cesanta / **mongoose** Public

Code | Issues 4 | Pull requests 5 | Discussions | Actions | Projects | Security and quality | Insights

master | **8** Branches | **58** Tags | Go to file | Go to file | Code

scaprile Merge pull request #3479 from cesanta/vlans 4ff5b60 · 16 hours ago

.github	update tests	last week
src	Add support for 802.1Q VLANs	16 hours ago
test	Merge pull request #3477 from cesanta/mem	17 hours ago
tutorials	Add support for 802.1Q VLANs	16 hours ago
.clang-format	Add badges	6 years ago
LICENSE	Update copyright year in LICENSE file	3 months ago
README.md	Add lwIP vs Mongoose benchmark article link	2 weeks ago
mongoose.c	Add support for 802.1Q VLANs	16 hours ago
mongoose.h	Add support for 802.1Q VLANs	16 hours ago

README | License

Mongoose - Embedded Web Server / Embedded Network Library

License GPLv2 or Commercial | Build and test - essentials passing | codecov 96% | oss-fuzz fuzzing

Mongoose is a network library for C/C++. It provides event-driven non-blocking APIs for TCP, UDP, HTTP, WebSocket, MQTT, and other protocols. It is designed for connecting devices and bringing them online. On the market since 2004, used by vast number of open source and commercial products - it even runs on the International Space Station! Mongoose makes embedded network programming fast, robust, and easy. Features include:

- Cross-platform:
  - works on Linux/UNIX, MacOS, Windows, Android
  - works on ST, NXP, ESP32, Nordic, TI, Microchip, Infineon, Renesas and other chips
  - write code once - and it'll work everywhere
  - ideal for the unification of the network infrastructure code across company
- Built-in protocols: plain TCP/UDP, SNTP, HTTP, MQTT, Websocket, and other
- Asynchronous DNS resolver
- Tiny static and run-time footprint
- Source code is both ISO C and ISO C++ compliant
- Easy to integrate: just copy mongoose.c and mongoose.h files to your source tree
- Built-in TCP/IP stack with drivers for bare metal or RTOS systems
  - Available drivers: STM32F, STM32H; NXP RT1xxx; TI TM4C; Microchip SAME54; Wiznet W5500
  - A complete Web device dashboard on bare metal ST Nucleo boards is only 6 files
  - For comparison, a CubeIDE generated HTTP example is 400+ files
- Can run on top of an existing TCP/IP stack with BSD API, e.g. lwIP, Zephyr, Azure, etc
- Built-in TLS 1.3 ECC stack. Also can use external TLS libraries - mbedTLS, OpenSSL, or other
- Does not depend on any other software to implement networking
- Built-in firmware updates for STM32 H5, STM32 H7

See https://mongoose.ws/ for complete documentation, videos, case studies, etc.

Supported platforms

Mongoose can work on top of any TCP/IP stack that supports BSD sockets API. Platforms supported by the 3rd party TCP/IP stacks:

TCP/IP stack	Notes
lwIP	All devices running lwIP, for example ESP32, ESP32S3, ESP32C3, ESP32C6, etc
Zephyr	All devices supported by Zephyr
Other	Any other TCP/IP stack that supports BSD socket API, for example Amazon FreeRTOS-TCP
Linux, Mac, Windows	Workstations, server, single board computers, embedded Linux devices running on MPUs or FPGAs

Optionally, Mongoose provides its own built-in TCP/IP stack, eliminating the need for additional software to implement networking functionality. The built-in stack supports operation in both bare-metal and RTOS environments. Platforms supported by the Mongoose built-in TCP/IP stack:

Hardware	Notes
STM32	All STM32 MCUs with built-in Ethernet: STM32Fxx, STM32H5xx, STM32H7xx
NXP	All NXP MCUs with built-in Ethernet: IMXRT102x, IMXRT104x, IMXRT105x, IMXRT106x, IMXRT117x, RW612, MCXN94x
Microchip	ATSAME54 MCUs with built-int Ethernet
Renesas	RA5M, RA6M, RA8M MCUs with built-in Ethernet
Infineon	XMC4, XMC7 MCUs with built-in Ethernet
Texas Instruments	TM4C, TMS570 MCUs with built-in Ethernet
Cypress WiFi	Any MCU with CY43xx WiFi chips, like RP2040 Pico-W, RP2350 Pico2-W, Arduino Portenta
Wiznet Ethernet	Any MCU that use Wiznet W5500 or Wiznet 5100 MAC+PHY chips
Cellular	NRF9160, SIM800

Usage Examples

Below are quick snippets that should give an idea how simple the API is and how easy it is to create applications with it.

Create a simple web server that serves a directory. The behavior of the HTTP server is specified by its event handler function:

```c
#include "mongoose.h"   // To build, run: cc main.c mongoose.c

// HTTP server event handler function
void ev_handler(struct mg_connection *c, int ev, void *ev_data) {
  if (ev == MG_EV_HTTP_MSG) {
    struct mg_http_message *hm = (struct mg_http_message *) ev_data;
    struct mg_http_serve_opts opts = { .root_dir = "./web_root/" };
    mg_http_serve_dir(c, hm, &opts);
  }
}

int main(void) {
  struct mg_mgr mgr;  // Declare event manager
  mg_mgr_init(&mgr);  // Initialise event manager
  mg_http_listen(&mgr, "http://0.0.0.0:8000", ev_handler, NULL);  // Setup listener
  for (;;) {          // Run an infinite event loop
    mg_mgr_poll(&mgr, 1000);
  }
  return 0;
}
```

HTTP server implements a REST API that returns current time. JSON formatting:

```
static void ev_handler(struct mg_connection *c, int ev, void *ev_data) {
  if (ev == MG_EV_HTTP_MSG) {
    struct mg_http_message *hm = (struct mg_http_message *) ev_data;
    if (mg_match(hm->uri, mg_str("/api/time/get"), NULL)) {
      mg_http_reply(c, 200, "", "{%m:%lu}\n", MG_ESC("time"), time(NULL));
    } else {
      mg_http_reply(c, 500, "", "{%m:%m}\n", MG_ESC("error"), MG_ESC("Unsupported URI"));
    }
  }
}
```

MQTT client that subscribes to a topic `device1/rx` and echoes incoming messages to `device1/tx`:

```
#include "mongoose.h"

static const char *s_mqtt_url = "mqtt://broker.hivemq.com:1883";
static struct mg_connection *s_mqtt_conn = NULL;

// MQTT connection event handler function
static void ev_handler(struct mg_connection *c, int ev, void *ev_data) {
  if (ev == MG_EV_OPEN) {
    MG_INFO(("%lu created, connecting to %s ...", c->id, s_mqtt_url));
  } else if (ev == MG_EV_MQTT_OPEN) {
    struct mg_mqtt_opts opts = {.qos = 1, .topic = mg_str("device1/rx")};
    mg_mqtt_sub(c, &opts);
    MG_INFO(("%lu connected, subscribing to %s", c->id, opts.topic.buf));
  } else if (ev == MG_EV_MQTT_MSG) {
    char response[100];
    struct mg_mqtt_message *mm = (struct mg_mqtt_message *) ev_data;
    struct mg_mqtt_opts opts = {.qos = 1, .topic = mg_str("device1/tx")};
    mg_snprintf(response, sizeof(response), "Received [%.*s] / [%.*s]",
                mm->topic.len, mm->topic.buf, mm->data.len, mm->data.buf);
    opts.message = mg_str(response);
    mg_mqtt_pub(c, &opts);
  } else if (ev == MG_EV_CLOSE) {
    MG_INFO(("%u closing", c->id));
    s_mqtt_conn = NULL;
  }
}

// Reconnection timer function. If we get disconnected, reconnect again
static void timer_fn(void *arg) {
  struct mg_mgr *mgr = (struct mg_mgr *) arg;
  if (s_mqtt_conn == NULL) {
    struct mg_mqtt_opts opts = {.clean = true};
    s_mqtt_conn = mg_mqtt_connect(mgr, s_mqtt_url, &opts, ev_handler, NULL);
  }
}

int main() {
  struct mg_mgr mgr;  // Mongoose event manager. Holds all connections
  mg_mgr_init(&mgr);  // Initialise event manager
  mg_timer_add(&mgr, 3000, MG_TIMER_REPEAT | MG_TIMER_RUN_NOW, timer_fn, &mgr);
  for (;;) {
    mg_mgr_poll(&mgr, 1000);  // Infinite event loop
  }
  return 0;
}
```

Commercial use

- Mongoose is used by hundreds of businesses, from Fortune500 giants like Siemens, Schneider Electric, Broadcom, Bosch, Google, Samsung, Qualcomm, Caterpillar to the small businesses
- Used to solve a wide range of business needs, like implementing Web UI interface on devices, RESTful API services, telemetry data exchange, remote control for a product, remote software updates, remote monitoring, and others

- Deployed to hundreds of millions devices in production environment worldwide
- See Case Studies from our respected customers like Schneider Electric (industrial automation), Broadcom (semiconductors), Pilz (industrial automation), and others
- See Testimonials from engineers that integrated Mongoose in their commercial products
- We provide Evaluation and Commercial licensing, support, consultancy and integration services - don't hesitate to contact us

Security

We take security seriously:

1. Mongoose repository runs a continuous integration test powered by GitHub, which runs through hundreds of unit tests on every commit to the repository. Our unit tests are built with modern address sanitizer technologies, which help to find security vulnerabilities early
2. Mongoose repository is integrated into Google's oss-fuzz continuous fuzzer which scans for potential vulnerabilities continuously
3. We receive periodic vulnerability reports from the independent security groups like Cisco Talos, Microsoft Security Response Center, MITRE Corporation, Compass Security and others. In case of the vulnerability found, we act according to the industry best practice: hold on to the publication, fix the software and notify all our customers that have an appropriate subscription
4. Some of our customers (for example NASA) have specific security requirements and run independent security audits, of which we get notified and in case of any issue, act similar to (3).

How to report security vulnerabilities

Please send an email to support@cesanta.com, with the full information. Do NOT create a github issue.

Articles

Technical guides and deep dives into embedded web servers, WebUI integration and embedded networking technologies:

- Embedded Web Server: A Comprehensive Guide for Modern Connected Devices
- Building Embedded Web Device Dashboards
- ESP32 Device Dashboard: A Step-by-Step Guide for Developers
- How to build an STM32 Web Dashboard
- STM32 WebSocket Guide
- Web File Manager on STM32, ESP32 and Embedded Linux
- Web dashboard on Zephyr RTOS
- Limiting TCP/IP RAM usage on STM32
- STM32 Ethernet explained
- MQTT on a Microcontroller
- STM32 OTA Firmware Update
- RP2350 OTA Firmware Update
- STM32 Ethernet and caches
- NXP RW612 OTA Firmware Update
- lwIP vs Mongoose - TCP/IP Stack Integration Benchmark

Contributions

Contributions are welcome! Please follow the guidelines below:

- Sign Cesanta CLA and send GitHub pull request

Releases 50

7.21 Latest
20 hours ago

+ 49 releases

Packages

No packages published

Contributors 125

+ 111 contributors

Languages

C 98.3% **Assembly** 0.6% **CMake** 0.3% **C++** 0.3% **Makefile** 0.3% **JavaScript** 0.1% **Other** 0.1%