



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

ELEKTRONINIŲ SISTEMŲ KATEDRA

LABORATORINIS DARBAS 1

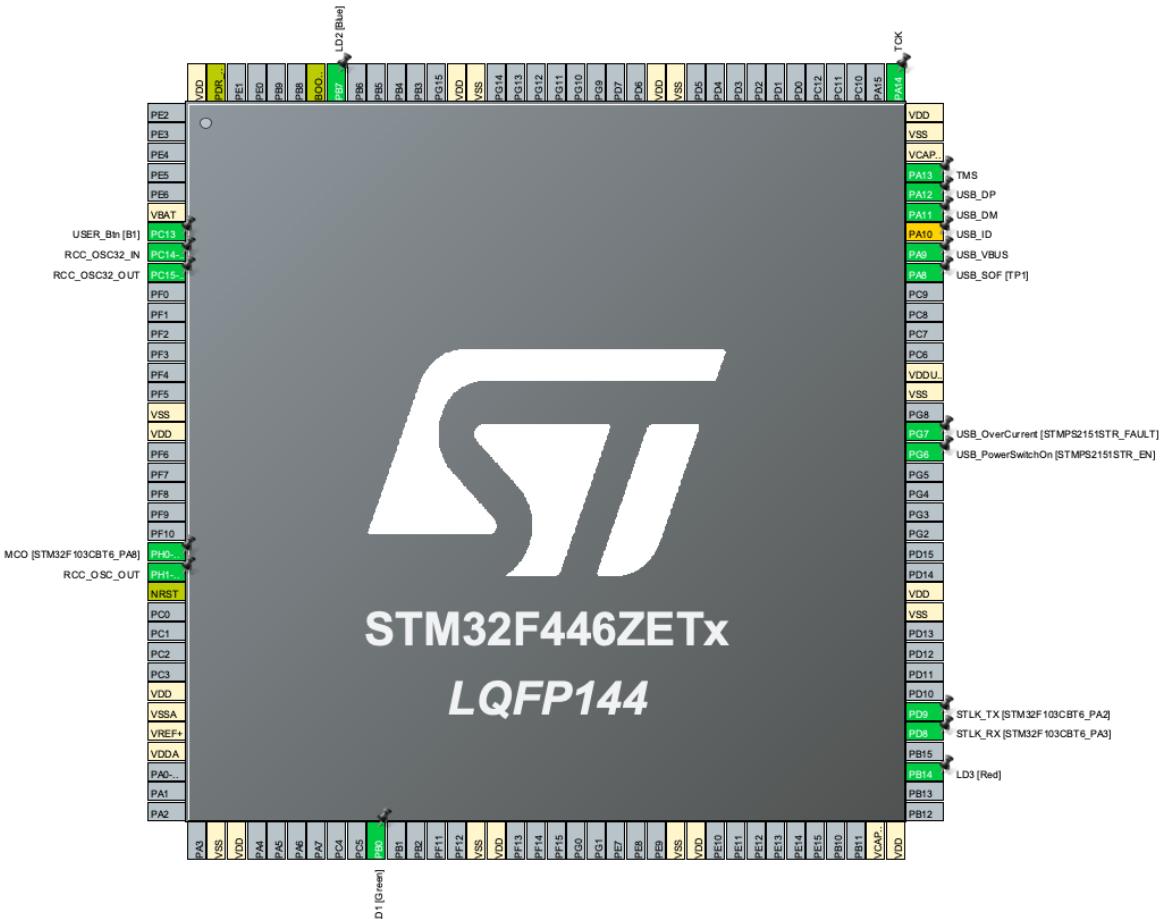
Įterptinių sistemų inžinerija

Atliko: EKSfm-24 gr. Ignas Malinauskas

Tikrino: dr. Eldar Šabanovič

VILNIUS 2025

1. 1 laboratorinio darbo mikrovaldiklio GPIO kaiščių konfigūracija



2. LED 1 mirksėjimo kodas

The screenshot shows the STM32CubeIDE interface with the main.c file open. The code initializes peripherals (MX_GPIO_Init(), MX_USART3_UART_Init(), MX_USB_OTG_FS_PCD_Init()), then enters an infinite loop. Inside the loop, it checks if a button is pressed (HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin) == GPIO_PIN_SET). If true, it toggles the LED1 pin (HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin)) and waits 1000ms (HAL_Delay(1000)). If false, it writes the pin to its reset state (HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET)). The code ends with a comment /* USER CODE END 3 */.

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART3_UART_Init();
MX_USB_OTG_FS_PCD_Init();
/* USER CODE BEGIN 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if(HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin) == GPIO_PIN_SET)
    {
        HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
        HAL_Delay(1000);
    }
    else
    {
        HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
    }
    /* USER CODE END 3 */
}
```

Below the code editor, the console output shows:

```
Time elapsed during verifying operation: 00:00:00.120
Download verified successfully
Shutting down...
Exit.
```

3. LED GPIO vertė keičiama per derinimą

Šioje vietoje išvestis pakeičiama per ODR (Output Data Register) susiradus LED1 atitinkanti bitą

The screenshot shows the STM32CubeIDE interface with the main.c file open. The error_handler() function is shown, which includes private defines for various pins and ports. The Registers window on the right shows the ODR0 register (Address [0:1], Value 0x1), which corresponds to the LED1 pin. The memory dump shows the value 0x1 at address 0x40020418, which is the address of the ODR0 register.

```
void error_handler(void);

/* USER CODE BEGIN EFP */
/* USER CODE END EFP */

/* Private defines */
#define USER_BTN_Pin GPIO_PIN_13
#define USER_BTN_GPIO_Port GPIOC
#define HCO_Pin GPIO_PIN_0
#define HCO_GPIO_Port GPIOH
#define LD1_Pin GPIO_PIN_0
#define LD1_GPIO_Port GPIOB
#define LD3_Pin GPIO_PIN_4
#define LD3_GPIO_Port GPIOB
#define STLK_RX_Pin GPIO_PIN_8
#define STLK_RX_GPIO_Port GPIOD
#define STLK_TX_Pin GPIO_PIN_9
#define STLK_TX_GPIO_Port GPIOD
#define STLK_TX_GPIO_Port GPIOD
#define USB_PowerSwitchOn_Pin GPIO_PIN_6
#define USB_PowerSwitchOn_GPIO_Port GPIOG
#define USB_DM_Pin GPIO_PIN_7
#define USB_DM_Port GPIOA
#define USB_Current_Gpio_Port GPIOG
#define USB_SOF_Pin GPIO_PIN_8
#define USB_SOF_GPIO_Port GPIOA
#define USB_VBUS_Pin GPIO_PIN_9
#define USB_VBUS_GPIO_Port GPIOA
#define USB_ID_Pin GPIO_PIN_10
#define USB_ID_Port GPIOA
#define USB_DM_Pin GPIO_PIN_11
#define USB_DM_GPIO_Port GPIOA
#define USB_DP_Pin GPIO_PIN_12
#define USB_DP_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TKE_Pin GPIO_PIN_14
#define TKE_GPIO_Port GPIOA
```

Registers window (ODR0 register values):

Register	Address	Value
ODR9	[9:1]	0x0
ODR8	[8:1]	0x0
ODR7	[7:1]	0x0
ODR6	[6:1]	0x0
ODR5	[5:1]	0x0
ODR4	[4:1]	0x0
ODR3	[3:1]	0x0
ODR2	[2:1]	0x0
ODR1	[1:1]	0x0
ODR0	[0:1]	0x1

4. 2 LED mirksėjimas nustatytu dažniu pagal wait_ms kintamąjį

```
uint16_t wait_ms = 500;
```

```
while (1)
{
    HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
    HAL_Delay(wait_ms);
}
```

5. Wait_ms kintamojo keitimas derinimo metu

The screenshot shows the STM32CubeIDE interface. On the left, the code editor displays the main.c file with the following content:

```
91 /* USER CODE BEGIN SysInit */
92
93 /* USER CODE END SysInit */
94
95 /* Initialize all configured peripherals */
96 MX_GPIO_Init();
97 MX_USART3_UART_Init();
98 MX_USB_OTG_FS_PCD_Init();
99 MX_TIM1_Init();
100 /* USER CODE BEGIN 2 */
101 HAL_TIM_Base_Start_IT(&htim1);
102 /* USER CODE END 2 */
103
104 /* Infinite loop */
105 /* USER CODE BEGIN WHILE */
106 while (1)
107 {
108     /* USER CODE END WHILE */
109
110     /* USER CODE BEGIN 3 */
111     if(HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin) == GPIO_PIN_SET)
112     {
113         HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
114         HAL_Delay(wait_ms);
115     }
116     else
117     {
118         HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
119     }
120
121 //     HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
122 //     HAL_Delay(wait_ms);
123
124 }
```

On the right, the debugger window shows a variable table with one entry:

Expression	Type	Value
wait_ms	uint16_t	500
Add new expression		

At the bottom, the console window shows the message "Download verified successfully".

STM32CubeIDE

c workspace_1.19.0 - LD1_Task1/Core/Src/main.c - STM32CubeIDE

```

88     /* Configure the system clock */
89     SystemClock_Config();
90
91     /* USER CODE BEGIN SysInit */
92
93     /* USER CODE END SysInit */
94
95     /* Initialize all configured peripherals */
96     MX_GPIO_Init();
97     MX_USART3_UART_Init();
98     MX_USB_OTG_FS_PCD_Init();
99     MX_TIM1_Init();
100    /* USER CODE BEGIN 2 */
101    HAL_TIM_Base_Start_IT(&htim1);
102    /* USER CODE END 2 */
103
104    /* Infinite loop */
105    /* USER CODE BEGIN WHILE */
106    while (1)
107    {
108        /* USER CODE END WHILE */
109
110        /* USER CODE BEGIN 3 */
111        if(HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin) == GPIO_PIN_SET)
112        {
113            HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
114            HAL_Delay(wait_ms);
115        }
116        else
117        {
118            HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
119        }
120
121    }
122    // HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
123    // HAL_Delay(wait_ms);
124}

```

Expression Type Value

wait_ms	uint16_t	500
---------	----------	-----

Name : wait_ms
 Details:1000
 Default:1000
 Decimal:1000
 Hex:0x3e8
 Binary:1111101000
~~Octal:1750~~

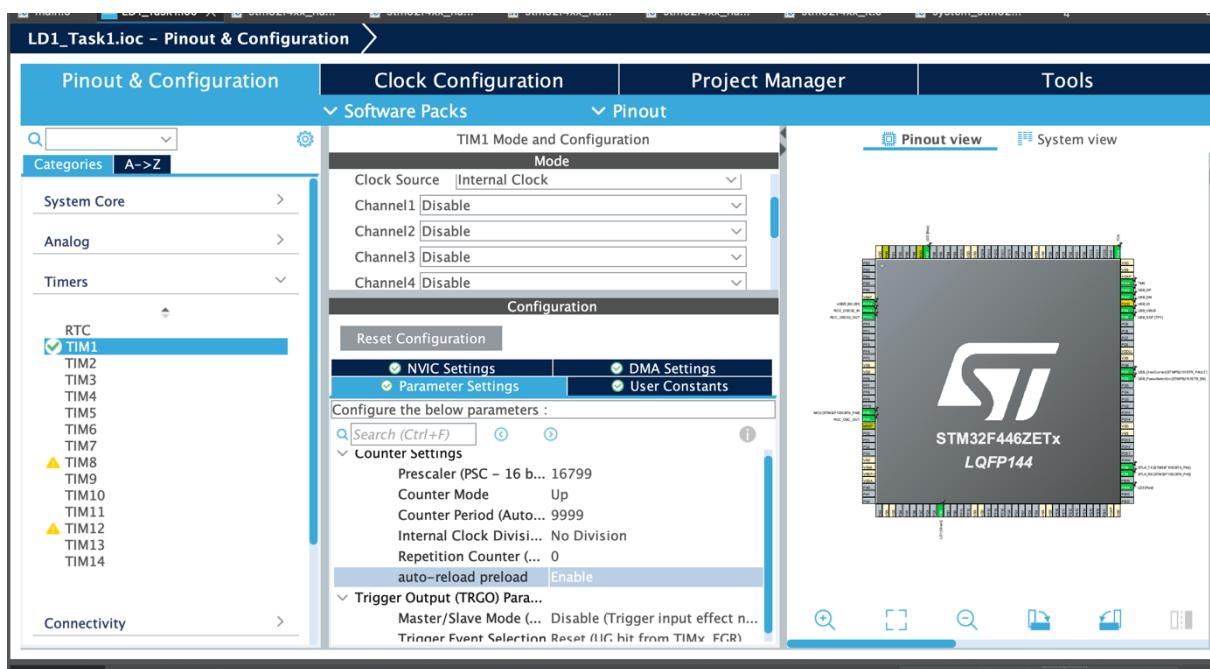
Time elapsed during verifying: 00:00:00.000000

Download verified successfully

Writable Smart Insert 115 : 30 [7]

6. Laikmačio ir pertraukties panaudojimas LED mirksėjimui

6.1. CubeMX nustatymuose sukonfigūruojamas vienos sekundės laikmatis



6.2. Pertraukties kodas, įvykdomas kai sutiksi laikmatis – aktyvuojamas handleris ir nukreipiamas į mirksėjimo funkciją

```
7. void TIM1_UP_TIM10_IRQHandler(void)
8. {
9.     /* USER CODE BEGIN TIM1_UP_TIM10_IRQHandler_0 */
10.
11.    /* USER CODE END TIM1_UP_TIM10_IRQHandler_0 */
12.    HAL_TIM_IRQHandler(&htim1);
13.    /* USER CODE BEGIN TIM1_UP_TIM10_IRQHandler_1 */
14.
15.    /* USER CODE END TIM1_UP_TIM10_IRQHandler_1 */
16. }
```

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    /* Prevent unused argument(s) compilation warning */
    if (htim->Instance == TIM1)
    {
        HAL_GPIO_TogglePin(LD3_GPIO_Port, LD3_Pin);
    }

    /* NOTE : This function should not be modified, when the
    callback is needed,
           the HAL_TIM_PeriodElapsedCallback could be
    implemented in the user file
    */
}
```

8. UART su pertrauktim panaudojimas

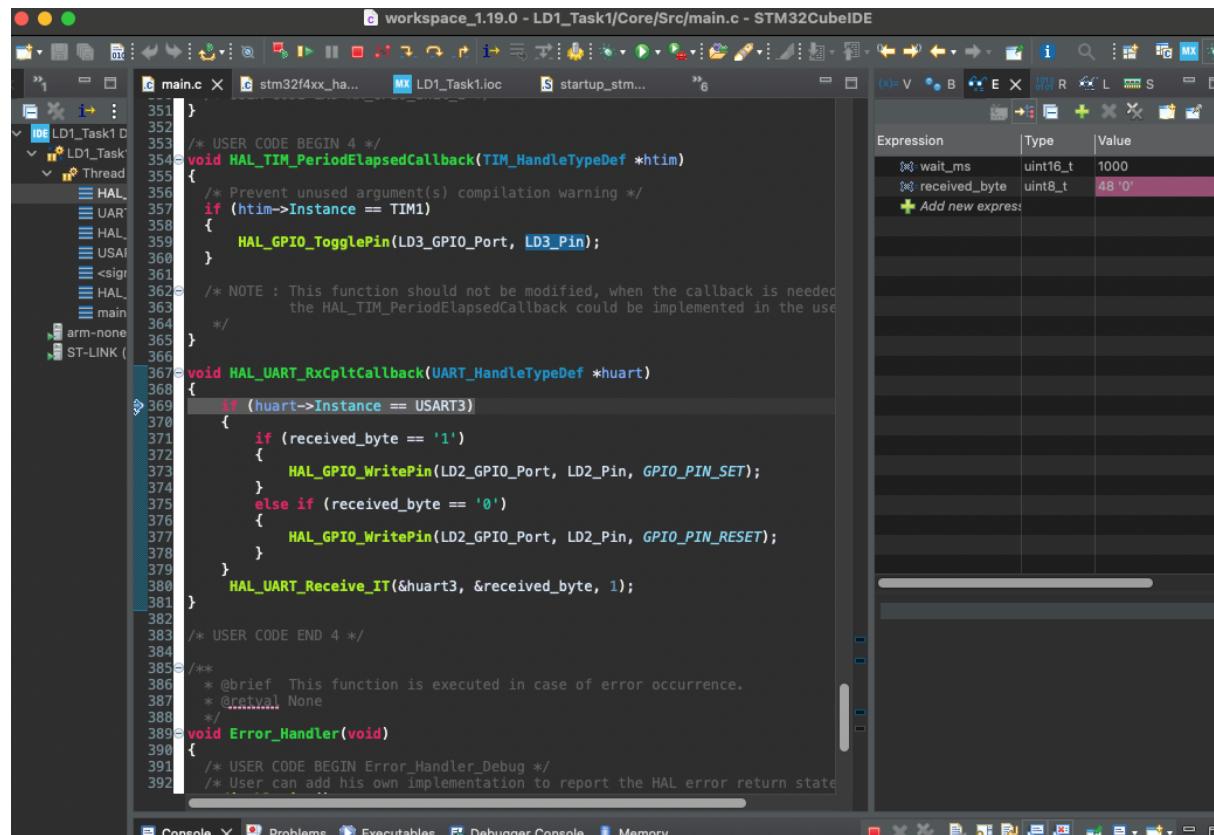
Kodo gabaliukas, kuriame naudojamas UART teksto spausdinimui į terminalą.

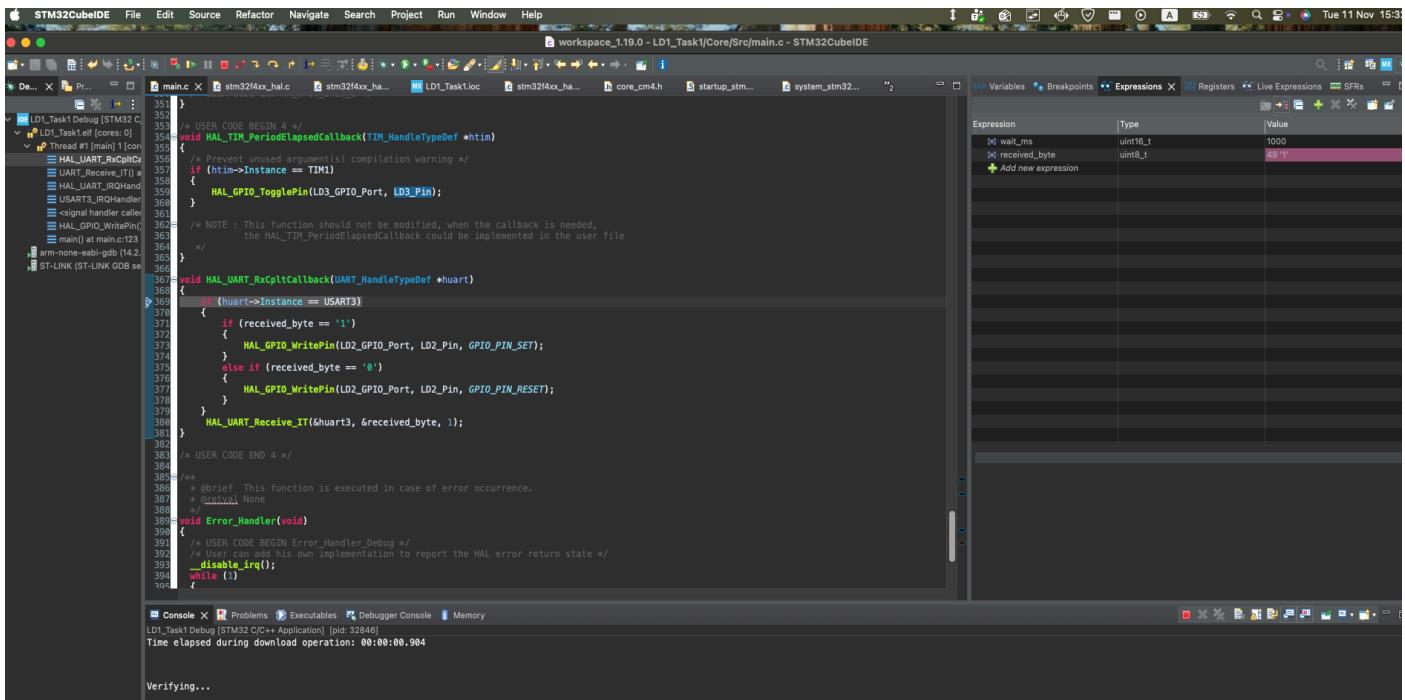
```
uint32_t last_print = 0;
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    if(HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin) == GPIO_PIN_SET)
    {
        HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
        HAL_Delay(wait_ms);
        HAL_UART_Transmit(&huart3, "Hello\r\n", 7, 1000);
    }
}
```

Kodo fragmentas, kuriame aprašyta logika – kai iš serijinio prievado gaunamos vertės 1 arba 0, vykdoma pertrauktis ir atitinkamai i Jungiama arba išjungiama LED2.





9. Print funkcijos kodas ir mikrovaldiklio bei serijinio terminalo nuotraukos

```
#define PRINT_BUFFER_SIZE 128
```

```
void print(const char *format, ...) {
    uint8_t buffer[PRINT_BUFFER_SIZE];
    va_list args;
    uint16_t len;

    va_start(args, format);

    len = vsnprintf((char*)buffer, PRINT_BUFFER_SIZE,
                    format, args);

    va_end(args);

    HAL_UART_Transmit(&huart3, buffer, len, HAL_MAX_DELAY);
}

print("-----testing print function -----");
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
uint32_t last_print = 0;
```

```
while (1)
{
    if (HAL_GetTick() - last_print >= 1000){
        last_print += 1000;
        print("MCU laikas dabar yra: %ld mms\r\n",
HAL_GetTick());
    }
}
```

