Instruction
for ESP Mesh-network
with WASM

Otoya Nakakaze

June 4, 2023

Contents

1 Introduction

This document explains usage of our ESP32 mesh-network prototype with WASM and Web IDE for updating WASM on the microcontroller.

2 Implementation

Our prototype builds a mesh-network between microcontrollers and allows for transmitting data and WASM binary data.

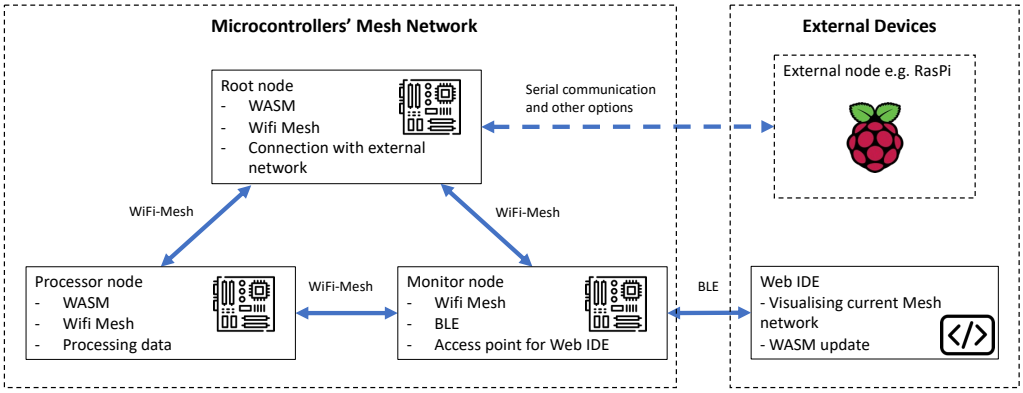

Figure 1: Overview

Figure 1 shows overview of our prototype. There are three types of nodes in microcontroller's Wifi-mesh network: root, processor, and monitor node.

- **Root node** provides connection with external networks/devices. For example, this node gathers data from other mesh-network's members and transmit to an external device.
- **Processor node** executes WASM module that processes logged data, e.g., filtering, sampling, and etc.
- **Monitor node** has access-point via BLE (bluetooth low energy). A user can access this node with browser, get view of current network, write new code, and update WASM on an abitrary node in the network.

Our implementation uses ESP-WiFi-Mesh[1]. This implementation assumes that the root node will be loaded by managing messages from mesh members

[1]https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/esp-wifi-mesh.html

a lot. Therefore, the root and the monitor node are sepaleted. If the root takes the role of monitor node (BLE functionarity), make sure that Wasm task is disabled.

3 Installation

This section explains how to upload sample program on an ESP microcontroller. The source code is available under `https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo`

3.1 Overview of Sample

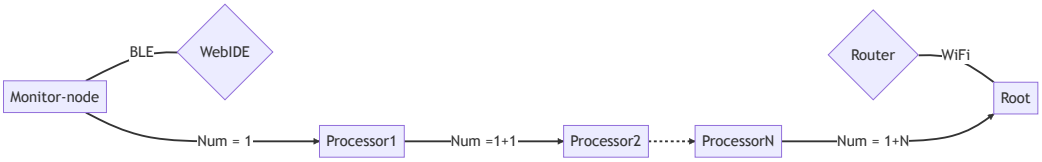

Figure 2: Overview of Sample ESP-Mesh-Network with WASM

Our sample mesh-network provides a simple data stream processing. A monitor node sends a number (default 1) to target nodes. If a processor node receives data, add 1 to the input number using WASM module and forward it.

3.2 Tools and Requirements

Tools to be installed

- Visual Studio Code
- Platform IO for VS code

Requirements

- This prototype uses ESP-WIFI-MESH and create a fixed-root mesh network, therefore, a router is necessary.
- At least **three** ESP boards are required for the root, processor, and monitor node.
- Exactly one ESP must be the root node.

3.3 Configuration

- Check your ESP board and edit platformio.ini
- In main.c for only root node, set your router's SSID and password
- The default max. number of data stream distination is TWO. To change this, set new MESH_DATA_STREAM_TABLE_LEN (number of destinations * 6) for target nodes

3.4 Upload the program to ESPs

Make sure that your board is connected and PIO destinates the corresponding port. After build and upload the program to the MCU, a sample WASM file main.wasm also has to be uploded. You find this operation button under `Project task -> <board name> -> platform -> upload file system image` in the PIO menu.

4 Web IDE

This section shows usage of Web IDE. Figure 3 shows the initial state of the IDE. You find links here.

- source code: `https://github.com/Ayato77/Wasm-ble-web-ide`.
- Web IDE: `https://wasm-ide-for-esp32.onrender.com/ide.html`

This experimental Web UI communicates an ESPs via BLE and provides the following functionarities:

- AssemblyScript IDE, compiling and uploading WASM
- Datastream modification based on the graph
  - data stream (add/remove links)
  - moving WASM module
  - delete WASM module

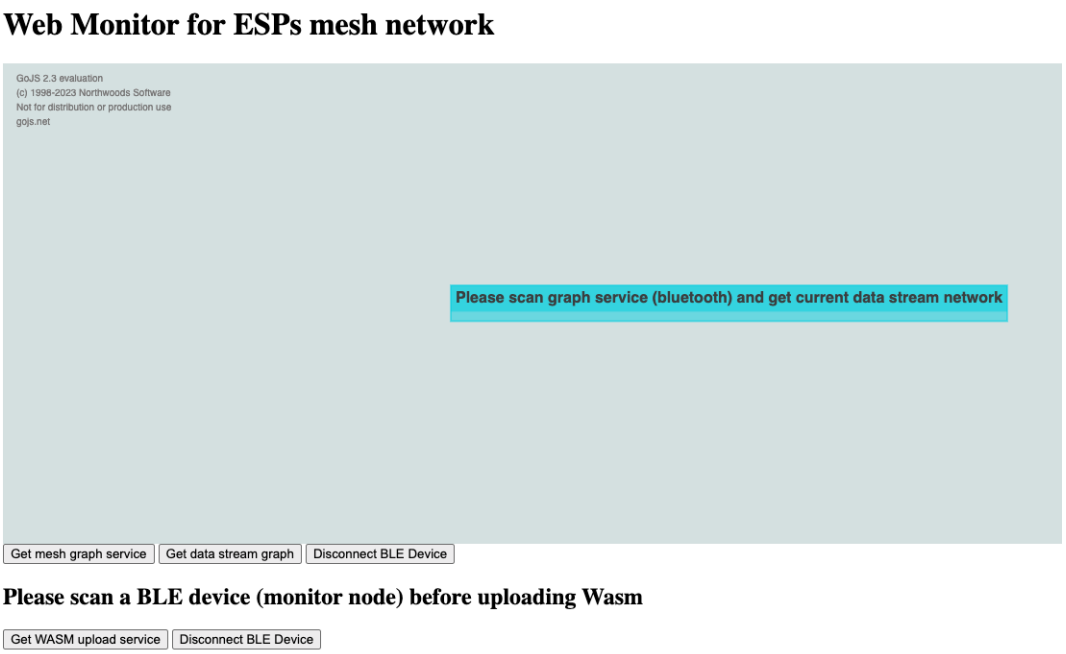

Figure 3: Web IDE image

4.1 Usage

Due to the implementation, there are two GATT services for uploading WASM and graph functions. Therefore, each service should be scanned and connected separately.

How to update data stream graph

1. Scan a monitor node with the button "Get mesh graph service"
2. Click the button "Get data stream graph" (Figure 4)
3. You can add a link between nodes by dragging (Figure 5).

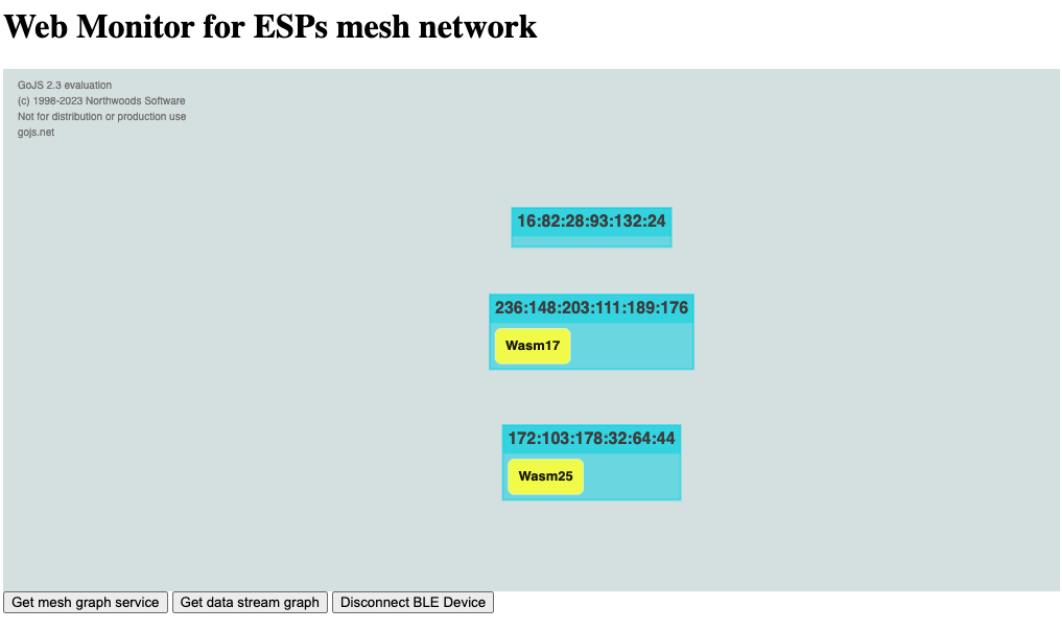

Figure 4: Graph after the initial scan

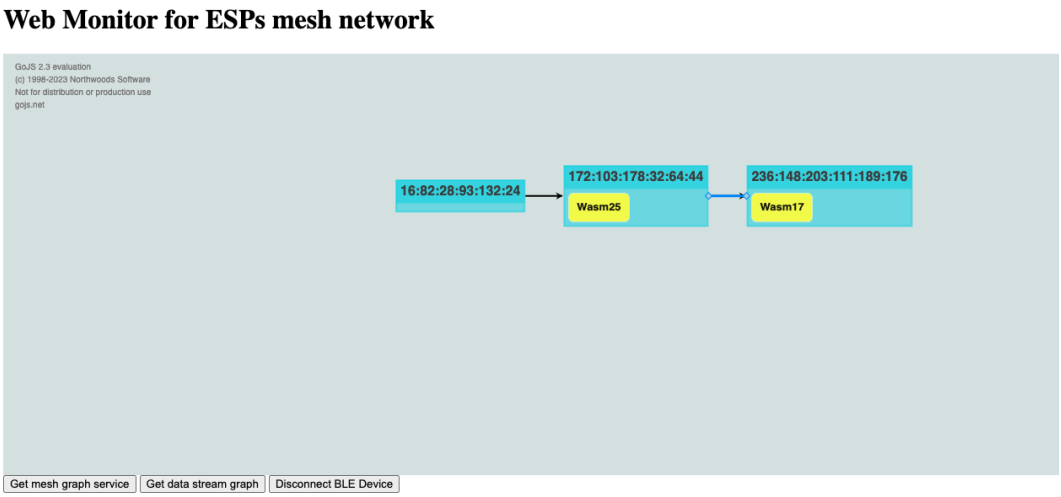

Figure 5: Graph with links

How to upload WASM

1. (Update data stream graph)

2. Click the button "Get WASM upload service"
3. Choose a target node in the graph. It checks whether the selected node has WASM runtime environment.
4. You can write your AssemblyScript code in the text field
5. Click "Compile and load"

Figure 6 shows the coding area and buttons.

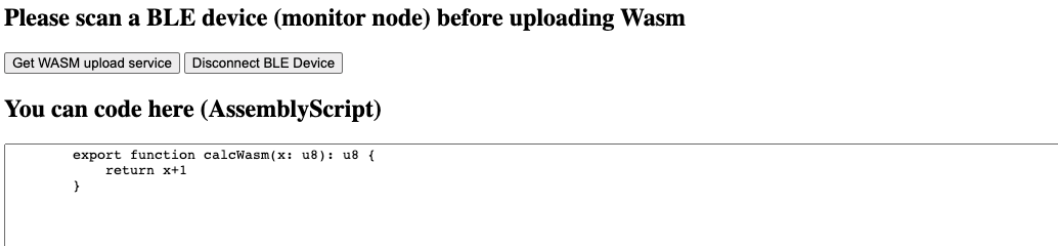

Figure 6: Coding area

5 Links

You find here links these source codes and documentations.

- Implementation of ESP-WiFi-Mesh with Wasm (This repository consists of multiple projects: root, processor, and monitor node): https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo
  - Root node: https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo/tree/main/root_node
  - Documentation of sorce code for the root node: https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo/tree/main/root_node/doxygen
  - Processor node: https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo/tree/main/data_processor
  - Documentation of source code for processor node: https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo/tree/main/data_processor/doxygen

  - Monitor node: https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo/tree/main/mesh_monitor
  - Documantation of source code for monitor node: https://github.com/internet-of-production/Wasm-ESP-IDF-Mesh-Demo/tree/main/mesh_monitor/doxygen
- Web-IDE:
  - Source code: https://github.com/Ayato77/Wasm-ble-web-ide.
  - Web app: https://wasm-ide-for-esp32.onrender.com/ide.html
- Report about various wireless communication ways for microcontrollers: https://git-ce.rwth-aachen.de/iop/workstreams/ws.a3/papers/2022-retrofitting-wasm/-/wikis/home

Other links

- Implementation with painless Mesh (Wifi mesh library for arduino framework): https://github.com/internet-of-production/WasmMeshDemo.
- Experiment with ESP-Now and BLE communication: https://github.com/internet-of-production/WasmMeshESP32
- Communication over OPCUA with configuration and data processing by Wasm in : https://github.com/internet-of-production/retrofit-simulation-with-wasm