МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи № 11
з дисципліни: *"Архітектура комп'ютера"*
на тему:*"Використання цифрових портів мікроконтролера STM32F401VE"*

Лектор:
доц. каф. ПЗ
Крук О.Г.

Виконав:
ст. гр. ПЗ-22
Солтисюк Д.А.

Прийняв:
доц. каф. ПЗ
Крук О.Г.

« ____ » _____ 2022 р.

Σ= _____ _____

Львів – 2022

Тема роботи: Використання цифрових портів мікроконтролера STM32F401VE.

Мета роботи: опанувати роботу з цифровими портами мікроконтролера STM32F401VE; розвинути навики складання програми мовою С для виведення і введення сигналів через цифрові порти; відтранслювати програму, складену відповідно до свого варіанту в середовищі програмування Keil µVision MDK-ARM; виконати моделювання схеми з мікроконтролером в системі Proteus.

Варіант: 22

22	C	6	E	6

Теоретичні відомості

Цифрові порти введення-виведення загального призначення

Кожний мікроконтролер має цифрові лінії введення або виведення. Кожну таку лінію можна програмним шляхом конфігурувати як цифровий вхід, або цифровий вихід, і використовувати для взаємодії із зовнішніми схемами. Для зручності використання лінії введення-виведення об'єднані в порти по 16 ліній. Такі порти називають портами введення-виведення загального призначення. В англомовній літературі лінії введення-виведення прийнято називати терміном GPIO - General-Purpose Input / Output.

До ліній, сконфігурованих як цифрові входи, під’єднують механічні кнопки, вимикачі, контакти реле, давачі тощо. За допомогою таких ліній мікроконтролер отримує інформацію від під’єднаних до нього пристроїв.

Лінії, сконфігуровані як цифрові виходи, дозволяють видавати сигнали керування для під’єднаних до мікроконтролера пристроїв. Таким сигналом можна безпосередньо засвітити світлодіод, а через відповідну схему можна запустити електродвигун, увімкнути електромагнітне реле або лампу розжарювання тощо.

Конфігурування ліній введення-виведення

Для того щоб почати використовувати лінії введення-виведення, потрібно попередньо конфігурувати їх відповідним чином.

На найнижчому рівні робота з портами введення-виведення (та й з усіма іншими периферійними пристроями) здійснюється за допомогою спеціальних регістрів мікроконтролера. Ці регістри доступні як комірки пам'яті, розташовані за певними адресами. Знаючи ці адреси (вони описані в документації на мікроконтролер), можна записувати в регістри певні значення, задаючи необхідну конфігурацію. Через інші регістри можна отримувати дані від периферійних пристроїв.

Портів введення-виведення загального призначення (GPIO – General Purpose Input Output) може бути різна кількість, у нашому випадку є 5 портів GPIO: A, B, C, D і E.

Кожен порт є 16-бітовим (має 16 ліній) і використовує десять 32-бітових регістрів:

чотири регістри конфігурації (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR);

два регістри даних (GPIOx_IDR, GPIOx_ODR);

регістр встановлення/скидання (GPIOx_BSRR);

регістр блокування (GPIOx_LCKR);

два регістри вибору альтернативної функції (GPIOx_AFRH, GPIOx_AFRL)

(x заміняє ім’я порту).

Якщо світлодіоди керуються лише портом D, надалі замість абстрактної назви GPIOx застосовуємо GPIOD. GPIOD_MODER дозволяє сконфігурувати напрямок даних. Для роботи зі світлодіодом слід сконфігурувати порт на вихід. З документації мікроконтролера (пункт 8.4.1), фрагмент якої показано на рис. 1, видно, що для цього слід задати значення 01 (тут значенням керує комбінація з 2 бітів).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Рис. 1. *Можливі значення регістра GPIOD_MODER*

Наразі приймемо, що GPIOD_OTYPER = 0.

Регістр GPIOD_SPEEDR відповідає за швидкість (є 4 рівні швидкості, відповідно, рівень швидкості кодується 2 бітами). Значення 00 відповідає найменшій швидкості, достатньо задати значення 0 для всіх виводів. Тому GPIOD_SPEEDR = 0.

Оскільки напрям передавання порту – на вихід, то з регістрів даних потрібно налаштовувати GPIOD_ODR (Output Data Register), а не GPIOD_IDR (Input Data Register).

У 32-бітовому регістрі GPIOD_ODR старші 16 бітів не використовуються (зарезервовані), а молодші 16 відповідають якраз виводам 16-розрядного порту (рис. 2).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

Рис. 2. *Регістр GPIOx_ODR*

Щоб засвітити світлодіод, треба подати 1 на відповідний пін. Відповідно якщо GPIOD_ODR = 0x9000, це відповідає числу 1001000000000000 у двійковій системі, тобто, згідно зі схемою на рис. 3, таким чином засвічуємо синій і зелений світлодіоди.

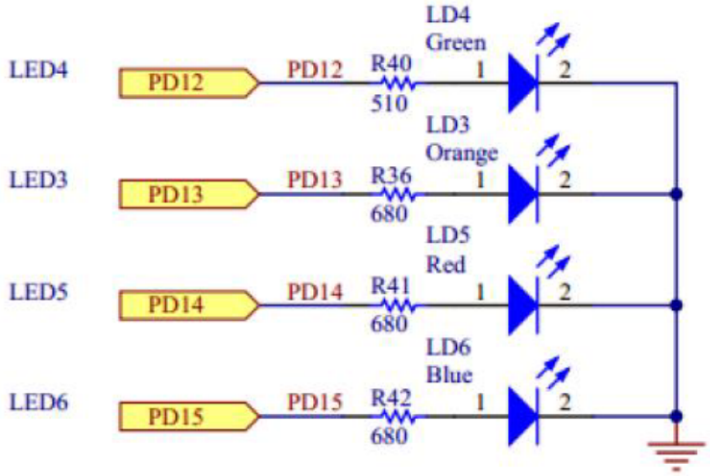

Рис. 3*. Підключення світлодіодів до виводів/пінів порту D*

PD – це порт D, числа 12, 13, 14, 15 – це номери виводів/пінів/ніжок цього порту.

Для забезпечення мінімального споживання енергії периферія у початковому стані відключена від живлення. Жоден з периферійних модулів мікроконтролера не працюватиме, поки на нього не надходитимуть тактові імпульси. Тому перш ніж щось робити з тим чи іншим периферійним модулем, спочатку слід увімкнути тактування.

Власне, це і робиться наступний рядком (див. лістинг 1):

```
RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;
```

RCC – Reset & Clock Control – блок регістрів для керування тактуванням. AHB1ENR – один з регістрів у цьому блоці (AHB1 – шина, ENR – Enable Register).

У ARM Cortex є шини AHB (ARM Hi-Speed Bus) та APB (ARM Peripheral Bus). Отже, AHB1 – одна з шин. Молодші 9 бітів регістра AHB1ENR відповідають за тактування 9 портів (наймолодший біт – за порт A, наступний за ним – за порт B тощо). Очевидно, що за порт D відповідає біт на 4-ій позиції справа. Щоб увімкнути тактування для порту D , не порушивши тактування інших портів, слід застосувати побітове «або», що і зроблено у коді-прикладі.

Константа RCC_AHB1ENR_GPIODEN є числом ….000001000. Щоб дізнатися, яким регістром слід скористатися для тактування певного порту, слід заглянути або у файли бібліотеки CMSIS, або у документацію на мікроконтролер (Reference Manual). У документації ці дані містяться у пункті 7.3.25 – RCC Register Map. Зі фрагмента таблиці, поданої у цьому пункті (рис. 4), видно, що тактування порту D вмикається регістром AHB1ENR.

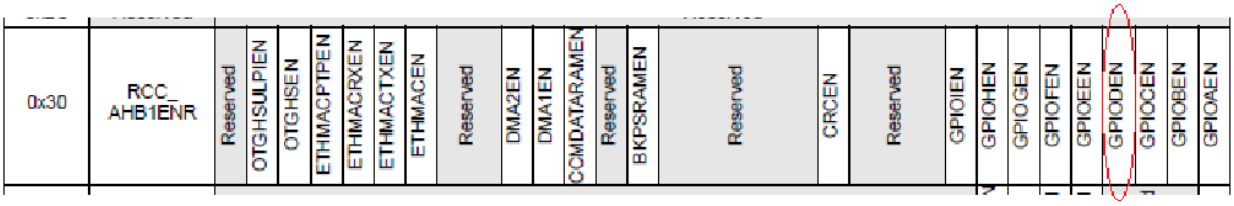

Рис. 4. Фрагмент документації для визначення регістра тактування певного порту

Для полегшення праці програміста багато розробників мікроконтролерів надають спеціальні бібліотеки функцій для роботи з периферійними пристроями.

Бібліотека CMSIS

CMSIS – бібліотека, стандартна для всіх МК з ядром ARM Cortex. Стандартизується ARM Ltd. Різні виробники МК з цим ядром доповнюють CMSIS файлами з описом периферійних модулів, специфічних для МК, які вони випускають.

Бібліотека надає зручний доступ до периферійних модулів, її застосування спрощує процес розроблення.

Бібліотека SPL

Standard Peripheral Library (SPL) - бібліотека, яка розробляється компанією STMicroelectronics і призначена для полегшення програмування мікроконтролерів виробництва цієї компанії.

Індивідуальне завдання

1. В середовищі програмування Keil µVision MDK-ARM створіть проєкт, наведений в інструкції, з використанням програми з лістингу 1 або з лістингу

2. Відкомпілюйте програму, створіть проєкт і в режимі відлагоджувача дослідіть його роботу (відслідкуйте стан регістра GPIOD_ODR).
3. Створіть за аналогією проєкт відповідно до свого варіанту (без миготіння) і збережіть його (в тому числі і hex-файл).
4. В системі Proteus введіть схему відповідно до свого варіанту. (для мікроконтролера в полі пошуку введіть stm; для кнопки – but; для світлодіода - led-; для "землі" активізуйте Terminals mode виберіть Ground).
5. Запустіть моделювання і натискайте та відпускайте кнопку.
6. Перевірте роботу програми.
7. У звіті наведіть структуру проєкту, текст програми, копії вікна з схемою.
8. Зробіть висновки про виконану роботу.

Завдання на лабораторну роботу №11: скласти програму для мікроконтролера STM32F401VE, яка працює таким чином: якщо кнопка, яка під'єднується до k-го біта порту W і до "землі", натиснута, то світиться світлодіод, який під'єднаний до m-го біта порту V, в протилежному випадку – світлодіод гасне.

Варіант: 22

22	C	6	E	6

Хід роботи

Код програми з "Лістингу 1":

```
#include <stm32f4xx.h>
uint16_t delay_c = 0;
void SysTick_Handler(void){
	if(delay_c > 0)
	delay_c--;
}
void delay_ms(uint16_t delay_t){
	delay_c = delay_t;
	while(delay_c){};
}
int main (void){
	SysTick_Config(SystemCoreClock/1000);
	RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;
	GPIOD->MODER = 0x55000000;
	GPIOD->OTYPER = 0;
	GPIOD->OSPEEDR = 0;
	while(1){
		GPIOD->ODR = 0x9000;
		delay_ms(500);
```

```
            GPIOD->ODR = 0x0000;
            delay_ms(500);
        }
}
```

Створив проєкт з використанням програми з “Лістингу 1”. Відкомпілював програму і в режимі відлагоджувача відслідкував стан регістра GPIOD_ODR:

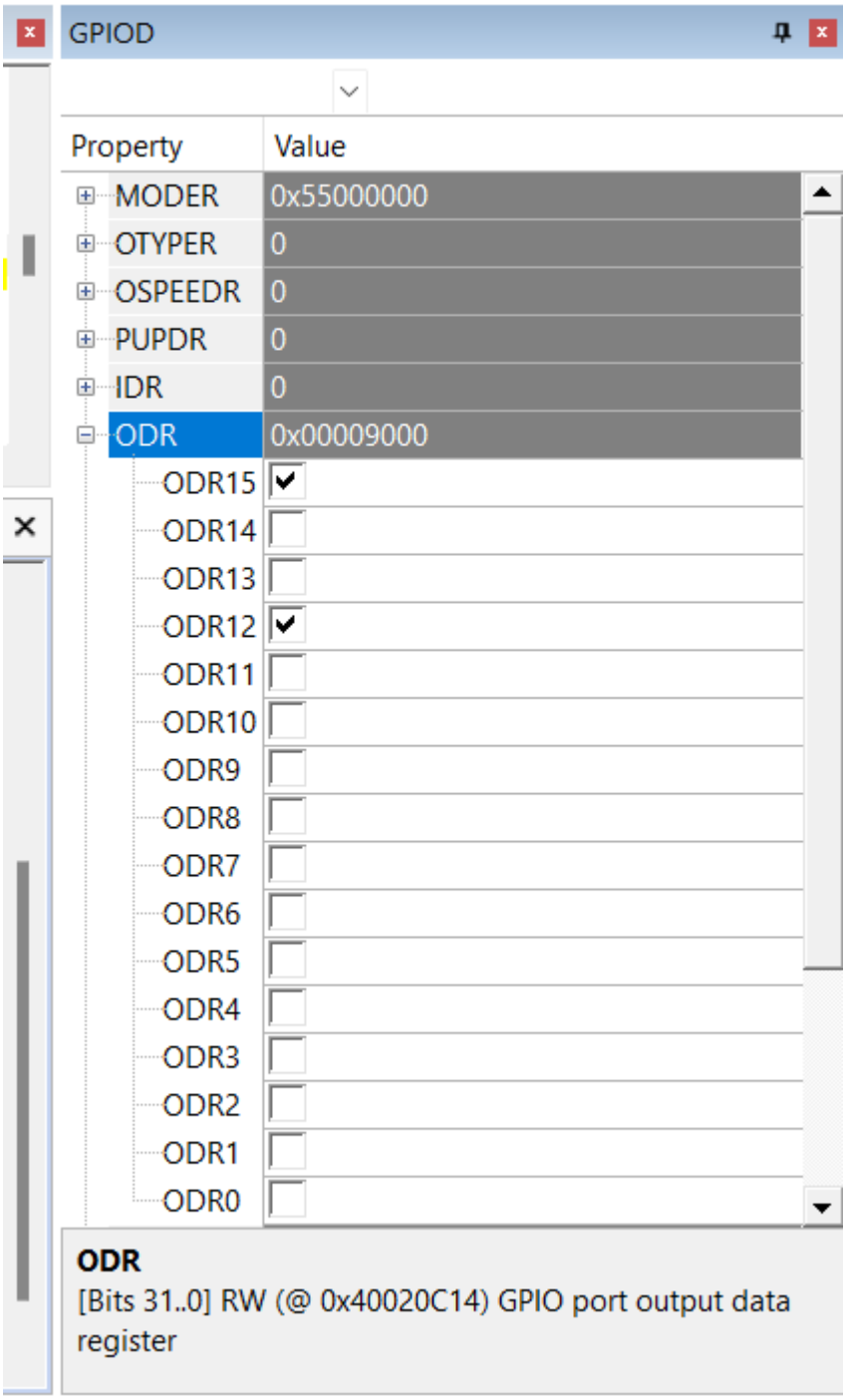

Рис 1. *GPIOD->ODR = 0x00009000*

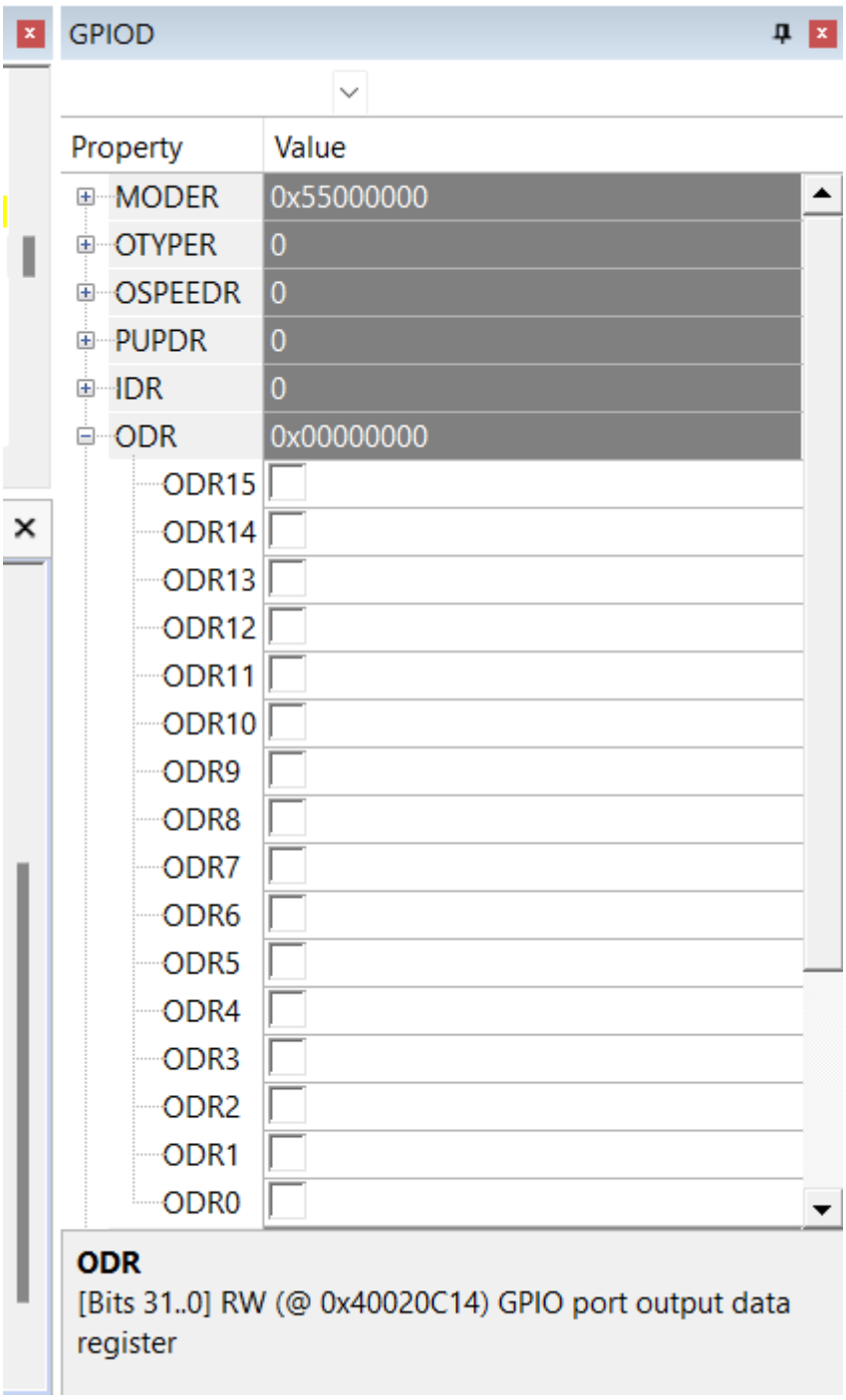

Рис 2. *GPIOD->ODR = 0x00009000*

Код програми lab11.c

```
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"


static GPIO_InitTypeDef PORTC6;
static GPIO_InitTypeDef PORTE6;

int main(void)
{
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC |
RCC_AHB1Periph_GPIOE,ENABLE);

		GPIO_StructInit(&PORTC6);
		PORTC6.GPIO_Pin = GPIO_Pin_6;
		PORTC6.GPIO_Mode = GPIO_Mode_IN;
		GPIO_Init(GPIOC, &PORTC6);

		GPIO_StructInit(&PORTE6);
		PORTE6.GPIO_Pin = GPIO_Pin_6;
		PORTE6.GPIO_Mode = GPIO_Mode_OUT;
		GPIO_Init(GPIOE, &PORTE6);

		while (1)
		{
			if(GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_6) == 0)
			{
				GPIO_SetBits(GPIOE,GPIO_Pin_6);
			}
			else
			{
				GPIO_ResetBits(GPIOE,GPIO_Pin_6);
			}
		}
}
```

Створив проєкт та склав програму ***lab11.c*** відповідно до свого варіанту. Програма розроблена для мікроконтролера STM32F401VE, яка працює таким чином: якщо кнопка, яка під’єднується до 13-го біта порту C і до "землі", натиснута, то світиться світлодіод, який під’єднаний до 6-го біта порту D, в протилежному випадку – світлодіод гасне.

В системі Proteus ввів схему відповідно до свого варіанту. Підключив до схеми складену програму та запустив моделювання:

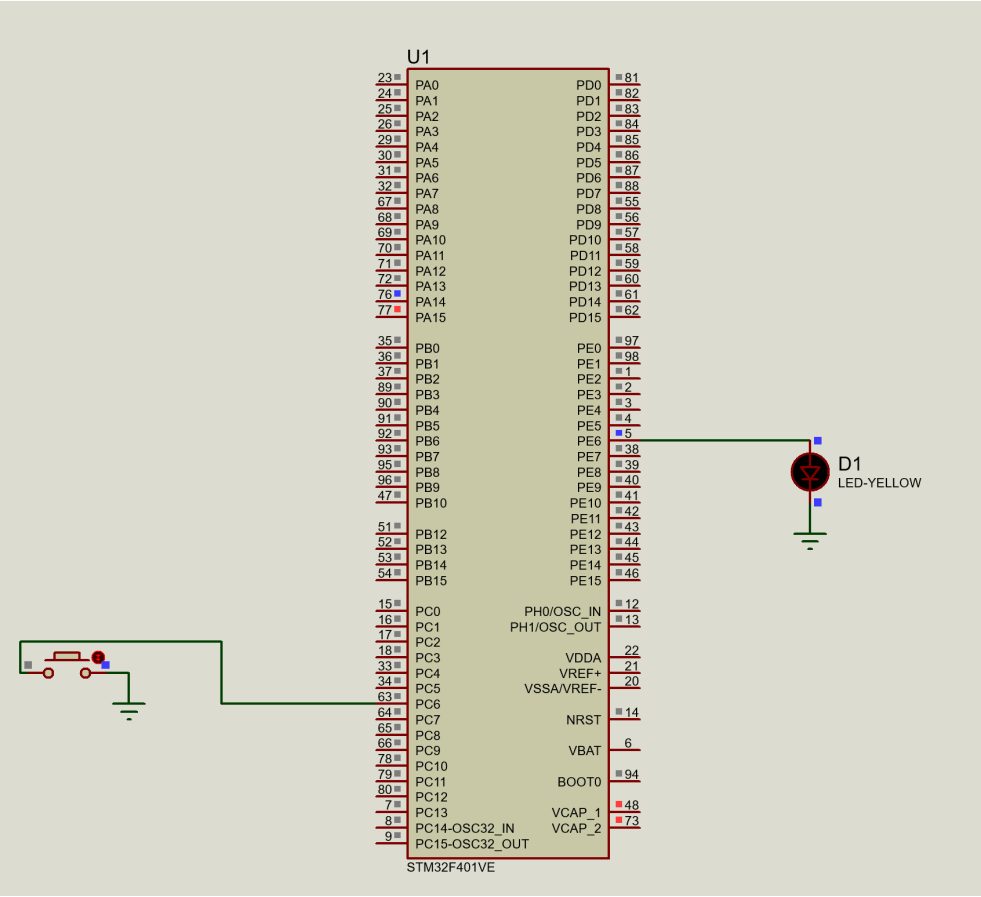

Рис 3. *Кнопка не натиснута, світлодіод не горить*

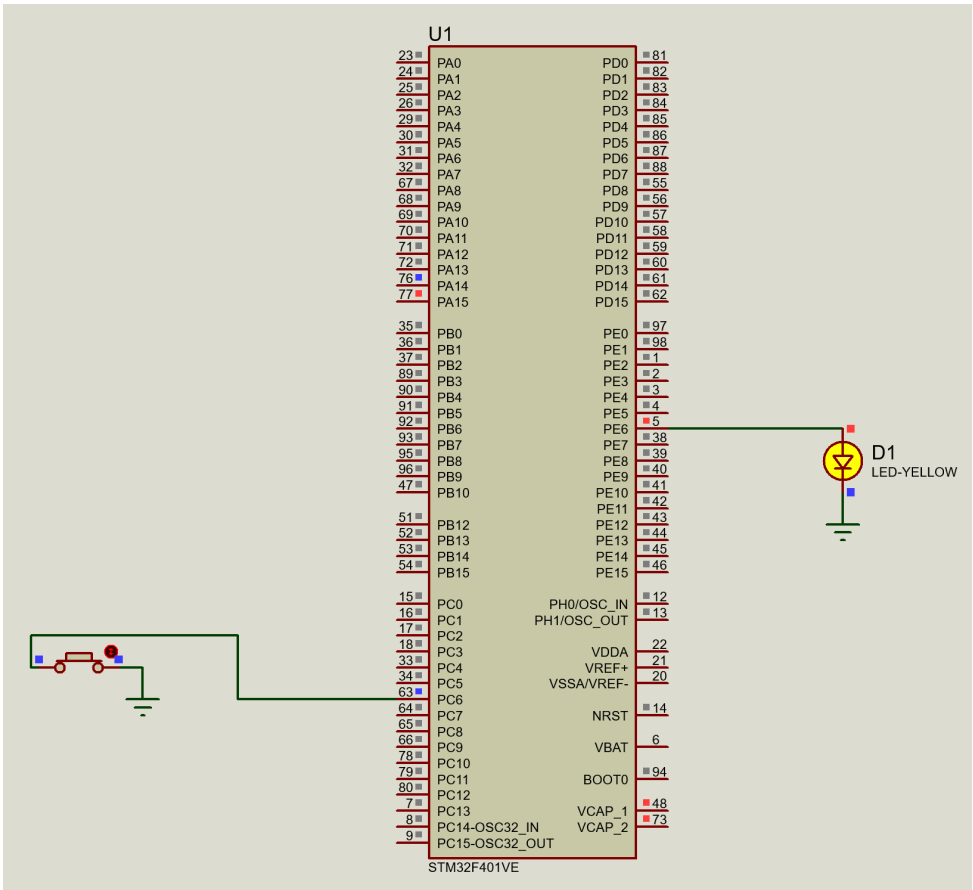

Рис 4. *Кнопка натиснути, світлодіод горить*

Висновки

У результаті виконання лабораторної роботи я опанував роботу з цифровими портами мікроконтролера STM32F401VE; розвинув навики складання програми мовою С для виведення і введення сигналів через цифрові порти; відтранслював програму, складену відповідно до свого варіанту в середовищі програмування Keil µVision MDK-ARM; виконав моделювання схеми з мікроконтролером в системі Proteus.