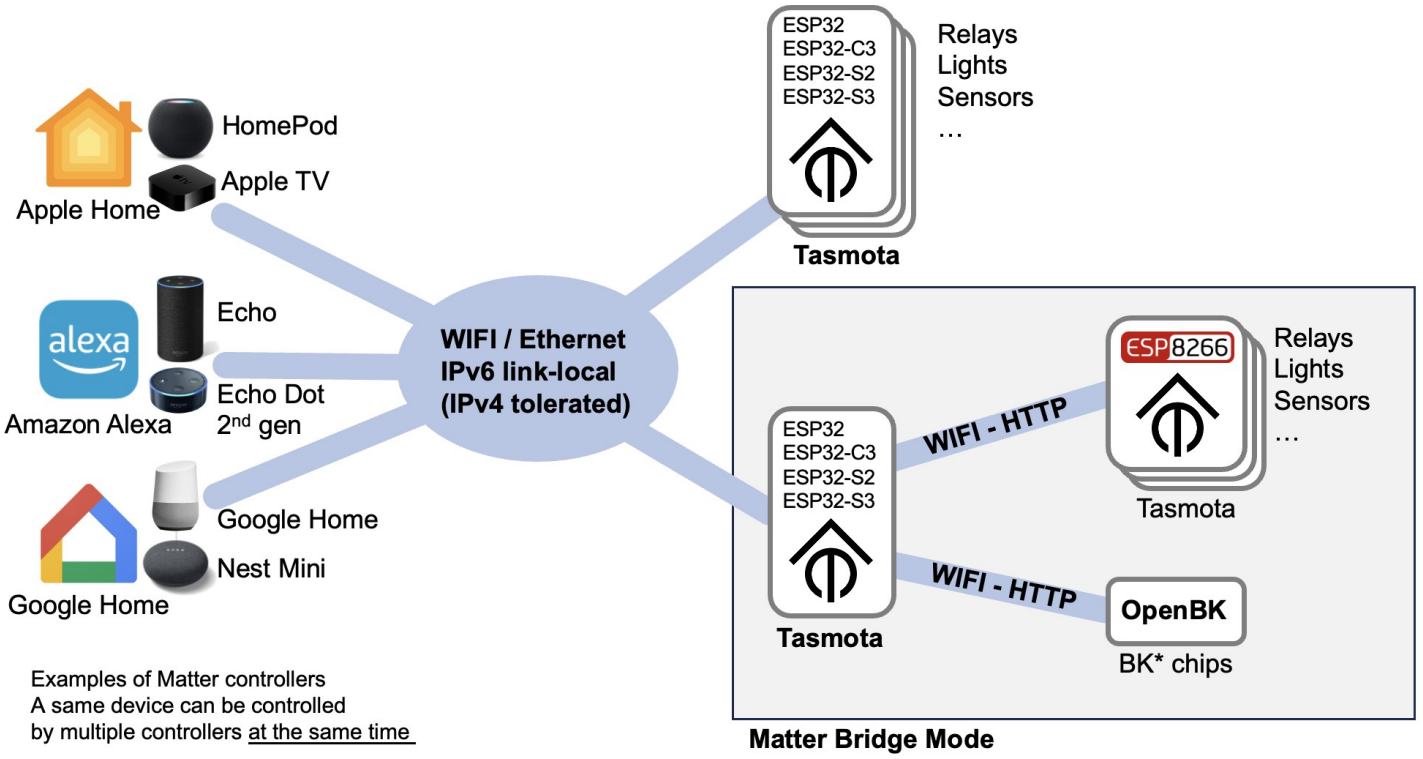


Image Source: https://tasmota.github.io/docs/Matter/

Key features of Tasmota

- To summarize, the following are key features of Tasmota:
  - Open source and free
  - Support for a wide variety of sensor modules
  - Customization, flexibility and local control
  - MQTT support
  - OTA firmware support
  - User-Friendly Web Interface
  - Low-code or no-code feature customization