ДОДАТОК А

Апробація результатів роботи

УДК 004.054; 004.055

АНАЛІЗ МОЖЛИВОСТІ ІНТЕГРАЦІЇ МІКРОКОНТРОЛЛЕРА НА БАЗІ STM32 В СУЧАСНІ ПРИЛАДИ КЕРУВАННЯ СВІТЛОФОРАМИ

Волонiхiн В.Д.
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: vadym.volonikhin@nure.ua

У роботі досліджено можливість інтеграції мікроконтролера на базі STM32 у сучасні прилади керування світлофорами. Проведено аналіз існуючих апаратних платформ (STM32 Nucleo/Discovery), сенсорних інтерфейсів, зокрема індукційних петель SWARCO , тепловізійних камер FLIR і пішохідних кнопок RTB , а також протоколів зв'язку DSRC/IEEE 802.11p для V2X-комунікацій. Оцінено ефективність реалізації адаптивної фазової логіки на основі STM32 (фіксовані та змінні цикли) і методи забезпечення відмовостійкості та аварійних сценаріїв згідно з вимогами DIN V VDE V 0823-500 та рекомендаціями RiLSA.

Ключові слова: STM32, світлофор, інтеграція, надійність, сенсори, адаптивне керування.

ANALYSIS OF THE POSSIBILITY OF INTEGRATION OF A STM32-BASED MICROCONTROLLER INTO MODERN TRAFFIC LIGHT CONTROL DEVICES

Volonikhin V.D.
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky ave. 14
E-mail: vadym.volonikhin@nure.ua

This work investigates the feasibility of integrating an STM32-based microcontroller into modern traffic signal control devices. An analysis was performed on existing hardware platforms (STM32 Nucleo/Discovery boards), sensor interfaces such as SWARCO inductive loops , FLIR TrafiOne thermal cameras , and RTB pedestrian push buttons , as well as DSRC/IEEE 802.11p protocols for V2X communication. The effectiveness of implementing adaptive phase logic on the STM32 platform (both fixed and variable cycles) was evaluated, along with fault-tolerance mechanisms and emergency scenarios in accordance with DIN V VDE V 0823-500 requirements and RiLSA guidelines .

Keywords: STM32, traffic signal, integration, reliability, sensors, adaptive control.

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ. Зважаючи на постійне зростання обсягів автомобільного потоку в місті, питання надійності й безпеки електронних систем керування світлофорами набуває особливої важливості. Сучасні перехрестя обслуговують тисячі автомобілів і пішоходів щодня, і будь-які збої в роботі контролера чи сенсорів (індукційні петлі, камери, пішохідні кнопки) можуть призводити до заторів, аварій або навіть соціальних конфліктів. Відповідність жорстким вимогам стандартів DIN V VDE V 0823-500 і рекомендацій RiLSA є ключовим фактором при проєктуванні таких систем, оскільки вони визначають критерії відмовостійкості, електромагнітної сумісності й аварійних сценаріїв.

ВСТУП. Для того щоб забезпечити ефективне й безперервне регулювання міського трафіку, необхідно враховувати такі основні загрози:

– відмови електроживлення або мережі зв'язку, які можуть призводити до повного знеструмлення контролера;

– електромагнітні перешкоди (EMC-проблеми), здатні впливати на точність роботи таймерів і інтерфейсів мікроконтролера;

– відмова сенсорів, наприклад індукційних петель чи відеокамер, що призводить до некоректної детекції транспортних засобів;

– кіберзагрози (атаки на протоколи V2X, DSRC / IEEE 802.11p), які можуть порушити обмін даними між вузлами інфраструктури.

Також передбачена матеріально-технічна база:

– апаратна база: мікроконтролер STM32H743VIT6 (ARM Cortex-M7, 400 МГц), індукційні петлі SWARCO FUTURIT, тепловізійні камери FLIR TrafiOne, пішохідні кнопки RTB APS із вібро- та акустичною індикацією.

– програмні інструменти: STM32CubeMX для конфігурації периферії, STM32CubeIDE для розробки коду, EasyEDA для моделювання електричних схем і таймінгів світлофорних фаз.

– нормативно-методичні джерела: DIN V VDE V 0823-500 (електротехнічні та програмні вимоги до приладів керування світлофорами), RiLSA (методики розташування сигналів, фазова логіка, «час затухання»), ISO/TS 14812 (термінологія ITS).

Формули для фазової логіки світлофора:

1) цикл роботи світлофора:

$$C = \frac{1.5L + 5}{1 - \sum_i Yi}$$

де C – оптимальна тривалість циклу(с), L – загальна тривалість втрат часу на старт/затухання (с), Y_i – ступінчасті насичення потоку для кожної фази

2) Розрахунок часу затухання:

$$R(clear) = \frac{d}{\upsilon(vehicle)}$$

де d – відстань, яку має подолати транспортний засіб для очищення перехрестя, $\upsilon_{vehicle}$ – середня швидкість транспортних засобів під час виїзду (м/с)

Після визначення часових інтервалів і алгоритмів фазової логіки доцільно зосередитися на ключових характеристиках мікроконтролера STM32H743VIT6, які роблять його оптимальним вибором для сучасних пристроїв керування світлофорами:

а) висока продуктивність:

– ARM Cortex-M7 зі швидкістю до 400 МГц забезпечує можливість виконувати складні обчислення в реальному часі — адаптивне регулювання фаз, обробку даних з камер FLIR та індукційних петель без затримок;

– вбудований двочастотний FPU (floating-point unit) прискорює математику з плаваючою комою, що корисно при розрахунку коефіцієнта завантаження й алгоритмах оптимізації цикла.

б) великий об'єм пам'яті:

– 2 МБ Flash і 1 МБ SRAM дають змогу розмістити як код керування світлофорами, так і буфери для зберігання кадрів з відеокамер та даних сенсорів;

– інтегрований ART Accelerator скорочує час доступу до Flash і прискорює виконання критичних завдань.

в) розвинені периферійні інтерфейси:

– до 3 каналів FMPI2C, 6 SPI та 8 USART для підключення індукційних петель, камер, пішохідних кнопок і модулів V2X;

– таймери з підтримкою "дед-тайму" та PWM-генерації дозволяють точно формувати сигнали керування для LED-індикаторів;

– два контролери Ethernet і CAN FD дають змогу реалізувати високошвидкісну та надійну мережеву взаємодію між перехрестями.

г) низьке енергоспоживання та надійність:

– режими Sleep/Stop зі споживанням від 50 мА гарантують енергоефективність навіть при автономному живленні;

– вбудовані блоки BOR, POR, мультилефазний WDT та ECC-перевірка пам'яті запобігають неконтрольованим відмовам у складних умовах експлуатації.

д) функції кібербезпеки та безпеки функціональної:

– апаратна підтримка TrustZone та AES-шифрування для захисту прошивки та каналів зв'язку V2X;

– відповідність вимогам стандарту IEC 61508 (SIL 3) та DIN V VDE V 0823-500 щодо самодіагностики і аварійного переходу в безпечний стан.

Формула періоду таймера (Timer):

$$T = \frac{(PSC + 1)(ARR + 1)}{F(TIM)}$$

де *PSC(prescaler)* – значення прескалера, ARR(Auto-Reload Register) – значення в регістрі автоперезавантаження, F_{Tim} – тактова частота таймера (частота шини APB, помножена на 1 або 2 залежно від конфігурації)

У таблиці 1 нижче наведено порівняння аналогів та можливих процесорів та мікроконтролерів які використовуються в актуальних приладах керування трафіком .

Таблиця 1 – Порівняння найпоширеніших процесорів та мікроконтроллерів для сучасних приладів керування світлофорами.

Процесор/мікроконтролер	Архітектура та ключові характеристики	Ціна за штуку($)
1	2	3
STM32H743VIT6	ARM Cortex-M7 @ 480 МГц; 2 МБ Flash, 1 МБ SRAM; DSP, FPU, Art Accelerator, L1-кеш; Вбудований контролер зовнішньої пам'яті, Ethernet, CAN-FD, USB, високошвидкісні інтерфейси.	16
NXP S32K344	ARM Cortex-M7 @ до 160 МГц (двоядерний); Сертифікація ISO 26262 ASIL-D; Вбудовані CAN, CAN-FD, LIN, Ethernet, апаратна безпека	21
TI TMS570LC4357	Два ARM Cortex-R5F у lock-step @ 300 МГц; 4 МБ Flash, 512 КБ RAM; Сертифікація IEC 61508 SIL-3, ISO 26262 ASIL-D	35

Продовження таблиці 1

1	2	3
Infineon AURIX TC275	3-ядерний TriCore™ @ 200 МГц + DSP; 4 МБ Flash з ECC, 472 КБ RAM; ISO 26262 ASIL-D, вбудований HSM;	~600
Renesas R-Car H3e	Quad ARM Cortex-A57 @ 2 ГГц + Quad Cortex-A53 @ 1.2 ГГц; Dual Cortex-R7 для реального часу ISO 26262 ASIL-B	Н/д*
Microchip SAMA5D27	ARM Cortex-A5 @ 500 МГц; Готовий модуль: DDR2, Ethernet PHY, Flash, PMIC; Підтримка real-time Linux	38
Xilinx Zynq-7000 XC7Z010	Dual ARM Cortex-A9 @ 667 МГц + FPGA Artix-7; 28 K LUT, HP-/ACP-інтерфейси для детермінованості; Аналіз відеопотоків у реальному часі	18

Основні переваги мікроконтролера на базі STM32.

– висока продуктивність ядра

Мікроконтролери STM32H7 засновані на ядрі ARM Cortex-M7 з тактовою частотою до 480 МГц та вбудованим FPU і DSP-оптимізаціями для швидкої обробки сигналів і математичних операцій.

– низьке енергоспоживання

Різноманітні режими енергозбереження (Stop, Standby, Low-power Run) та технологія динамічного масштабування напруги дають змогу зменшити споживання до кількох мікроамперів, що критично для енергонезалежних систем

– широкий набір периферійних інтерфейсів. Підтримка USART, SPI, I²C, CAN-FD, USB, Ethernet, ADC/DAC, таймерів та інших інтерфейсів дозволяє реалізувати складну систему керування світлофорами без додаткових контролерів;

– розвинена екосистема розробки. Інструменти STM32CubeMX (графічне налаштування периферії і генерація коду) та STM32CubeIDE (редактор, компілятор, відлагодження) спрощують і прискорюють розробку програмного забезпечення;

– масштабованість і портабельність коду. Уніфікована модель регістрів і API HAL/LL по всіх сімействах STM32 забезпечує зручний перенос проєктів між різними мікроконтролерами з мінімальними змінами коду;

– функції безпеки та криптографії. Моделі серії STM32L5/M33 із TrustZone, вбудовані криптографічні прискорювачі та механізми захисту пам'яті (MPU, PC-ROP) дозволяють реалізувати безпечні сценарії обробки конфіденційних даних;

– покращена обробка пам'яті через кеш і prefetch. Кеш L1 і механізми prefetch у STM32H7-сімействі знижують затримки доступу до Flash-пам'яті й підвищують загальну продуктивність додатків;

– гнучкі умови живлення. Діапазон живлення 1,7 … 3,6 В дозволяє застосовувати STM32 у різних системах, включаючи автономні батарейні рішення та живлення від сонячних панелей;

– вбудований DMA-контролер. Підтримка DMA звільняє процесор від обробки великих

ДОДАТОК Б

Лістинг матеріалів друкованої плати (BILL OF MATERIALS):

Name		Designator	Footprint
100nF		C1,C2,C3,C4,C5,C6,C7,C8,C9,C17, C21,C31,C32,C33,C34,C35	C0805
100nF		C10,C11,C15,C16	CASE-A_3216
10uF		C12,C13,C20	CAP-SMD_L3.5-W2.8-R-RD
2.2uF		C14	CAP-SMD_L3.2-W1.6-R-RD
10nF		C18,C29,C30	C0805
4.7uF		C19	C0805
18pF		C22,C23	C0805
1uF		C24	CASE-A_3216
22pF		C36,C37	C1206
100nF		C38,C39	C1206
1N4148W T4		D1,D2,D3,D4,D5,D6,D7,D8,D9, D10,D11,D12,D14,D16	SOD-123_L2.8-W1.8-LS3.7-RD
BZX84C3V3LT1G		D13	SOT-23-3_L2.9-W1.6-P1.90-LS2.8-BR
PZ254-2-03-Z-8.5		H1	HDR-TH_6P-P2.54-V-M-R2-C3-S2.54
JUMPER		JP1,JP2,JP3,JP4,JP5,JP6,JP7,JP8,JP9,JP10	JUMPER_FOOTPRINT
HK115FD-DC5V-SG		K1	RELAY-TH_HK115FH-DCX-X-XX
HI1206P121R-10		L1	L1206
12V		LED1	OSRAM_LR_A67F
SiSi Scharf		LED2	OSRAM_LR_A67F
1 Herz Impuls		LED4	OSRAM_LR_A67F
LED5		LED5	OSRAM_LR_A67F
09 18 510 6904		P1	918510X904
BCX70H,215		Q1,Q2,Q3,Q4,Q5,Q6	SOT-23-3_L2.9-W1.3-P1.90-LS2.4-BR
2N7002		Q7	SOT-23-3_L2.9-W1.3-P1.90-LS2.4-BR
10k		R1,R2,R12	R0603
	162	R3,R23	R1206
1K		R4,R6,R7,R8,R9,R16	R1206
10K		R5,R10,R11,R24,R40,R41,R42	R1206
	510	R13,R14	R1206
	560	R15	R1206
	470	R19	R1206
	150	R20	R1206
1MΩ		R21,R22	R1206
12.4K		R37	R1206
1.47M		R50	R1206
	36	R56	R1206
	82	R57	R1206

10K	R58,R59	R0603
Resistor Network	RN1,RN2,RN3,RN4,RN6	RESNET1206
Resistot Network	RN5	RESNET1206
DevEBox-STM32H743VI	U1	DEVEBOX_STM32H743
PCA9555D,112	U2,U3,U9,U12	SO-24_L15.6-W7.6-P1.27-LS10.6-BL
CD74HC245M96	U4,U5,U6,U7,U19	SOIC-20_L12.8-W7.5-P1.27-LS10.3-BL
MAX485CSA+T-JSM	U8	SOP-8_L4.9-W3.9-P1.27-LS6.0-BL
ATMEGA328P-AU	U10	TQFP-32_L7.0-W7.0-P0.80-LS9.0-BL
TXS0108EPWR	U11,U16	TSSOP-20_L6.5-W4.4-P0.65-LS6.4-BL
MM74HCT00MX	U13,U14	SOIC-14_L8.7-W3.9-P1.27-LS6.0-BL
PCF8574PW	U15	TSSOP-20_L6.5-W4.4-P0.65-LS6.4-BL
W5500	U17	LQFP-48_L7.0-W7.0-P0.50-LS9.0-BL
RJMG163118101NR	U18	RJMG16XXXXXX1XX
MAX3232IPW	U20	TSSOP-16_L5.0-W4.4-P0.65-LS6.4-BL
ZX-TYPEA-2-WTM4	USB1	USB-A-SMD_ZX-TYPEA-2-WTM4
25MHZ	X3	OSC-SMD_4P-L5.0-W3.2-BL
8MHz	X4	CRYSTAL-SMD_4P-L3.2-W2.5-BL
5535043-4	ST1	DIN41612_96WAY_RA

ДОДАТОК В

Лістинг коду для CUBE IDE:

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
  * @file           : main.c
  * @brief          : Main program body

******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "fatfs.h"
#include "i2c.h"
```

```
#include "memorymap.h"
#include "sdmmc.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"
#include <string.h>
#include <stdbool.h>
#include <stdio.h>
#include "socket.h"
#include "dhcp.h"
#include "httpServer.h"
#include "fonts.h"
#include "ssd1306.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
```

```
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
char ip_str[16];
extern UART_HandleTypeDef huart2;
uint8_t sdMountedFlag = 0;
char tick_buf[32];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MPU_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

uint8_t buff[64];

#define DHCP_SOCKET     0
#define DNS_SOCKET      1
#define HTTP_SOCKET     2
#define SOCK_TCPS       0
#define SOCK_UDPS       1
#define PORT_TCPS       5000
```

```
#define PORT_UDPS       3000
#define MAX_HTTPSOCK    6
#define DHCP_SOCKET     0
#define DNS_SOCKET      1
#define HTTP_SOCKET     2
#define SOCK_TCPS       0
#define SOCK_UDPS       1
#define PORT_TCPS       5000
#define PORT_UDPS       3000
#define MAX_HTTPSOCK    6
#define index_page "<!DOCTYPE html>"\
 "<html>"\
  "<head>"\
   "<title>W5500-STM32 Web Server</title>"\
   "<meta http-equiv='Content-Type' content='text/html; charset=utf-8'>"\
   "<link href=\"\" rel=\"icon\" type=\"image/x-
icon\">"\
   "<style>"\
    "html {display: inline-block; margin: 0px auto; text-align: center;}"\
    "body{margin-top: 50px;}"\
    ".button {display: block;"\
     "width: 70px;"\
     "background-color: #008000;"\
     "border: none;"\
     "color: white;"\
     "padding: 14px 28px;"\
     "text-decoration: none;"\
     "font-size: 24px;"\
     "margin: 0px auto 36px;"\
     "border-radius: 5px;}"\
```

```
      ".button-on {background-color: #008000;}"\
      ".button-on:active{background-color: #008000;}"\
      ".button-off {background-color: #808080;}"\
      ".button-off:active {background-color: #808080;}"\
      "p {font-size: 20px;color: #808080;margin-bottom: 20px;}"\
    "</style>"\
  "</head>"\
  "<body>"\
    "<h1>STM32 - W5500</h1>"\
    "<p>Control the light via Ethernet</p>"\
    "<a class=\"button button-on\" href=\"/ledon.html\">ON</a>"\
    "<a class=\"button button-off\" href=\"/ledoff.html\">OFF</a>"\
  "</body>"\
 "</html>"

#define ledon_page" <!DOCTYPE html>"\
 "<html>"\
  "<head>"\
    "<title>W5500-STM32 Web Server</title>"\
    "<meta http-equiv='Content-Type' content='text/html; charset=utf-8'>"\
    "<link href=\"\" rel=\"icon\" type=\"image/x-
icon\">"\
    "<style>"\
      "html {display: inline-block; margin: 0px auto; text-align: center;}"\
      "body{margin-top: 50px;}"\
      ".button {display: block;"\
        "width: 70px;"\
        "background-color: #008000;"\
        "border: none;"\
        "color: white;"\
```

```
        "padding: 14px 28px;"\
        "text-decoration: none;"\
        "font-size: 24px;"\
        "margin: 0px auto 36px;"\
        "border-radius: 5px;}"\
       ".button-on {background-color: #008000;}"\
       ".button-on:active{background-color: #008000;}"\
       ".button-off {background-color: #808080;}"\
       ".button-off:active {background-color: #808080;}"\
       "p {font-size: 20px;color: #808080;margin-bottom: 20px;}"\
     "</style>"\
   "</head>"\
   "<body>"\
     "<h1>STM32 - W5500</h1>"\
     "<p>Light is currently on</p>"\
     "<a class=\"button button-off\" href=\"/ledoff.html\">OFF</a>"\
   "</body>"\
  "</html>"

#define ledoff_page"<!DOCTYPE html>"\
  "<html>"\
    "<head>"\
     "<title>W5500-STM32 Web Server</title>"\
     "<meta http-equiv='Content-Type' content='text/html; charset=utf-8'>"\
     "<link href=\"\" rel=\"icon\" type=\"image/x-
icon\">"\
     "<style>"\
       "html {display: inline-block; margin: 0px auto; text-align: center;}"\
       "body{margin-top: 50px;}"\
       ".button {display: block;"\
```

```
        "width: 70px;"\
        "background-color: #008000;"\
        "border: none;"\
        "color: white;"\
        "padding: 14px 28px;"\
        "text-decoration: none;"\
        "font-size: 24px;"\
        "margin: 0px auto 36px;"\
        "border-radius: 5px;}"\
      ".button-on {background-color: #008000;}"\
      ".button-on:active{background-color: #008000;}"\
      ".button-off {background-color: #808080;}"\
      ".button-off:active {background-color: #808080;}"\
      "p {font-size: 20px;color: #808080;margin-bottom: 20px;}"\
    "</style>"\
  "</head>"\
  "<body>"\
    "<h1>STM32 - W5500</h1>"\
    "<p>Light is currently off</p>"\
    "<a class=\"button button-on\" href=\"/ledon.html\">ON</a>"\
  "</body>"\
 "</html>"

uint8_t socknumlist[] = {2, 3, 4, 5, 6, 7};
uint8_t RX_BUF[1024];
uint8_t TX_BUF[1024];
wiz_NetInfo net_info = {
   .mac  = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED },
   .dhcp = NETINFO_DHCP
};
```

```
void wizchipSelect(void) {
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
}

void wizchipUnselect(void) {
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
}

void wizchipReadBurst(uint8_t* buff, uint16_t len) {
   HAL_SPI_Receive(&hspi1, buff, len, HAL_MAX_DELAY);
}

void wizchipWriteBurst(uint8_t* buff, uint16_t len) {
   HAL_SPI_Transmit(&hspi1, buff, len, HAL_MAX_DELAY);
}

uint8_t wizchipReadByte(void) {
   uint8_t byte;
   wizchipReadBurst(&byte, sizeof(byte));
   return byte;
}

void wizchipWriteByte(uint8_t byte) {
   wizchipWriteBurst(&byte, sizeof(byte));
}

volatile bool ip_assigned = false;

void Callback_IPAssigned(void) {
```

```
    ip_assigned = true;
}

void Callback_IPConflict(void) {
    ip_assigned = false;
}

uint8_t dhcp_buffer[1024];
uint8_t dns_buffer[1024];

void W5500Init() {
    // Register W5500 callbacks
    reg_wizchip_cs_cbfunc(wizchipSelect, wizchipUnselect);
    reg_wizchip_spi_cbfunc(wizchipReadByte, wizchipWriteByte);
    reg_wizchip_spiburst_cbfunc(wizchipReadBurst, wizchipWriteBurst);

    uint8_t rx_tx_buff_sizes[] = {2, 2, 2, 2, 2, 2, 2, 2};
    wizchip_init(rx_tx_buff_sizes, rx_tx_buff_sizes);

    // set MAC address before using DHCP
    setSHAR(net_info.mac);
    DHCP_init(DHCP_SOCKET, dhcp_buffer);

    reg_dhcp_cbfunc(
        Callback_IPAssigned,
        Callback_IPAssigned,
        Callback_IPConflict
    );

    uint32_t ctr = 10000;
```

```
    while((!ip_assigned) && (ctr > 0)) {
        DHCP_run();
        ctr--;
    }
    if(!ip_assigned) {
        return;
    }

    getIPfromDHCP(net_info.ip);
    getGWfromDHCP(net_info.gw);
    getSNfromDHCP(net_info.sn);

//    char charData[200]; // Data holder
//
sprintf(charData,"IP:  %d.%d.%d.%d\r\nGW:  %d.%d.%d.%d\r\nNet: %d.%d.%d.%
d\r\n",
//        net_info.ip[0], net_info.ip[1], net_info.ip[2], net_info.ip[3],
//        net_info.gw[0], net_info.gw[1], net_info.gw[2], net_info.gw[3],
//        net_info.sn[0], net_info.sn[1], net_info.sn[2], net_info.sn[3]
//    );
//    HAL_UART_Transmit(&huart1,(uint8_t *)charData,strlen(charData),1000);


    wizchip_setnetinfo(&net_info);
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
```

```
 */
int main(void)
{

 /* USER CODE BEGIN 1 */

 /* USER CODE END 1 */

 /* MPU Configuration--------------------------------------------------------*/
 MPU_Config();

 /* MCU Configuration--------------------------------------------------------*/

 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 HAL_Init();

 /* USER CODE BEGIN Init */

 /* USER CODE END Init */

 /* Configure the system clock */
 SystemClock_Config();

 /* USER CODE BEGIN SysInit */

 /* USER CODE END SysInit */

 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_SPI1_Init();
```

```
MX_USART2_UART_Init();
MX_SDMMC1_SD_Init();
MX_FATFS_Init();
MX_I2C1_Init();
/* USER CODE BEGIN 2 */



SSD1306_Init();
SSD1306_Clear();
FRESULT res = f_mount (&SDFatFS, SDPath, 1);
if (res == FR_OK)
{
    SSD1306_Puts("SD MOUNT OK", &Font_11x18, 1);
    sdMountedFlag = 1;
  res = f_open(&SDFile, "test.txt", FA_WRITE | FA_OPEN_ALWAYS);
  if (res == FR_OK)
  {
     UINT i;
     f_write(&SDFile, "TEST\r\n", 6, &i);
     f_close(&SDFile);
  }
  res = f_open(&SDFile, "test.txt", FA_READ);
     if (res == FR_OK)
     {
            UINT i;
            f_read(&SDFile, buff, 6, &i);
            f_close(&SDFile);
}
}
```

```
else
  {
     SSD1306_Puts("SD MOUNT FAIL", &Font_11x18, 1);
     sdMountedFlag = 0;
  }
  SSD1306_UpdateScreen();
  HAL_Delay(1000);

W5500Init();
SSD1306_Clear();
  SSD1306_GotoXY(0, 0);
  if (sdMountedFlag == 1)
  {
     SSD1306_Puts("SD MOUNT OK", &Font_11x18, 1);
  }
  else
  {
     SSD1306_Puts("SD MOUNT FAIL", &Font_11x18, 1);
  }

  SSD1306_GotoXY(0, 30);
  if (ip_assigned)
  {
     sprintf(ip_str, "%d.%d.%d.%d",
           net_info.ip[0], net_info.ip[1],
           net_info.ip[2], net_info.ip[3]);
     SSD1306_Puts(ip_str, &Font_7x10, 1);
  }
  else
  {
```

```
    SSD1306_Puts("DHCP FAIL", &Font_11x18, 1);
  }
  SSD1306_UpdateScreen();
  HAL_Delay(2000);

  httpServer_init(TX_BUF, RX_BUF, MAX_HTTPSOCK, socknumlist);
  reg_httpServer_cbfunc(NVIC_SystemReset, NULL);
  /* Web content registration */
  reg_httpServer_webContent((uint8_t *)"index.html", (uint8_t *)index_page);
  reg_httpServer_webContent((uint8_t *)"ledon.html", (uint8_t *)ledon_page);
  reg_httpServer_webContent((uint8_t *)"ledoff.html", (uint8_t *)ledoff_page);

  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
/* USER CODE END 2 */

 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
  HAL_Delay(100);
 while (1)
 {
  /* USER CODE END WHILE */
     uint32_t t = HAL_GetTick();
        int len = sprintf(tick_buf, "Tick: %lu\r\n", (unsigned long)t);
                        HAL_UART_Transmit(&huart2,  (uint8_t*)tick_buf,  len,
HAL_MAX_DELAY);
        HAL_Delay(1000);
  /* USER CODE BEGIN 3 */
      }
  /* USER CODE BEGIN 3 */
```

```
  }

  /* USER CODE END 3 */


/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Supply configuration update enable
  */
  HAL_PWREx_ConfigSupply(PWR_LDO_SUPPLY);

  /** Configure the main internal regulator output voltage
  */

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE
_SCALE3);

  while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
```

```
  RCC_OscInitStruct.HSIState = RCC_HSI_DIV1;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 12;
  RCC_OscInitStruct.PLL.PLLP = 2;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  RCC_OscInitStruct.PLL.PLLR = 2;
  RCC_OscInitStruct.PLL.PLLRGE = RCC_PLL1VCIRANGE_3;
  RCC_OscInitStruct.PLL.PLLVCOSEL = RCC_PLL1VCOWIDE;
  RCC_OscInitStruct.PLL.PLLFRACN = 4096;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2

|RCC_CLOCKTYPE_D3PCLK1|RCC_CLOCKTYPE_D1PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.SYSCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB3CLKDivider = RCC_APB3_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV2;
```

```
  RCC_ClkInitStruct.APB2CLKDivider = RCC_APB2_DIV1;
  RCC_ClkInitStruct.APB4CLKDivider = RCC_APB4_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) !=
HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

 /* MPU Configuration */

void MPU_Config(void)
{
  MPU_Region_InitTypeDef MPU_InitStruct = {0};

  /* Disables the MPU */
  HAL_MPU_Disable();

  /** Initializes and configures the Region and the memory to be protected
  */
  MPU_InitStruct.Enable = MPU_REGION_ENABLE;
  MPU_InitStruct.Number = MPU_REGION_NUMBER0;
  MPU_InitStruct.BaseAddress = 0x0;
  MPU_InitStruct.Size = MPU_REGION_SIZE_4GB;
  MPU_InitStruct.SubRegionDisable = 0x87;
```

```
  MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
  MPU_InitStruct.AccessPermission = MPU_REGION_NO_ACCESS;
  MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
  MPU_InitStruct.IsShareable = MPU_ACCESS_SHAREABLE;
  MPU_InitStruct.IsCacheable = MPU_ACCESS_NOT_CACHEABLE;
  MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;

  HAL_MPU_ConfigRegion(&MPU_InitStruct);
  /* Enables the MPU */
  HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);

}

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
```

```
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

ДОДАТОК Г

Демонстраційний матеріал

Формат	Позиція	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кіль-кість	Примітки
			Текстові документи		
А4	1	ГЮИК ХХХХХХ.004 ПЗ	Пояснювальна записка	60 с.	
			Інші документи		
А4	2		Лістинг коду для CUBE IDE	19 с.	
А4	3		Лістинг матеріалів друкованої	2 с.	
			плати (BILL OF MATERIALS)		
А4	4		Демонстраційний матеріал	13 с.	

					ГЮИК ХХХХХХ.004 ВД			
мін.	А	Номер докум.	Підпис	Дата				
Розроб.		Волоніхін В.Д.			Розроблення автоматизованої системи контролю міського трафіку на базі мікроконтролера stm32	Літера	Аркуш	Аркуш
Перевір.		Демська Н.П				Н	1	1
						едра КІТАР ХНУРЕ		
Н.контр.		Демська Н.П.						
Затв.		Невлюдов І.Ш.						