Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 5

з дисципліни: «Тестування та діагностика програмно-апаратних засобів»

на тему: «Дослідження особливостей взаємодії апаратних та програмних засобів»

Виконав:

ст. гр. КІ-303

Порубайміх О.Є.

Перевірив:

Ст.викл. Хомуляк М.О.

Львів – 2024

Зміст

[Зміст 2](#__RefHeading___Toc55759_1171611487)

[Перелік рисунків 2](#__RefHeading___Toc55761_1171611487)

[РОЗДІЛ 1. Мета 4](#__RefHeading___Toc14349_1995045277)

[РОЗДІЛ 2. Теоретичні відомості 5](#__RefHeading___Toc14355_1995045277)

[РОЗДІЛ 3. Індивідуальне завдання 7](#__RefHeading___Toc14353_1995045277)

[РОЗДІЛ 4. Виконання завдання 8](#__RefHeading___Toc14351_1995045277)

[4.1. Створення проєкту. 8](#__RefHeading___Toc14357_1995045277)

[4.2. Програмування власного варіанту. 12](#__RefHeading___Toc14369_1995045277)

[Висновок 14](#__RefHeading___Toc14347_1995045277)

[Список використаних джерел 15](#__RefHeading___Toc14345_1995045277)

[Додаток А. Код програми 16](#__RefHeading___Toc8664_2297683834)

Перелік рисунків

[Рис. 4.1. Стартове вікно STM32CubeMX. 7](#%252525D0%252525A0%252525D0%252525B8%252)

[Рис. 4.2. Вікно вибору плати. 7](#%252525D0%252525A0%252525D0%252525B8%251)

[Рис. 4.3. Короткий опис плати STM32F407G-DISC1. 8](#%252525D0%252525A0%252525D0%252525B8%253)

[Рис. 4.4. Проєкт з типовими параметрами плати. 8](#%252525D0%252525A0%252525D0%252525B8%254)

[Рис. 4.5. Відключення кварцу 32 кГц. 9](#%252525D0%252525A0%252525D0%252525B8%255)

[Рис. 4.6. Основні параметри проєкту. 9](#%252525D0%252525A0%252525D0%252525B8%256)

[Рис. 4.7. Запит на відкриття проєкту в MDK-ARM. 10](#%252525D0%252525A0%252525D0%252525B8%257)

[Рис. 4.8. Вікно MDK-ARM (IDE Keil μVision). 10](#%252525D0%252525A0%252525D0%252525B8%258)

[Рис. 4.9. Головний файл проєкту. 11](#%252525D0%252525A0%252525D0%252525B8%259)

[Рис. 4.10. Успішна компіляція. 11](#%252525D0%252525A0%252525D0%252525B8%25a)

[Рис. 4.11. спішна компіляція з оновленим кодом програми. 12](#%252525D0%252525A0%252525D0%252525B8%25b)

1. Мета

Ознайомитися з порядком розгортання та засобами програмного конфігурування і перевірки функціональних вузлів дослідницько-налагоджувальної плати на базі мікроконтролера STM32F407VGT6.

1. Теоретичні відомості

Плата розробника STM32F407G-DISC1 заснована на високопродуктивному 32-бітовому мікроконтролері STM32F407VGT6 з ядром Cortex-M4F. Вона призначена для швидкої та простої розробки програм для різних застосунків, включаючи:

* **Аудіо**: Плата має вбудований цифровий мікрофон, ЦАП, підсилювач потужності та роз'єм для підключення SD-карти, що робить її ідеальною для розробки аудіосистем.
* **Вбудовані системи**: STM32F407G-DISC1 має широкий спектр периферійних пристроїв, включаючи акселерометр, порти UART, SPI, I2C, CAN та Ethernet, що робить її універсальною платформою для розробки вбудованих систем.
* **Прототипування**: Плата має зручні роз'єми та виводи, що робить її простою у використанні для прототипування нових ідей.

**Основні характеристики.**

Мікроконтролер: STM32F407VGT6 з ядром Cortex-M4F.

Тактова частота: до 168 Мгц.

Пам'ять:

* Flash: 1 МБ
* SRAM: 192 КБ

Інтерфейси:

* 3 x SPI
* 3 x I2C
* 6 x UART
* 2 x CAN
* 1 x SDIO
* USB OTG

Периферійні пристрої:

* Вбудований цифровий мікрофон MP45DT02
* Зовнішній ЦАП CS43L22 з підсилювачем потужності класу D
* Трьохосьовий акселерометр LIS302DL
* 8 світлодіодів
* 2 кнопки

Живлення:

* Регулятор напруги 3,3 В
* Живлення через USB або зовнішній адаптер

Габарити: 97 x 66 мм

**Програмне забезпечення**

STM32F407G-DISC1 сумісна з різними середовищами розробки, включаючи:

* RealView Development Suite (ARM C / C++ Compiler);
* Keil MDK-ARM (ARM C / C++ Compiler);
* IAR Embedded Workbench for ARM (IAR C / C++ Compiler);
* MULTI IDE for ARM (Green Hills C / C++ Compiler);
* TASKING VX-toolset for ARM (Altium C / C++ Compiler);
* Sourcery CodeBench (GCC Compiler);
* Rowley CrossWorks for ARM (GCC Compiler);
* Atollic TrueSTUDIO (GCC Compiler);
* RAISONANCE Ride7 IDE for ARM (GCC Compiler);
* CooCox CoIDE (GCC Compiler);
* MikroC for ARM (MikroElektronika C Compiler).

1. Індивідуальне завдання
2. Створити проєкт за допомогою STM32CubeMX і перевірити його правильність компілюванням в Keil MDK-ARM.
3. Перевірити можливість програмування плати STM32F407G-DISC1.
4. Послідовно перевірити роботу світлодіодів користувача / розробника LD3 – LD6.
5. Модифікувати програму відповідно до індивідуального завдання.

Варіант завдання:

* Червоний, помаранчевий світлодіоди;
* Інтервал блимання кожного світлодіода: 0,2 сек.

1. Виконання завдання
   1. Створення проєкту.

Запустив інтегроване середовище розробки STM32CubeMX.

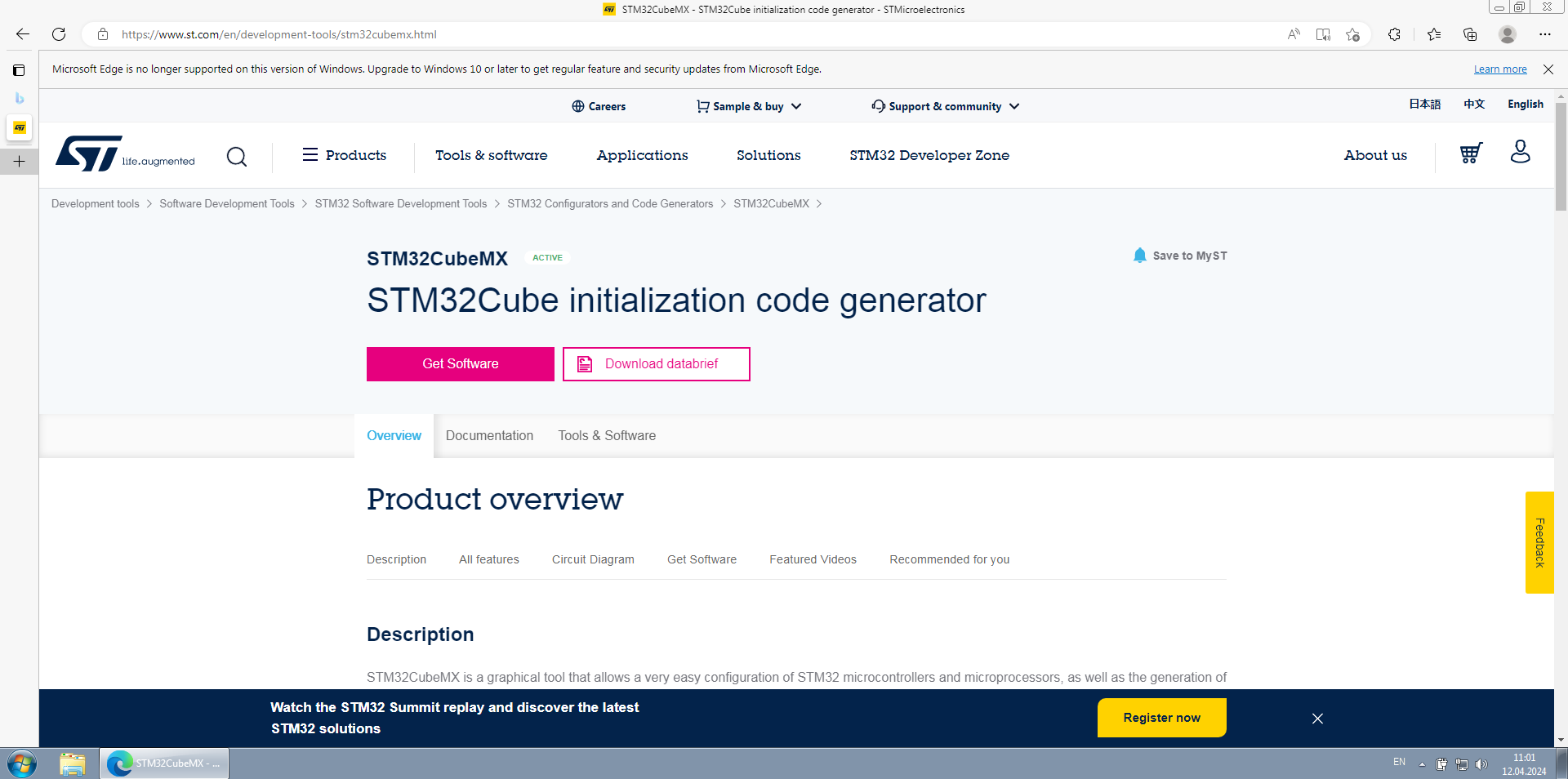


Рис. 4.1. Стартове вікно STM32CubeMX.

В меню «New Project» натиснув кнопку «ACCESS TO BOARD SELECTOR». Дочекався завантаження даних і відкриття вікна вибору плати.

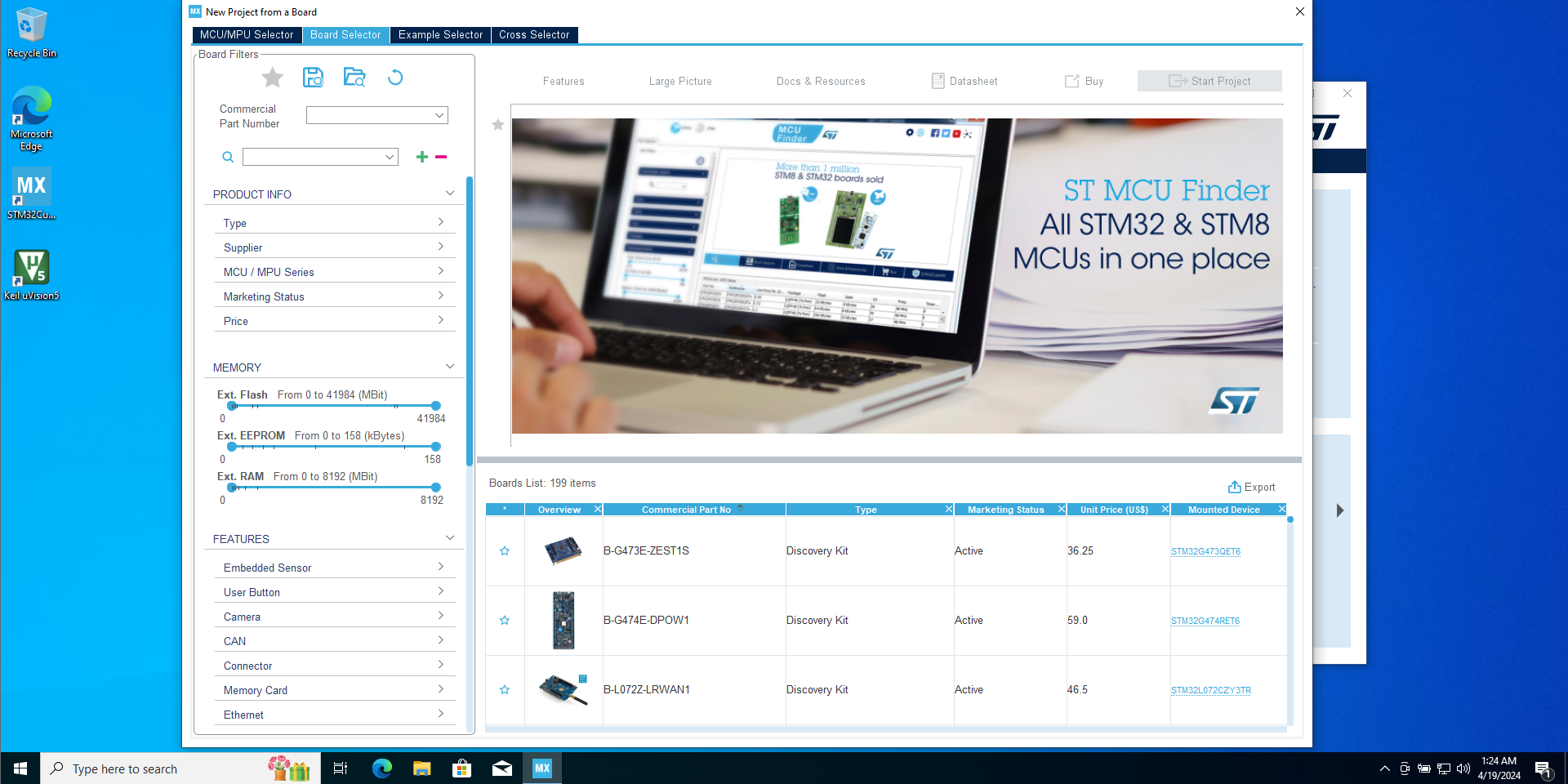


Рис. 4.2. Вікно вибору плати.

У пошуковому полі ввів STM32F407G-DISC1 і обрав необхідну плату.

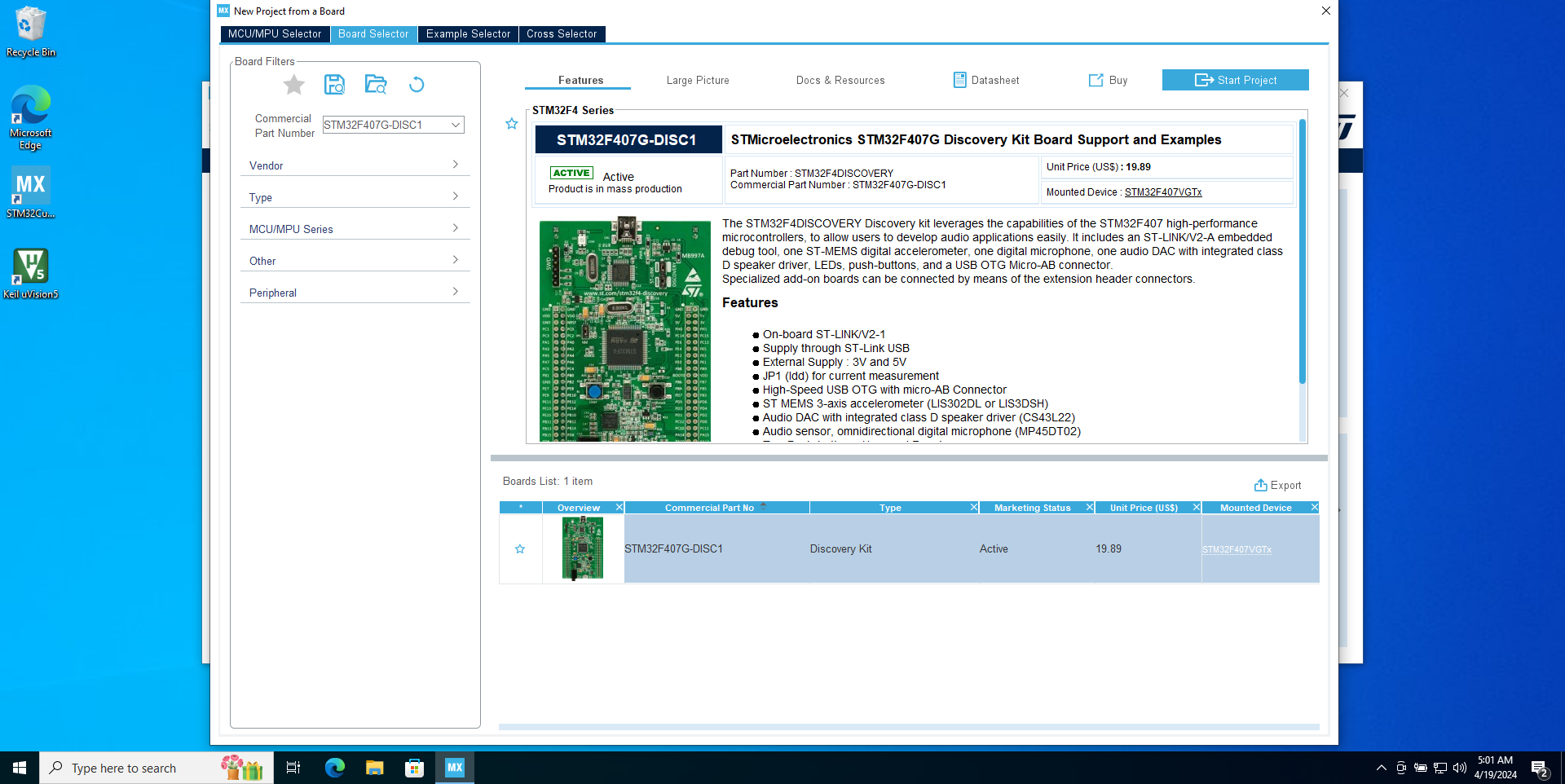


Рис. 4.3. Короткий опис плати STM32F407G-DISC1.

Дочекався створення і відкриття вікна проєкту.

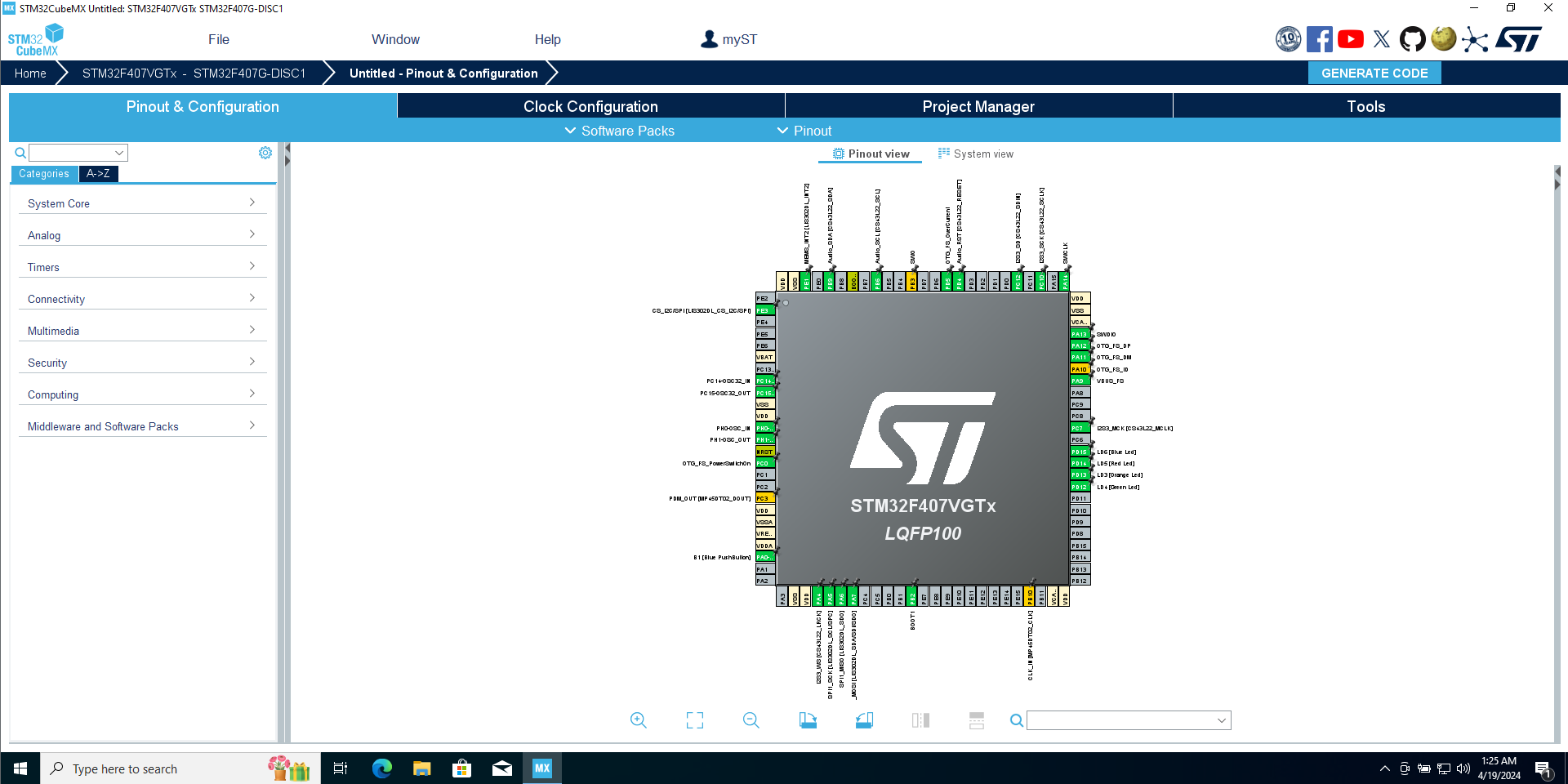


Рис. 4.4. Проєкт з типовими параметрами плати.

Розкрив меню «System Core» і натиснув «RCC». Встановив значення «Disable» для пункту «Low Speed Clock (LSP)».

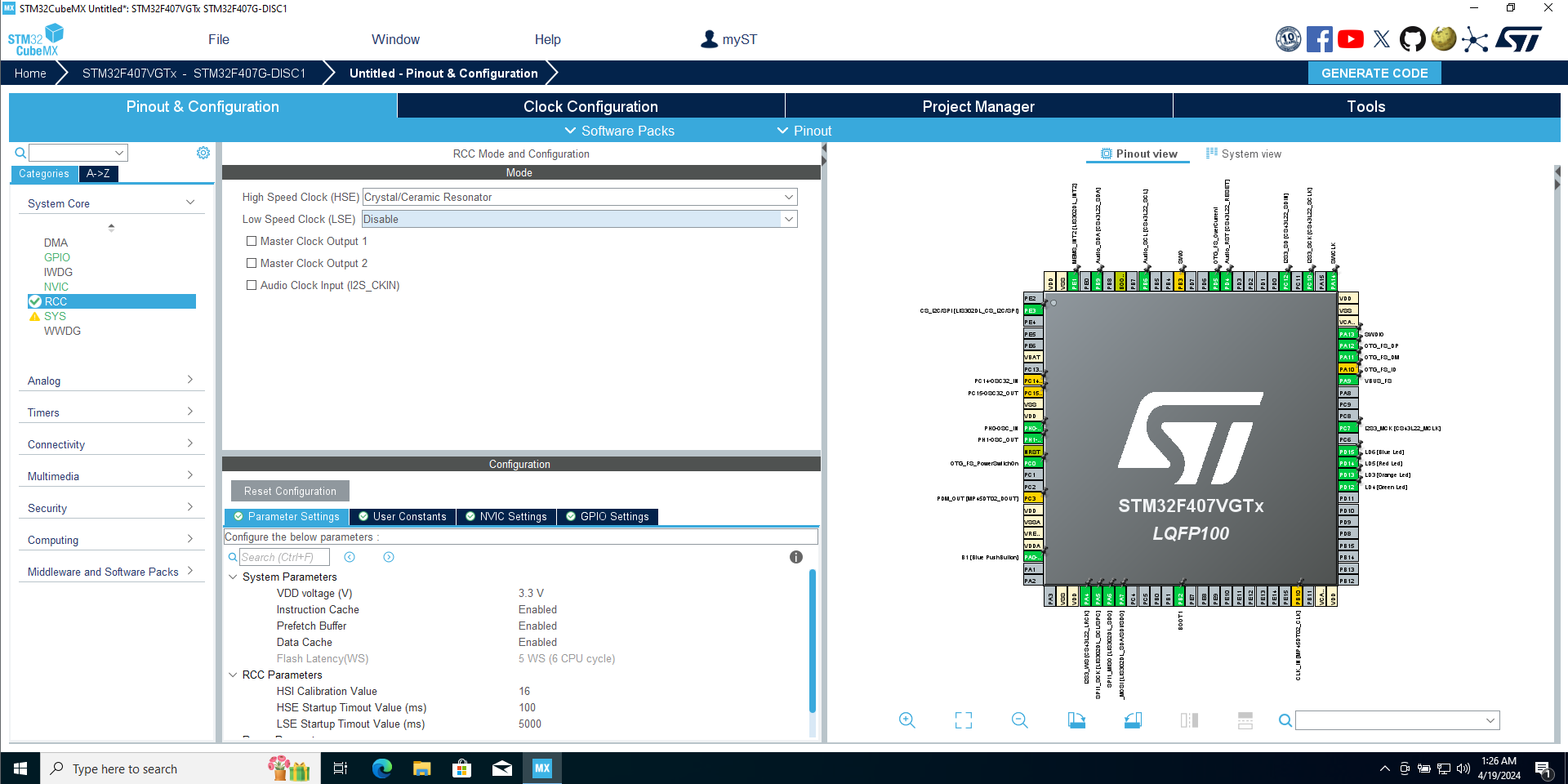


Рис. 4.5. Відключення кварцу 32 кГц.

Перейшов до вкладки «Project Manager». Визначив ім’я проєкту, його шлях, обрав MDK-ARM як інтегровану середу розробки.

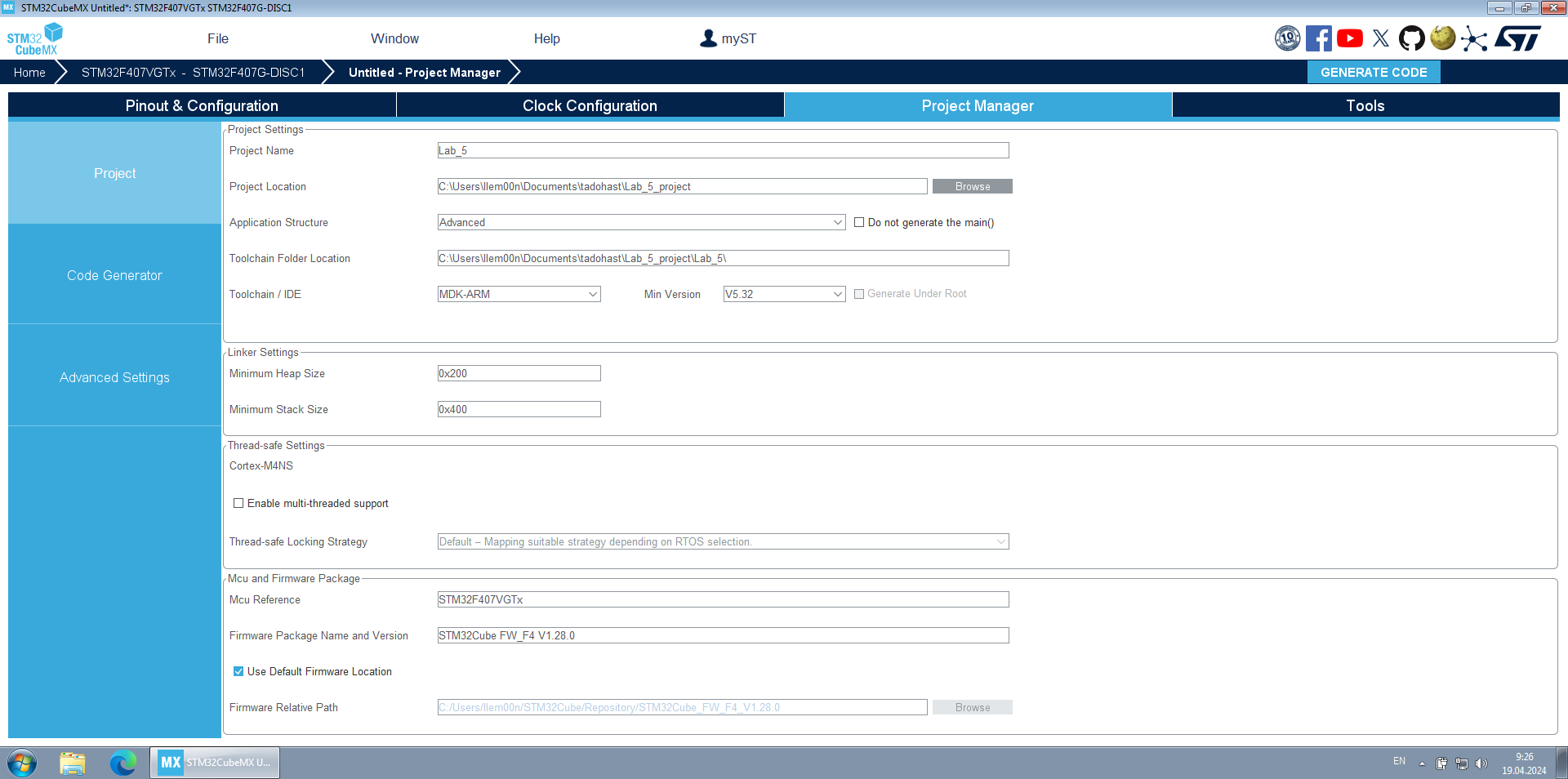


Рис. 4.6. Основні параметри проєкту.

Натиснув кнопку «GENERATE CODE». Дочекався створення проєкту.

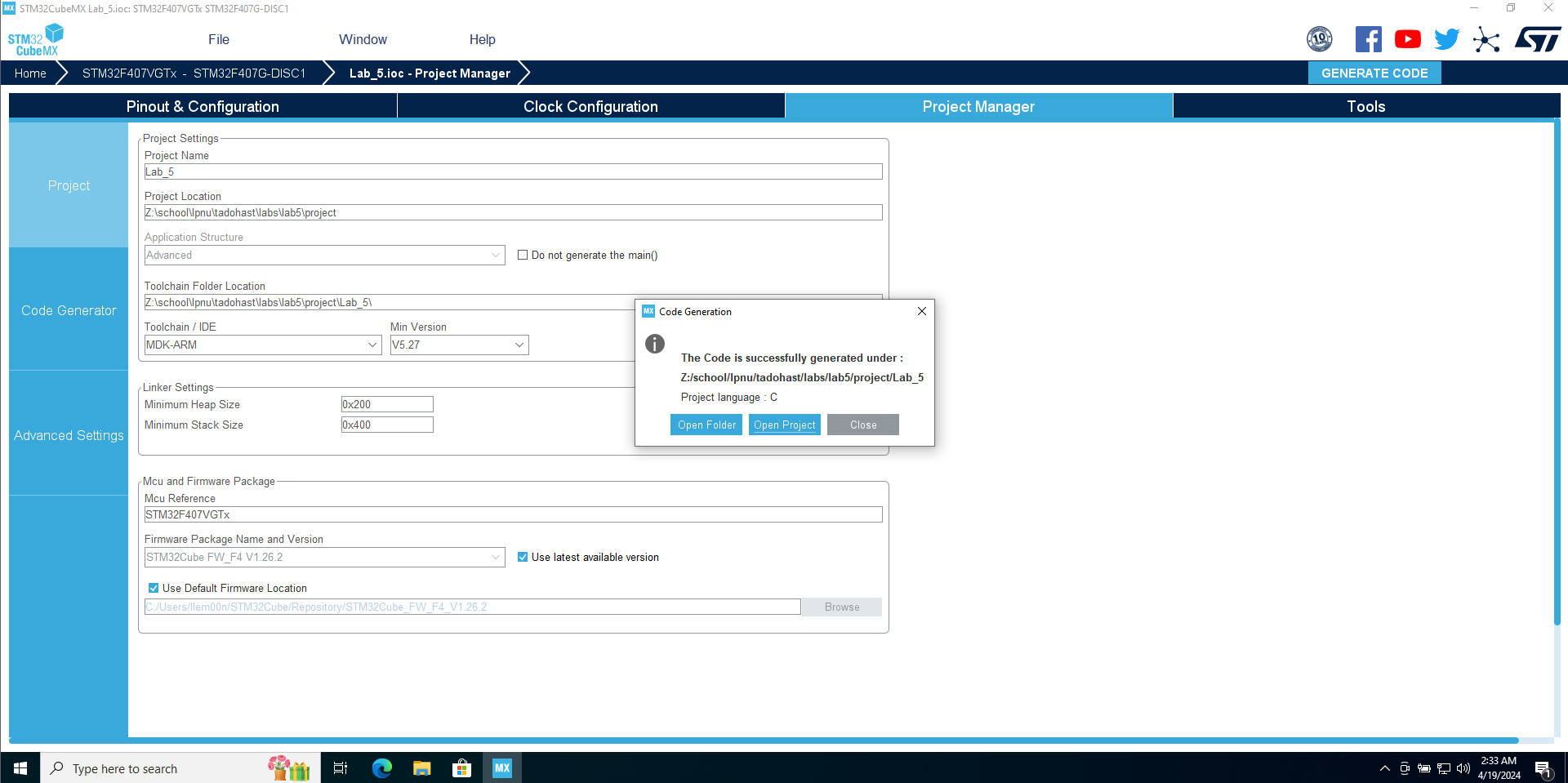


Рис. 4.7. Запит на відкриття проєкту в MDK-ARM.

Натиснув кнопку «Open Project». Дочекався відкриття вікна Keil μVision.

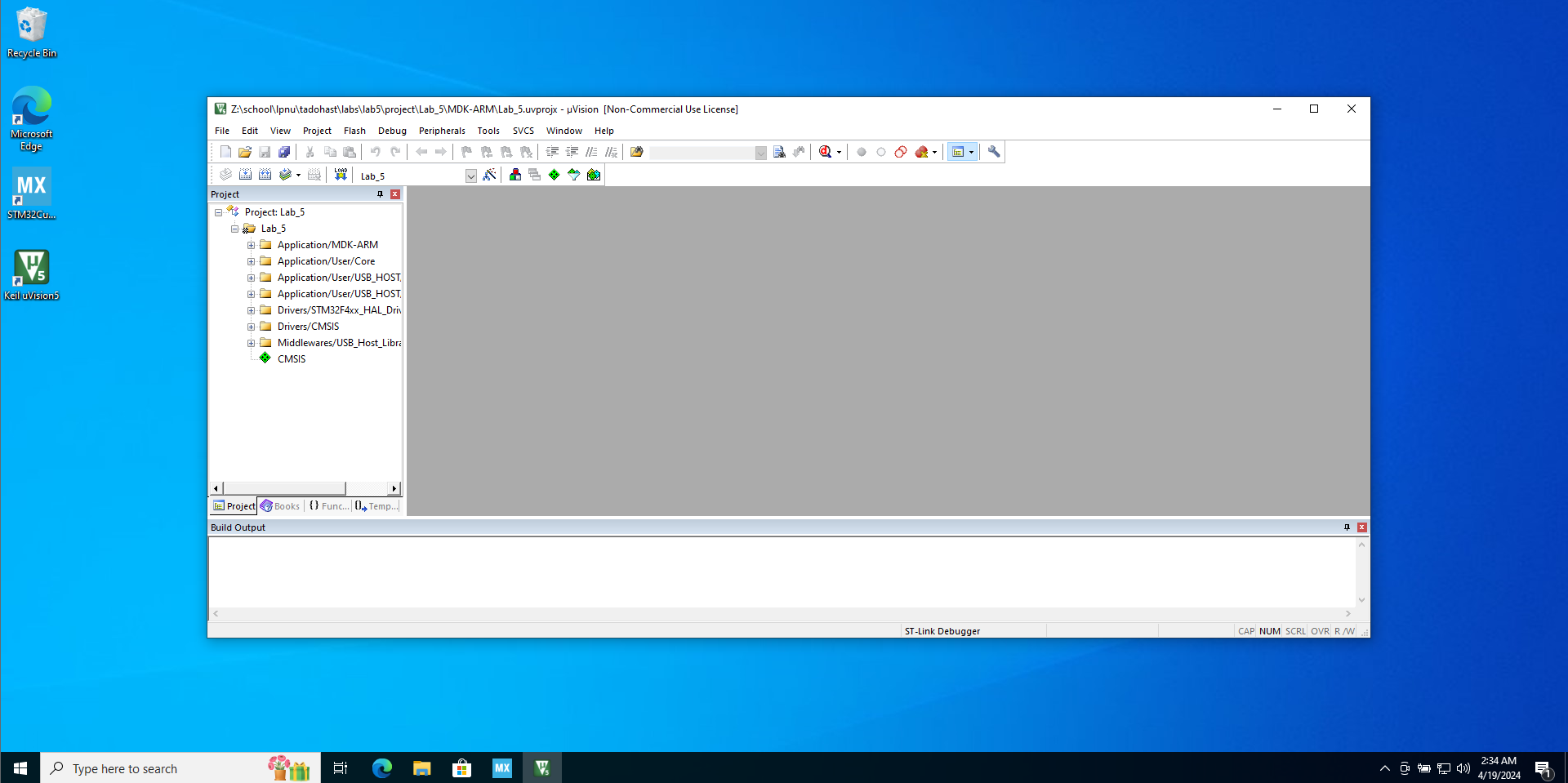


Рис. 4.8. Вікно MDK-ARM (IDE Keil μVision).

Відкрив головний файл проєкту «main.c».

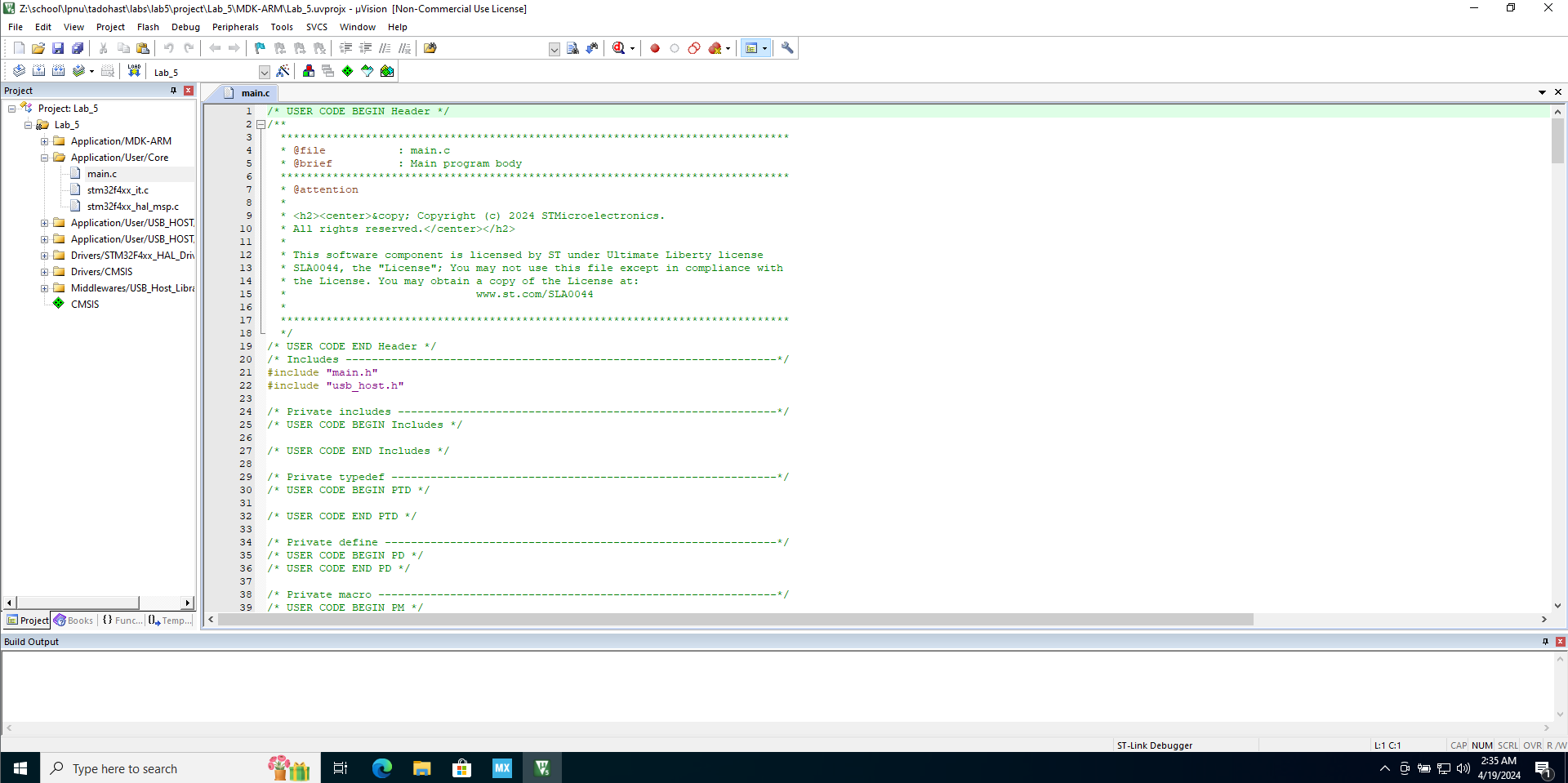


Рис. 4.9. Головний файл проєкту.

Натиснув F7, дочекався компіляції проєкту.

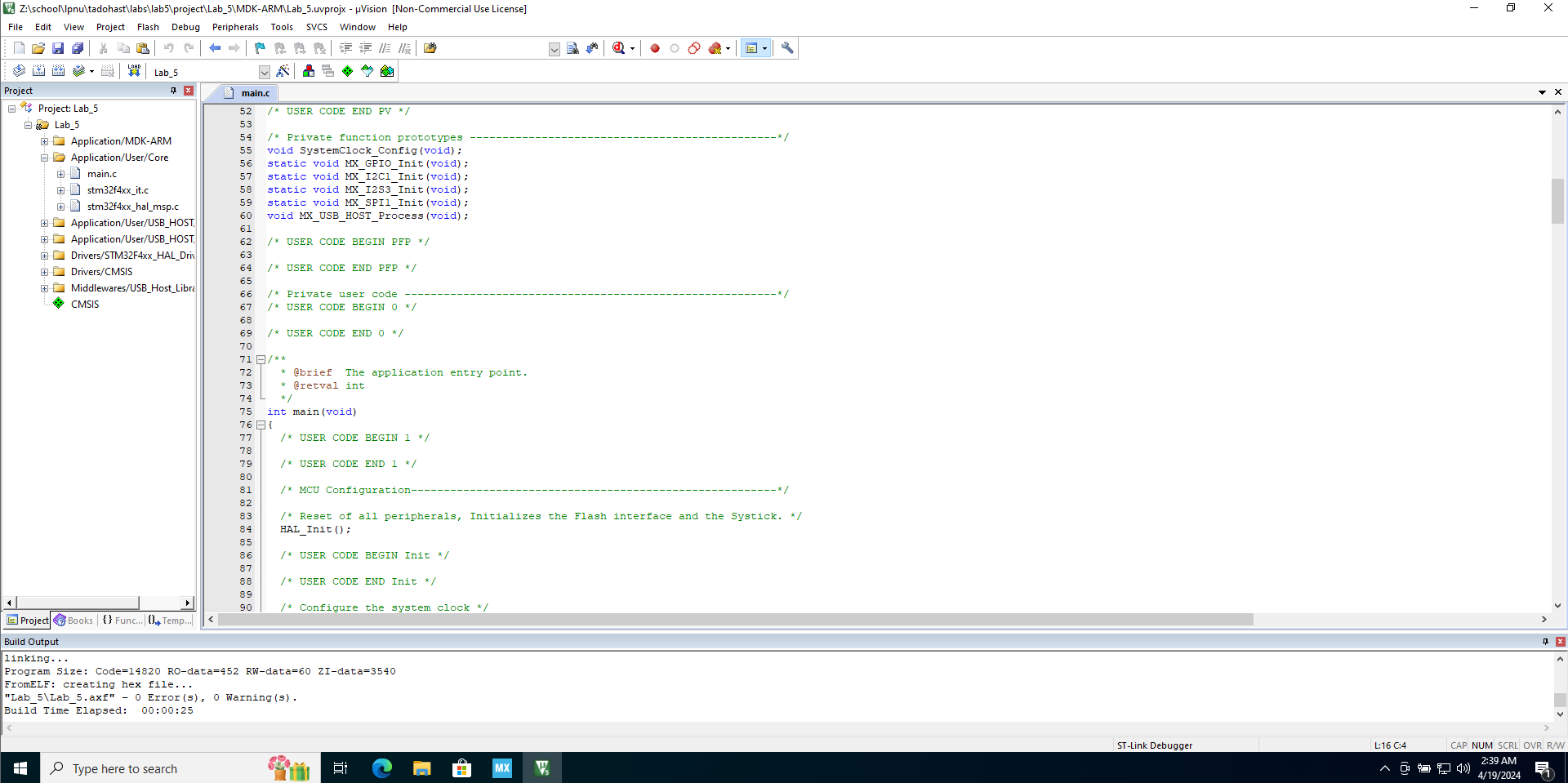


Рис. 4.10. Успішна компіляція.

* 1. Програмування власного варіанту.

У файлі «main.c» після коментаря «/\* USER CODE BEGIN 3\*/» вставив код, що відповідає виконання мого індивідуального завдання, а саме: блимання червоного та помаранчевого світлодіодів з інтервалом у 0.2 секунд.

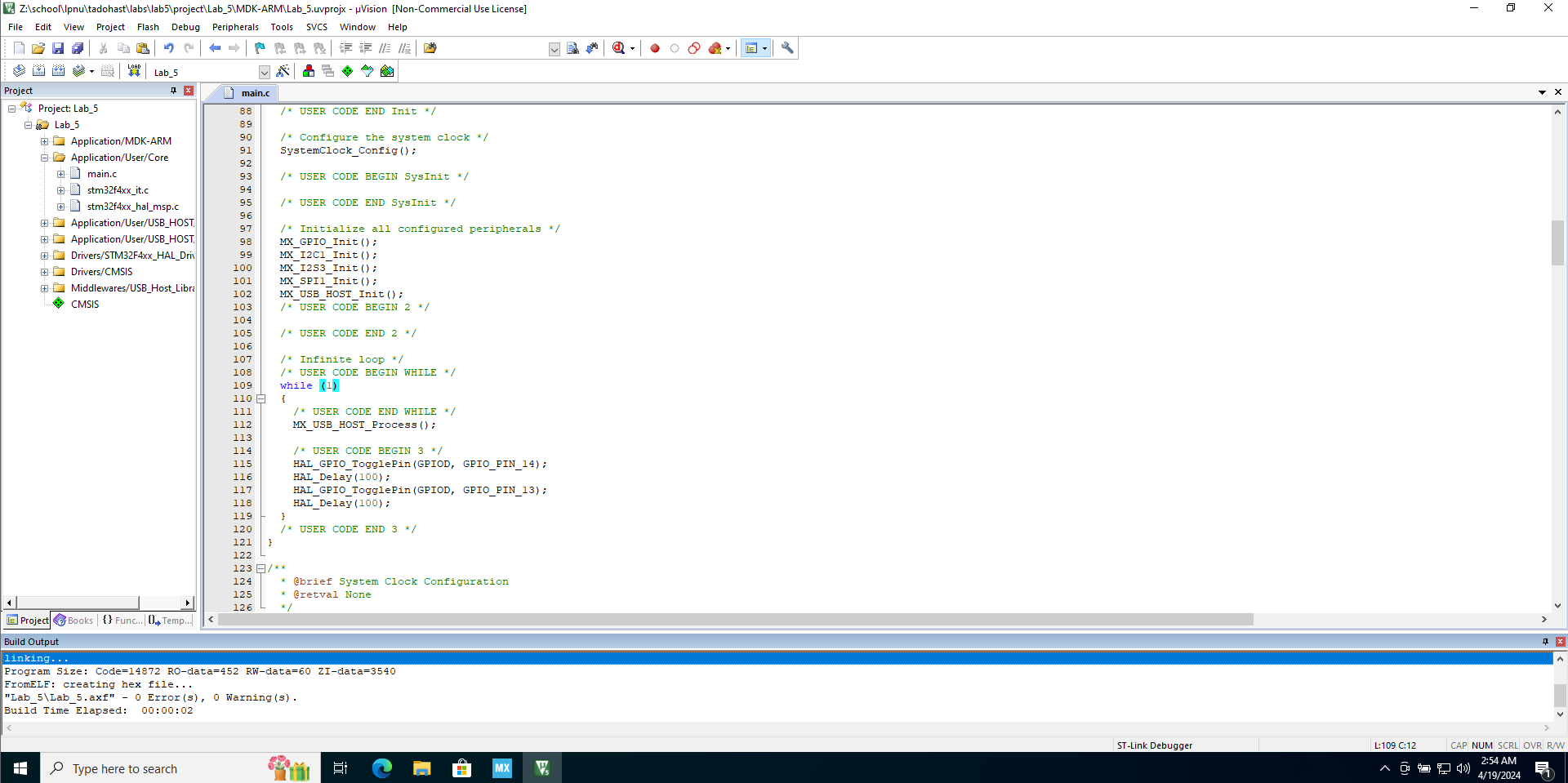


Рис. 4.11. спішна компіляція з оновленим кодом програми.

Висновок

Я ознайомився з порядком розгортання та засобами програмного конфігурування і перевірки функціональних вузлів дослідницько-налагоджувальної плати на базі мікроконтролера STM32F407VGT6.

Список використаних джерел

1. UM1472. User manual. Discovery kit with STM32F407VG MCU [Електронний ресурс] – Режим доступу: https://www.st.com/content/ccc/resource/technical/document/user\_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf.
2. STM32F405xx. STM32F407xx. Datasheet – production data [Електронний ресурс] – Режим доступу: https://www.st.com/resource/en/datasheet/stm32f407vg.pdf.
3. Налагоджувальний комплекс DISCOVERY STM32F407G-DISC1 [Електронний ресурс] – Режим доступу: http://www.kosmodrom.com.ua/el.php?name=STM32F407G-DISC1.
4. Налагоджувальна плата STM32F4 Discovery (STM32F407G-DISC1) [Електронний ресурс] – Режим доступу: <https://xcraft.com.ua/stm32f4-discovery-stm32f407g-disc1>.

Додаток А. Код програми

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2024 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under Ultimate Liberty license

\* SLA0044, the "License"; You may not use this file except in compliance with

\* the License. You may obtain a copy of the License at:

\* www.st.com/SLA0044

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "usb\_host.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 ---------------------------------------------------------\*/

I2C\_HandleTypeDef hi2c1;

I2S\_HandleTypeDef hi2s3;

SPI\_HandleTypeDef hspi1;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_I2C1\_Init(void);

static void MX\_I2S3\_Init(void);

static void MX\_SPI1\_Init(void);

void MX\_USB\_HOST\_Process(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* 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\_I2C1\_Init();

MX\_I2S3\_Init();

MX\_SPI1\_Init();

MX\_USB\_HOST\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

MX\_USB\_HOST\_Process();

/\* USER CODE BEGIN 3 \*/

HAL\_GPIO\_TogglePin(GPIOD, GPIO\_PIN\_14);

HAL\_Delay(100);

HAL\_GPIO\_TogglePin(GPIOD, GPIO\_PIN\_13);

HAL\_Delay(100);

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

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

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 336;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

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\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK)

{

Error\_Handler();

}

PeriphClkInitStruct.PeriphClockSelection = RCC\_PERIPHCLK\_I2S;

PeriphClkInitStruct.PLLI2S.PLLI2SN = 192;

PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInitStruct) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief I2C1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2C1\_Init(void)

{

/\* USER CODE BEGIN I2C1\_Init 0 \*/

/\* USER CODE END I2C1\_Init 0 \*/

/\* USER CODE BEGIN I2C1\_Init 1 \*/

/\* USER CODE END I2C1\_Init 1 \*/

hi2c1.Instance = I2C1;

hi2c1.Init.ClockSpeed = 100000;

hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c1.Init.OwnAddress1 = 0;

hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c1.Init.OwnAddress2 = 0;

hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C1\_Init 2 \*/

/\* USER CODE END I2C1\_Init 2 \*/

}

/\*\*

\* @brief I2S3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2S3\_Init(void)

{

/\* USER CODE BEGIN I2S3\_Init 0 \*/

/\* USER CODE END I2S3\_Init 0 \*/

/\* USER CODE BEGIN I2S3\_Init 1 \*/

/\* USER CODE END I2S3\_Init 1 \*/

hi2s3.Instance = SPI3;

hi2s3.Init.Mode = I2S\_MODE\_MASTER\_TX;

hi2s3.Init.Standard = I2S\_STANDARD\_PHILIPS;

hi2s3.Init.DataFormat = I2S\_DATAFORMAT\_16B;

hi2s3.Init.MCLKOutput = I2S\_MCLKOUTPUT\_ENABLE;

hi2s3.Init.AudioFreq = I2S\_AUDIOFREQ\_96K;

hi2s3.Init.CPOL = I2S\_CPOL\_LOW;

hi2s3.Init.ClockSource = I2S\_CLOCK\_PLL;

hi2s3.Init.FullDuplexMode = I2S\_FULLDUPLEXMODE\_DISABLE;

if (HAL\_I2S\_Init(&hi2s3) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2S3\_Init 2 \*/

/\* USER CODE END I2S3\_Init 2 \*/

}

/\*\*

\* @brief SPI1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_SPI1\_Init(void)

{

/\* USER CODE BEGIN SPI1\_Init 0 \*/

/\* USER CODE END SPI1\_Init 0 \*/

/\* USER CODE BEGIN SPI1\_Init 1 \*/

/\* USER CODE END SPI1\_Init 1 \*/

/\* SPI1 parameter configuration\*/

hspi1.Instance = SPI1;

hspi1.Init.Mode = SPI\_MODE\_MASTER;

hspi1.Init.Direction = SPI\_DIRECTION\_2LINES;

hspi1.Init.DataSize = SPI\_DATASIZE\_8BIT;

hspi1.Init.CLKPolarity = SPI\_POLARITY\_LOW;

hspi1.Init.CLKPhase = SPI\_PHASE\_1EDGE;

hspi1.Init.NSS = SPI\_NSS\_SOFT;

hspi1.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_2;

hspi1.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

hspi1.Init.TIMode = SPI\_TIMODE\_DISABLE;

hspi1.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;

hspi1.Init.CRCPolynomial = 10;

if (HAL\_SPI\_Init(&hspi1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN SPI1\_Init 2 \*/

/\* USER CODE END SPI1\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(CS\_I2C\_SPI\_GPIO\_Port, CS\_I2C\_SPI\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(OTG\_FS\_PowerSwitchOn\_GPIO\_Port, OTG\_FS\_PowerSwitchOn\_Pin, GPIO\_PIN\_SET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, LD4\_Pin|LD3\_Pin|LD5\_Pin|LD6\_Pin

|Audio\_RST\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : CS\_I2C\_SPI\_Pin \*/

GPIO\_InitStruct.Pin = CS\_I2C\_SPI\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(CS\_I2C\_SPI\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_PowerSwitchOn\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_PowerSwitchOn\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(OTG\_FS\_PowerSwitchOn\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : PDM\_OUT\_Pin \*/

GPIO\_InitStruct.Pin = PDM\_OUT\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(PDM\_OUT\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : BOOT1\_Pin \*/

GPIO\_InitStruct.Pin = BOOT1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(BOOT1\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : CLK\_IN\_Pin \*/

GPIO\_InitStruct.Pin = CLK\_IN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(CLK\_IN\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : LD4\_Pin LD3\_Pin LD5\_Pin LD6\_Pin

Audio\_RST\_Pin \*/

GPIO\_InitStruct.Pin = LD4\_Pin|LD3\_Pin|LD5\_Pin|LD6\_Pin

|Audio\_RST\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pin : OTG\_FS\_OverCurrent\_Pin \*/

GPIO\_InitStruct.Pin = OTG\_FS\_OverCurrent\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(OTG\_FS\_OverCurrent\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : MEMS\_INT2\_Pin \*/

GPIO\_InitStruct.Pin = MEMS\_INT2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_EVT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(MEMS\_INT2\_GPIO\_Port, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @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 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/