Національний технічний університет України

“ Київський політехнічний інститут ”



Лабораторна робота №1

По курсу: ” МІКРОПРОЦЕСОРНІ СИСТЕМИ НА БАЗІ ARM ПРОЦЕСОРІВ*”*

Тема: “ Ознайомлення з середовищем розробки STM32CubeIDE. Ініціалізація, налаштування та використання портів та таймерів ”

Варіант 2

Виконав:Студент І курсу

Групи ДС-31мп

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Київ 2024 р.**

***Завдання***

Порти введення-виведення:

1. Ініціалізувати вивід порту PA, за номером варіанту = PA2,на вихід, та вивід РВ0 на вхід у режимі підтяжки “No pull-up, no pull-down” (floating input).

2. Створити програмний код, що реалізує нескінченне опитування виводу РВ0 на предмет присутності вхідного сигналу. За присутності вхідного сигналу, відбувається підключення периферії підключеної до ініціалізованого виводу порту РА. Врахувати необхідність усунення явища «тремтіння контактів» на програмному рівні.

3. Ініціалізувати вивід порту PA за подвоєним номером варіанту =PA4 на вихід, та вивід порту РВ за номером варіанту =PB2 на вхід у режимі генерації зовнішнього переривання “EXTI”, підтяжка в режимі “Pull-down”.

4. Створити програмний код, що реалізує підключення периферії підключеної до ініціалізованого виводу порту РА за виникнення зовнішнього переривання від ініціалізованого вхідного виводу порту РВ.

5. Створити модель у середовищі “Proteus”, що демонструє роботу створеного коду. Врахувати необхідність усунення явища «тремтіння контактів» на апаратному рівні.

Таймери:

1. Налаштувати таймер ТІМ 1. Часовий інтервал розраховується за формулою: N\*2000+1000\*mod(N,2) мс, де N – номер варіанту. Підключити роботу з перериваннями. 2\*2000+1000\*mod(2,2)=4000 мс.

2. Ініціалізувати вивід порту PA за подвоєним номером варіанту =PA4 на вихід, вивід порту PB0 – вхід.

3. Створити програмний код, що реалізує запуск таймеру за появи вхідного сигналу на виводі РВ0. Таймер, за досягнення заданого часового інтервалу ініціалізує запуск периферії підключеної до ініціалізованого виводу порту РА. Врахувати необхідність усунення явища «тремтіння контактів».

4. Створити програмний код, що реалізує зміну часового інтервалу вимірювання таймера через певний часовий інтервал після запуску таймеру відповідно до п.3 на половину від величини інтервалу визначеної в п.3.

5. Створити модель у середовищі “Proteus”, що демонструє роботу створеного коду.

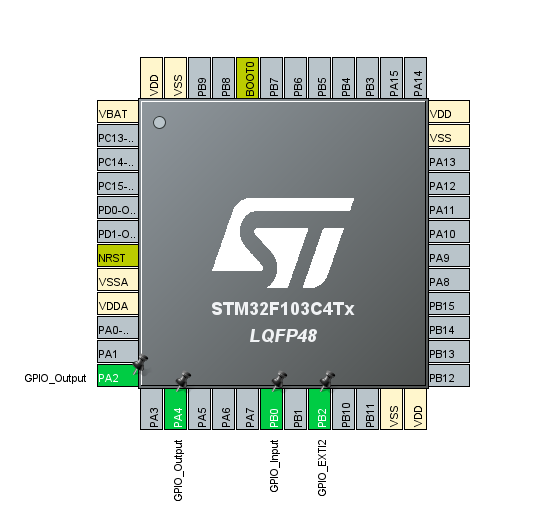


Рис. 1 Конфігурація пінів

*Лістинг програми*

Порти введення-виведення:

main.c

**int** **main**(**void**)

{

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

**if**(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0) == *GPIO\_PIN\_SET*) // Якщо кнопка натиснута

{

HAL\_Delay(250); // встановлюємо затримку

**if**(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0) == *GPIO\_PIN\_SET*) // Перевірка після затримки

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_2, *GPIO\_PIN\_SET*); // Активуємо PA2

}

}

**else**

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_2, *GPIO\_PIN\_RESET*); // Деактивуємо PA2, якщо сигнал відсутній

}

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

…

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_2|GPIO\_PIN\_4, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pins : PA2 PA4 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_2|GPIO\_PIN\_4;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

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

/\*Configure GPIO pin : PB0 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

/\*Configure GPIO pin : PB2 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_2;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(*EXTI2\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*EXTI2\_IRQn*);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

stm32f1xx\_it.c

**void** **EXTI2\_IRQHandler**(**void**)

{

/\* USER CODE BEGIN EXTI2\_IRQn 0 \*/

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4);

/\* USER CODE END EXTI2\_IRQn 0 \*/

HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_2);

/\* USER CODE BEGIN EXTI2\_IRQn 1 \*/

/\* USER CODE END EXTI2\_IRQn 1 \*/

}

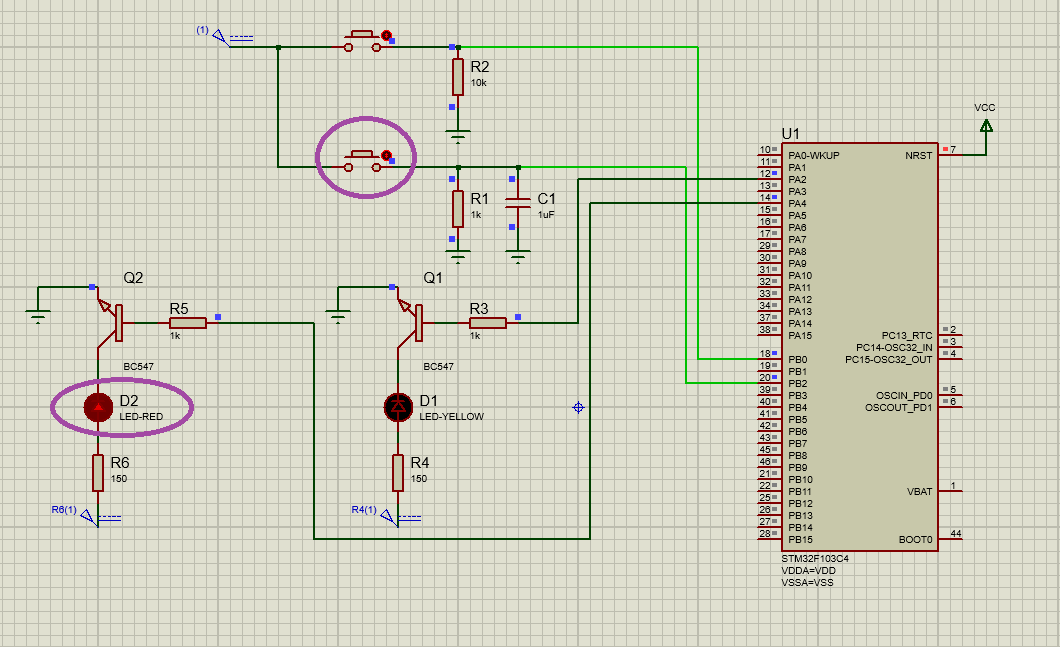


Рис. 2 Після натиску на кнопку 2, маємо зовнішнє переривання

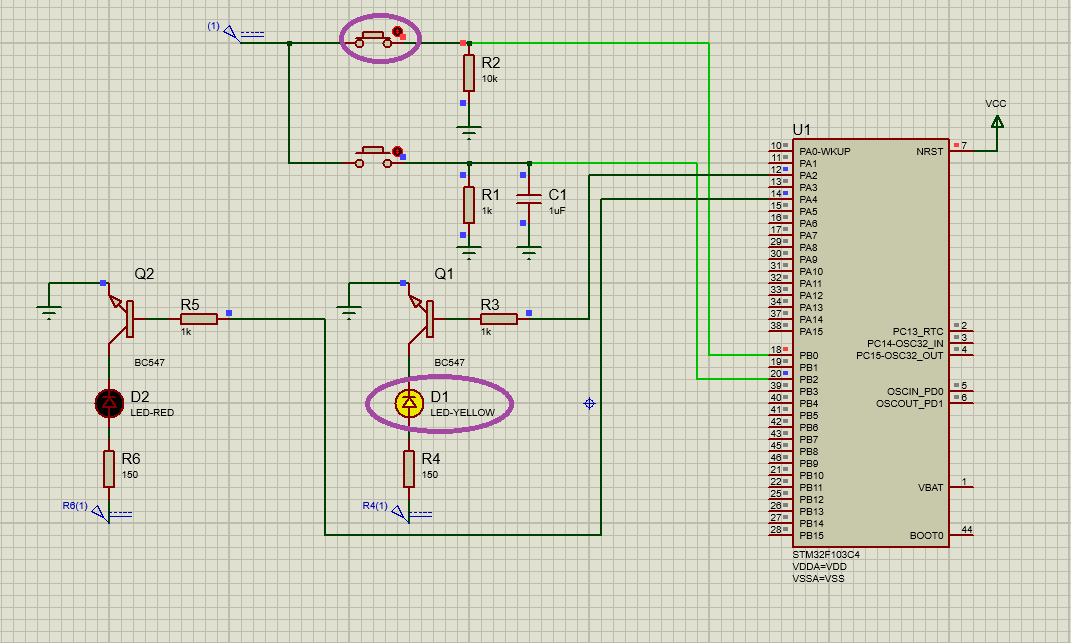


Рис. 3 Після натиску на кнопку 1, загоряється жовтий світлодіод

*Лістинг програми*

Таймери:

main.c

**while** (1)

{

/\* USER CODE END WHILE \*/

//if(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0)==GPIO\_PIN\_SET)

**if**(GPIOB->IDR & GPIO\_IDR\_IDR0) //якщо кнопка на РВ0 натиснута

{

HAL\_Delay(250); // встановлюємо затримку для нейтралізації "тремтіння контактів"

**if**(GPIOB->IDR & GPIO\_IDR\_IDR0) // якщо кнопка все ще нетиснута то виконуємо код

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_2, *GPIO\_PIN\_SET*); // включаємо світлодіод на піні РA2

HAL\_TIM\_Base\_Start\_IT(&htim1); // запускаємо таймер та переривання від таймера ТІМ1

HAL\_Delay(16000); // затримка протягом якої період роботи таймера не змінюється

HAL\_TIM\_Base\_Stop\_IT(&htim1); // зупиняємо таймер ТІМ1 і забороняємо його перериваання

\_\_HAL\_TIM\_SET\_AUTORELOAD(&htim1, 0x07CF); //завантажуємо нове значення регістру перезавантаження таймеру у 1999 у hex-форматі.При нашому подільнику частоти період таймеру складатиме 1 секунду.

\_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0x0000); // завантажуємо нульове значення у розрахунковий регістр для відрахунку часовогг інтервалу спочатку

HAL\_TIM\_Base\_Start\_IT(&htim1); // знову запускаємо таймер ТІМ1 і дозволяємо його переривання

}

}

/\* USER CODE BEGIN 3 \*/

}

…

**static** **void** **MX\_TIM1\_Init**(**void**)

{

/\* USER CODE BEGIN TIM1\_Init 0 \*/

/\* USER CODE END TIM1\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM1\_Init 1 \*/

/\* USER CODE END TIM1\_Init 1 \*/

htim1.Instance = TIM1;

htim1.Init.Prescaler = 7999;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 3999;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim1) != *HAL\_OK*)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != *HAL\_OK*)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM1\_Init 2 \*/

/\* USER CODE END TIM1\_Init 2 \*/

}

stm32f1xx\_it.c

**void** **TIM1\_UP\_IRQHandler**(**void**)

{

/\* USER CODE BEGIN TIM1\_UP\_IRQn 0 \*/

HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_4); // блимаємо світлодіодом при приході переривання переповненням

/\* USER CODE END TIM1\_UP\_IRQn 0 \*/

HAL\_TIM\_IRQHandler(&htim1);

/\* USER CODE BEGIN TIM1\_UP\_IRQn 1 \*/

/\* USER CODE END TIM1\_UP\_IRQn 1 \*/

}

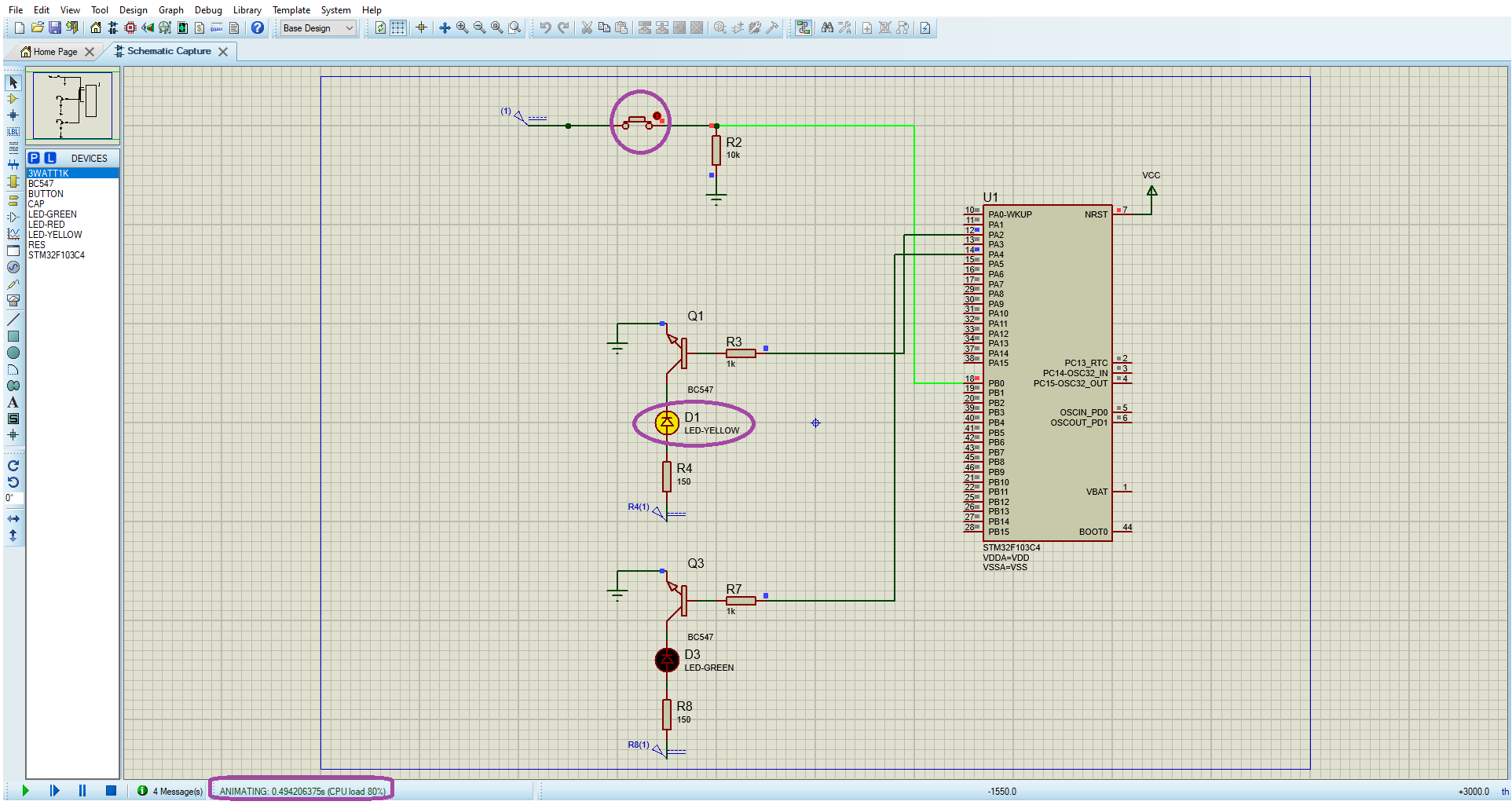


Рис. 4 Натиснута кнопка 1, загорівся світлодіод, запуск таймеру

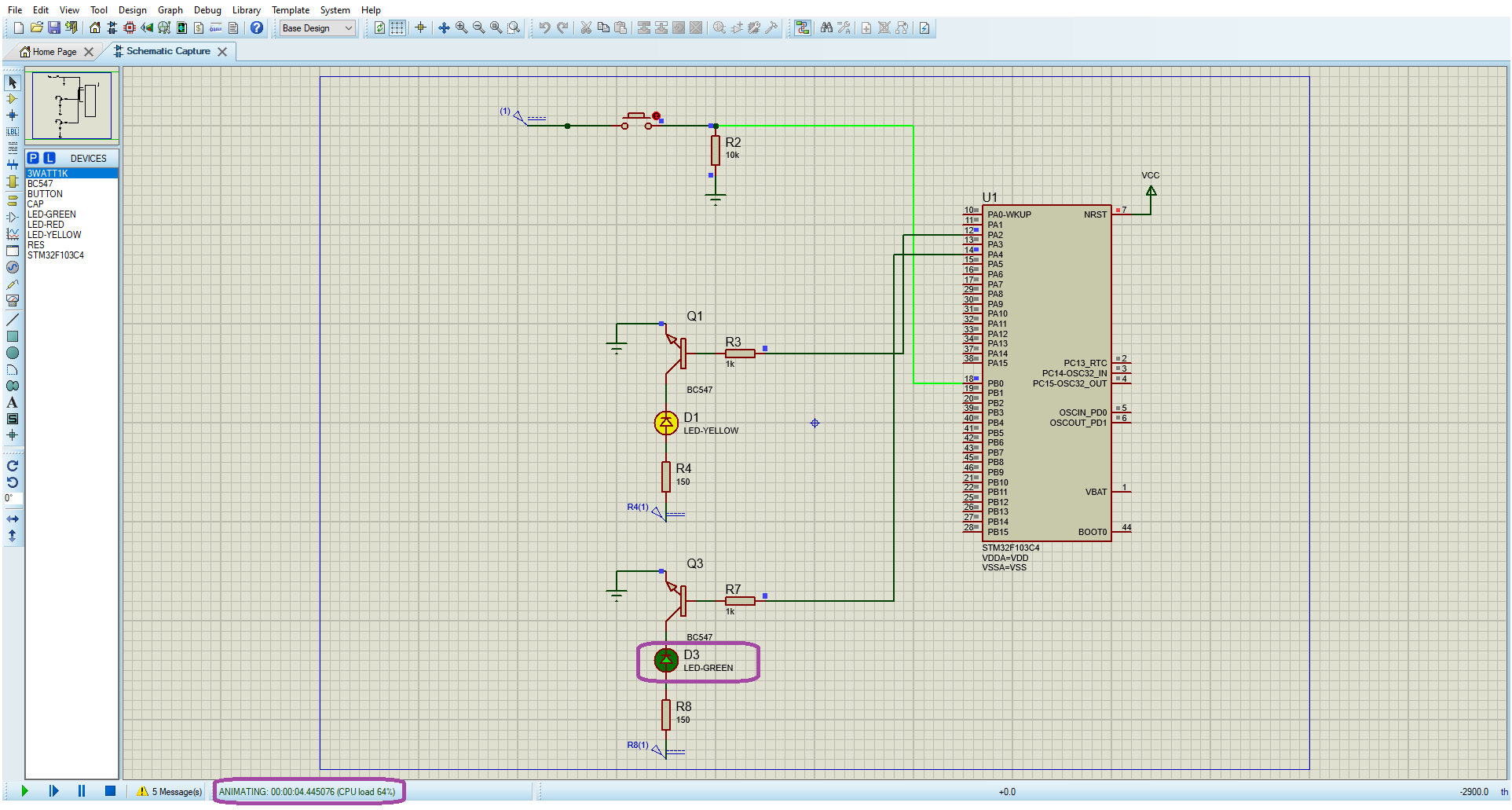


Рис. 5 Після 4 секунд загорівся зелений світлодіод

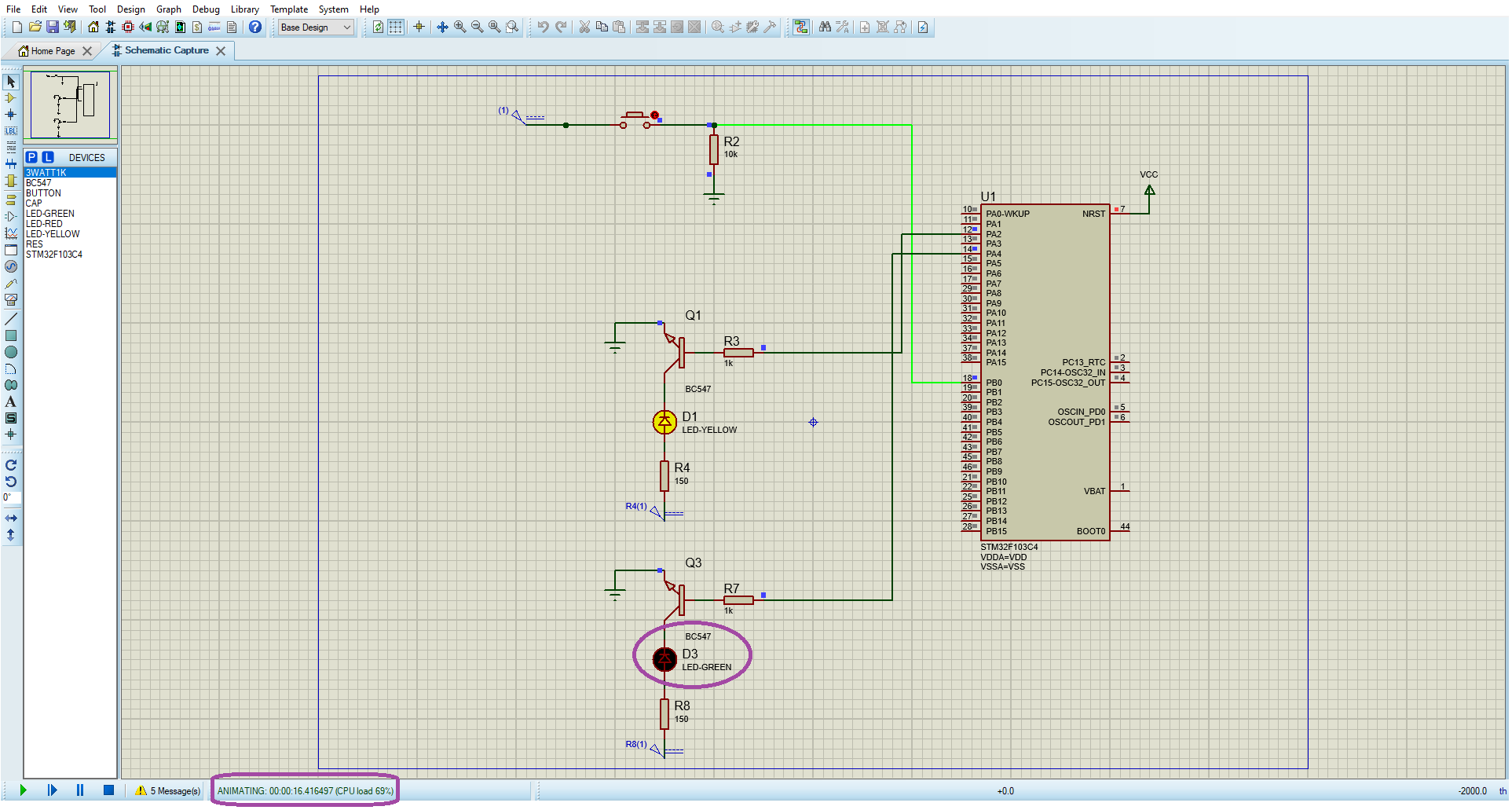


Рис. 6 Зміна часового інтервалу на 2 секунду

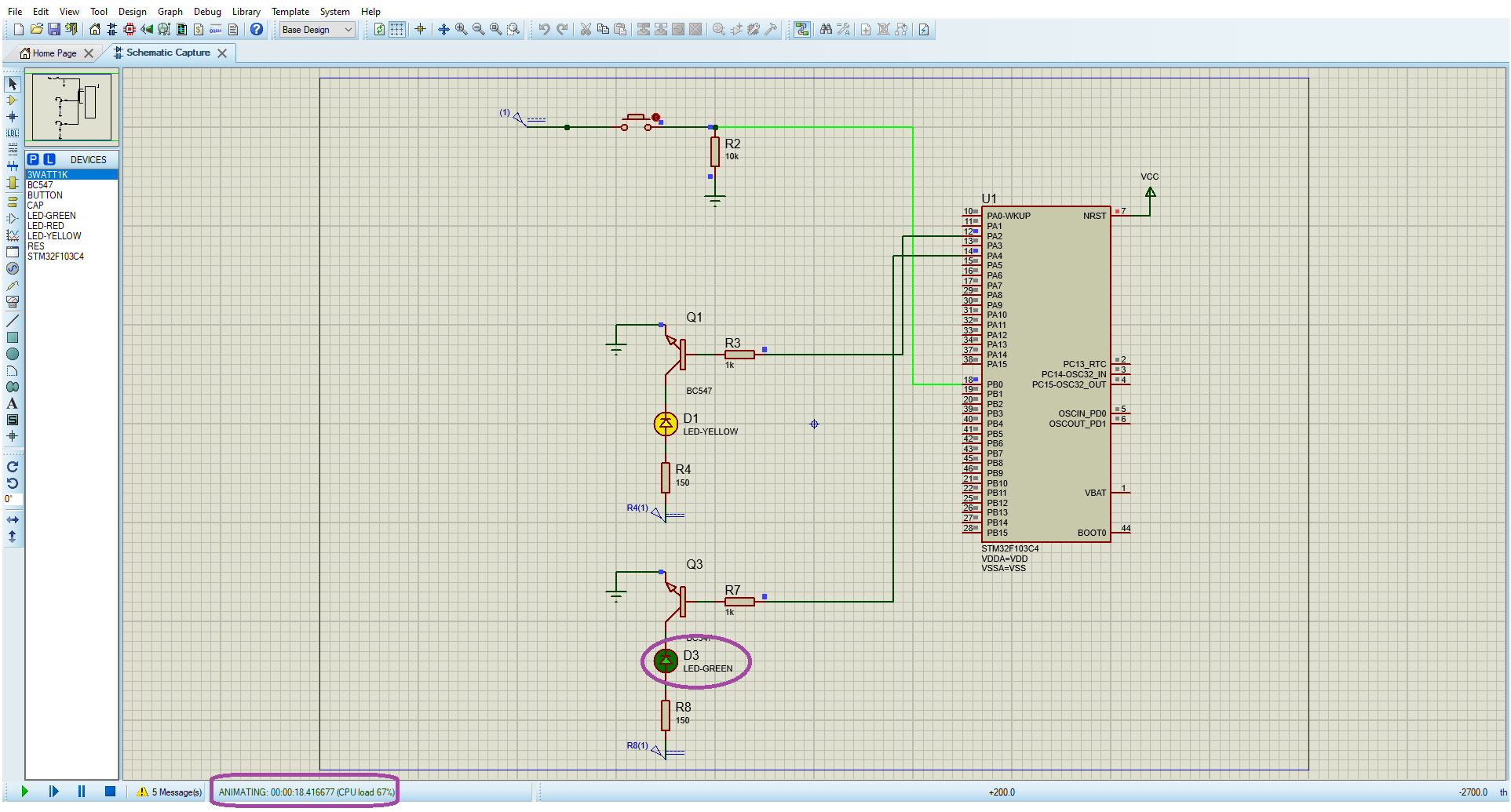


Рис. 7 Загоряння світло діоду через 2 секунди

***Висновок:***

У ході виконання лабораторної роботи ми успішно сконфігурували вхідно-вихідні порти, програмували безкінечне опитування та зовнішні переривання, а також налаштували таймери з заданими інтервалами. Реалізація моделі в Proteus дозволила нам продемонструвати функціонал розробленої програми і вирішити проблему дребезгу контактів на програмному та апаратному рівнях. Результати роботи підтвердили важливість глибокого розуміння принципів роботи мікропроцесорних систем та здатність застосовувати ці знання на практиці, що є ключовим для розробки сучасних електронних пристроїв.