

VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



MICROPROCESSOR MICRONTROLLER (CO3009)

LAB 1

LED ANIMATION

Class: L04 – Semester HK251
Submission Date: 18/09/2025

Instructor: Mr. Ton Huynh Long

| Student Name | Student ID |
|--------------|------------|
| Phạm Công Võ | 2313946 |

Ho Chi Minh City, September 2025



Table of Contents

| | | |
|----------|-----------------------------------|-----------|
| 1 | Introduction | 3 |
| 2 | First project on STM32Cube | 4 |
| 3 | Simulation on Proteus | 9 |
| 4 | Exercise and Report | 15 |
| 4.1 | Exercise 1 | 15 |
| 4.2 | Exercise 2 | 16 |
| 4.3 | Exercise 3 | 18 |
| 4.4 | Exercise 4 | 20 |
| 4.5 | Exercise 5 | 22 |
| 4.6 | Exercise 6 | 30 |
| 4.7 | Exercise 7 | 33 |
| 4.8 | Exercise 8 | 34 |
| 4.9 | Exercise 9 | 35 |
| 4.10 | Exercise 10 | 36 |
| | References | 38 |

List of Figures

| | | |
|----|---|----|
| 1 | <i>STM32Cube IDE for STM32 Programming</i> | 3 |
| 2 | <i>Create a new project on STM32CubeIDE</i> | 4 |
| 3 | <i>Select the target device</i> | 4 |
| 4 | <i>Select the target device</i> | 4 |
| 5 | <i>Keep default firmware version</i> | 5 |
| 6 | <i>Set PA5 to GPIO Output mode</i> | 5 |
| 7 | <i>Provide a name for PA5</i> | 6 |
| 8 | <i>Config for hex file output</i> | 7 |
| 9 | <i>Compile the project and generate Hex file</i> | 8 |
| 10 | <i>Create a new project on Proteus</i> | 9 |
| 11 | <i>Provide project name and location</i> | 9 |
| 12 | <i>Keep the default options by clicking on Next</i> | 9 |
| 13 | <i>Finish the project wizard</i> | 10 |
| 14 | <i>Select a component from the library</i> | 10 |
| 15 | <i>Select STM32F103C6</i> | 11 |
| 16 | <i>STM32 and an LED in the project</i> | 11 |
| 17 | <i>Place components to the project</i> | 12 |
| 18 | <i>Connect components and set the power to 3.3V</i> | 12 |
| 19 | <i>Place label for Power and Ground</i> | 13 |
| 20 | <i>Finalize the schematic</i> | 13 |
| 21 | <i>Set the program of the STM32 to the hex file from Cube IDE</i> | 14 |
| 22 | <i>Quick access buttons to start and stop the simulation</i> | 14 |
| 23 | <i>State transitions for 2 LEDs</i> | 15 |
| 24 | <i>Proteus Ex1 ToggleLED</i> | 15 |
| 25 | <i>Proteus Ex2 TrafficLight</i> | 17 |
| 26 | <i>Reference design for a 4 way traffic light</i> | 18 |
| 27 | <i>Proteus Ex3 Four Way TrafficLight</i> | 18 |
| 28 | <i>Display a number on 7 segment LED</i> | 20 |
| 29 | <i>Proteus Ex4 7SEG Display7SEG</i> | 20 |
| 30 | <i>Proteus Ex5 Traffic with 7SEG</i> | 23 |
| 31 | <i>12 LEDs for an analog clock</i> | 31 |
| 32 | <i>Proteus Ex6 AnalogClock 12LEDs</i> | 31 |

1 Introduction

In this manual, the STM32CubeIDE is used as an editor to program the ARM micro-controller. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors.



Figure 1: *STM32Cube IDE for STM32 Programming*

The most interest of STM32CubeIDE is that after the selection of an empty STM32 MCU or MPU, or preconfigured microcontroller or microprocessor from the selection of a board, the initialization code generated automatically. At any time during the development, the user can return to the initialization and configuration of the peripherals or middleware and regenerate the initialization code with no impact on the user code. This feature can simplify the initialization process and speedup the development application running on STM32 micro-controller. The software can be downloaded from the link bellow:

https://ubc.sgp1.digitaloceanspaces.com/BKU_Softwares/STM32/stm32cubeide_1.7.0.zip

Moreover, for a hangout class, the program is firstly simulated on Proteus. Students are also supposed to download and install this software as well:

https://ubc.sgp1.digitaloceanspaces.com/BKU_Softwares/STM32/Proteus_8.10_SP0_Pro.exe

The rest of this manual consists of:

- Create a project on STM32Cube IDE
- Create a project on Proteus
- Simulate the project on Proteus

Finally, students are supposed to finish 10 different projects.

2 First project on STM32Cube

Step 1: Launch STM32CubeIDE, from the menu **File**, select **New**, then chose **STM32 Project**

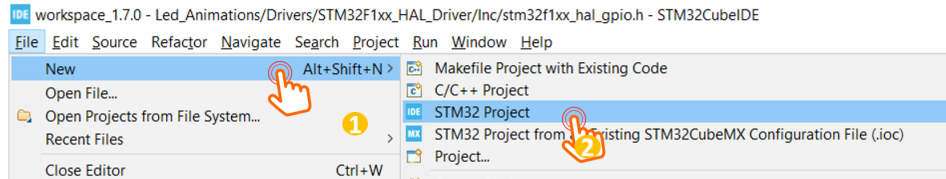


Figure 2: Create a new project on STM32CubeIDE

The IDE needs to download some packages, which normally takes time in this first time a project is created.

Step 2: Select the STM32F103C6 in the following dialog, then click on **Next**

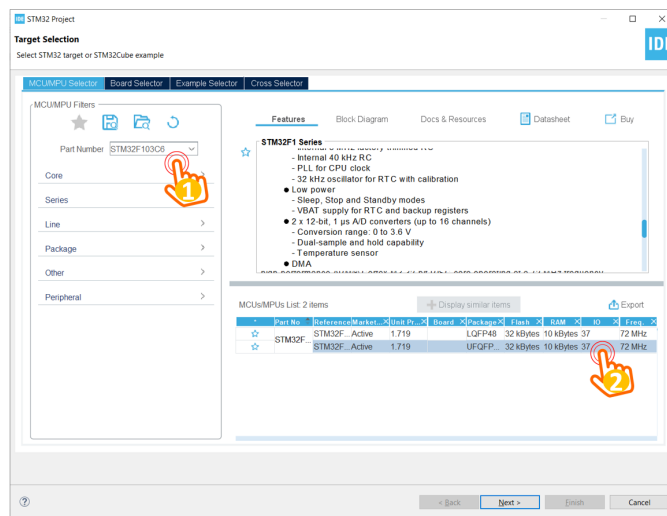


Figure 3: Select the target device

Step 3: Provide the **Name** and the **Location** for the project.

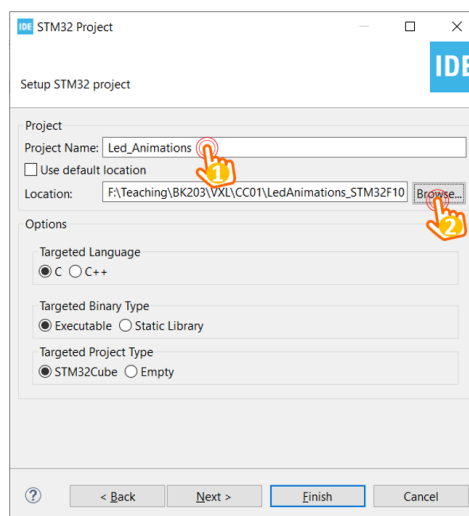


Figure 4: Select the target device

It is important to notice that the **Targeted Project Type** should be **STM32Cube**. In the case this option is disable, step 1 must be repeated. The location path should not contain special characters

(e.g. the space). Finally, click on the **Next** button.

Step 4: On the last dialog, just keep the default firmware version and click on **Finish** button.

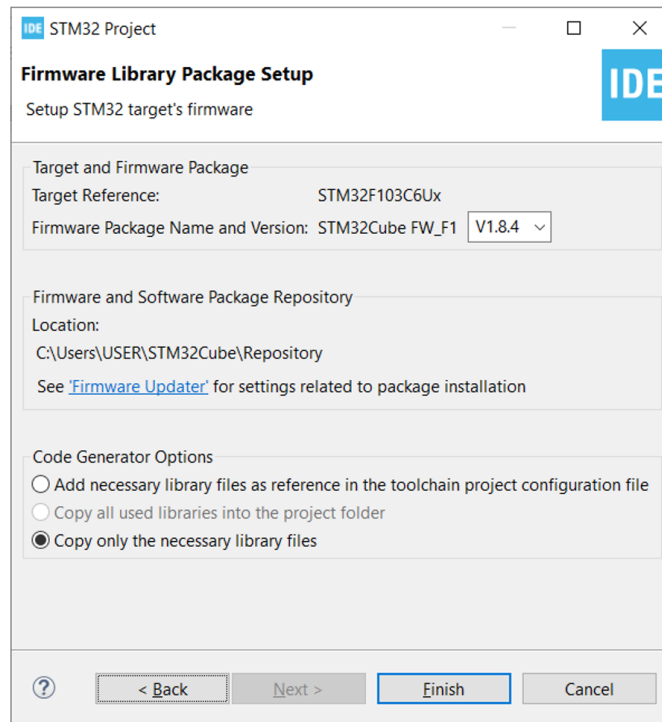


Figure 5: Keep default firmware version

Step 5: The project is created and the wizard for configuration is display. This utility from CubeIDE can simplify the configuration process for an ARM micro-controller like the STM32.

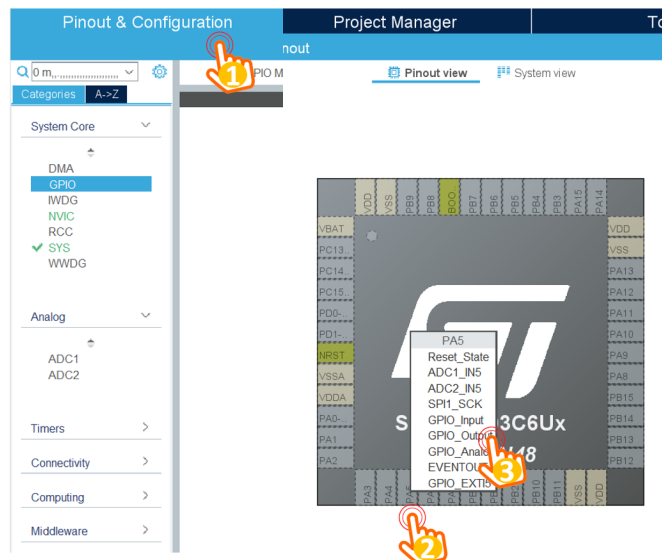


Figure 6: Set PA5 to GPIO Output mode

From the configuration windows, select **Pin configuration**, select the pin **PA5** and set to **GPIO Output** mode, since this pin is connected to an LED in the STM32 development kit.

Step 6: Right click on PA5 and select **Enter user label**, and provide the name for this pin (e.g. **LED_RED**). This step helps programming afterward more memorable.

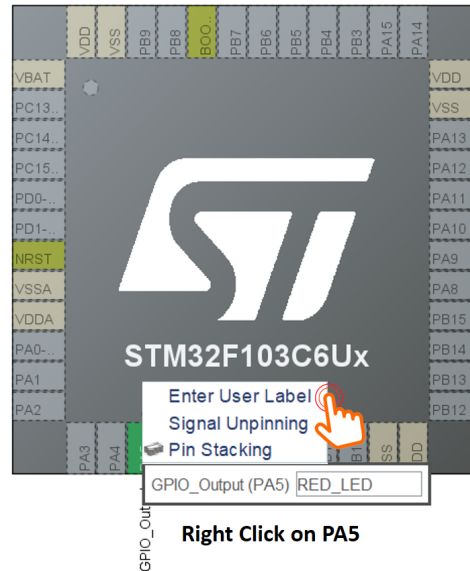


Figure 7: Provide a name for PA5

Finally, save the configuration process by pressing **Ctrl + S** and confirm this step by clicking on **OK** button. The code generation is started.

Step 7: Implement the first blinky project in the main function as follow:

```

1  int main(void)
2  {
3      /* MCU Configuration
4       -----*/
5      /* Reset of all peripherals, Initializes the Flash interface and
6       the SysTick. */
7      HAL_Init();
8
9      /* USER CODE BEGIN Init */
10
11     /* USER CODE END Init */
12
13     /* Configure the system clock */
14     SystemClock_Config();
15
16     /* USER CODE BEGIN SysInit */
17
18     /* USER CODE END SysInit */
19
20     /* Initialize all configured peripherals */
21     MX_GPIO_Init();

```

```

20  /* USER CODE BEGIN 2 */
21
22  /* USER CODE END 2 */
23
24  /* Infinite loop */
25  /* USER CODE BEGIN WHILE */
26  while (1)
27  {
28      HAL_GPIO_TogglePin(LED_RED_GPIO_Port , LED_RED_Pin);
29      HAL_Delay(1000);
30      /* USER CODE END WHILE */
31      /* USER CODE BEGIN 3 */
32  }
33  /* USER CODE END 3 */
34  }

```

Actually, what is added to the main function is line number 34 and 35. Please put your code in a right place, otherwise it can be deleted when the code is generated (e.g. change the configuration of the project). When coding, frequently use the suggestions by pressing **Ctrl+Space**.

Step 8: Due to the simulation on Proteus, the hex file should be generated from STM32Cube IDE. From menu **Project**, select **Properties** to open the dialog below:

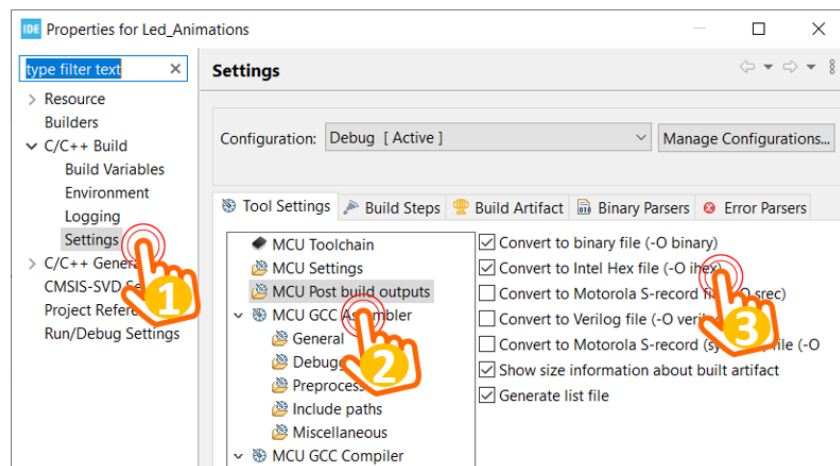


Figure 8: Config for hex file output

Navigate to **C/C++ Build**, select **Settings**, **MCU Post build outputs**, and check to the **Intel Hex file**.

Step 9: Build the project by clicking on menu **Project** and select **Build Project**. Please check on the output console of the IDE to be sure that the hex file is generated, as follow:

```
22:36:06 **** Incremental Build of configuration Debug for project Led_Animations ****
make -j8 all
arm-none-eabi-size  Led_Animations.elf
   text    data     bss      dec       hex filename
   4596     20    1572    6188    182c Led_Animations.elf
Finished building: default.size.stdout

22:36:06 Build Finished. 0 errors, 0 warnings. (took 272ms)
```

Figure 9: *Compile the project and generate Hex file*

The hex file is located under the **Debug** folder of your project, which is used for the simulation in Proteus afterward. In the case a development kit is connected to your PC, from menu **Run**, select **Run** to download the program to the hardware platform.

In the case there are multiple project in a work-space, double click on the project name to activate this project. Whenever a project is built, check the output files to make sure that you are working in a right project.

3 Simulation on Proteus

For an online training, a simulation on Proteus can be used. The details to create an STM32 project on Proteus are described bellow.

Step 1: Launch Proteus (with administration access) and from menu **File**, select **New Project**.



Figure 10: Create a new project on Proteus

Step 2: Provide the name and the location of the project, then click on **Next** button.

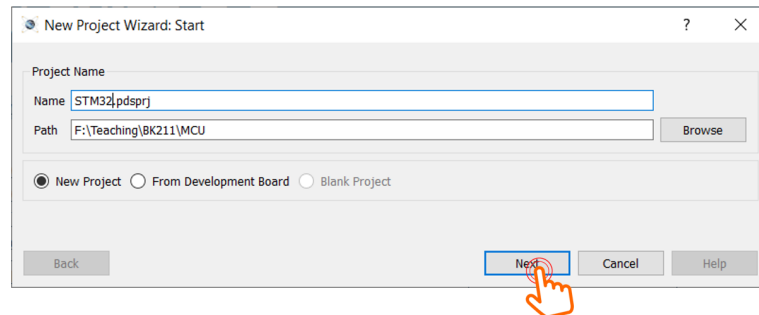


Figure 11: Provide project name and location

Step 3: For following dialog, just click on **Next** button as just a schematic is required for the lab.

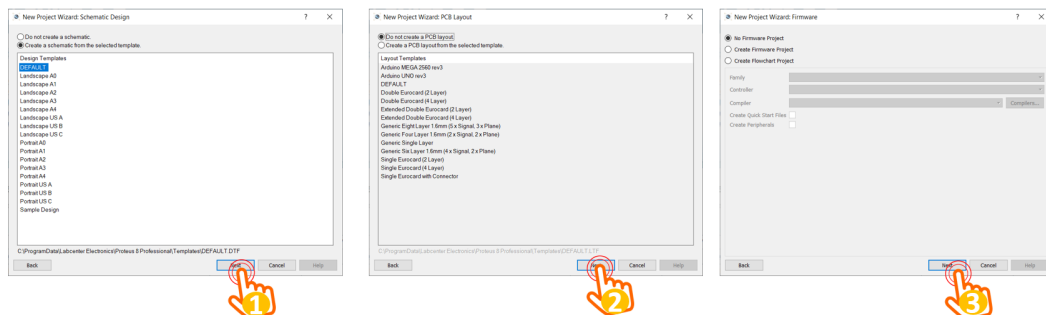


Figure 12: Keep the default options by clicking on Next

Step 4: Finally, click on **Finish** button to close the project wizard.

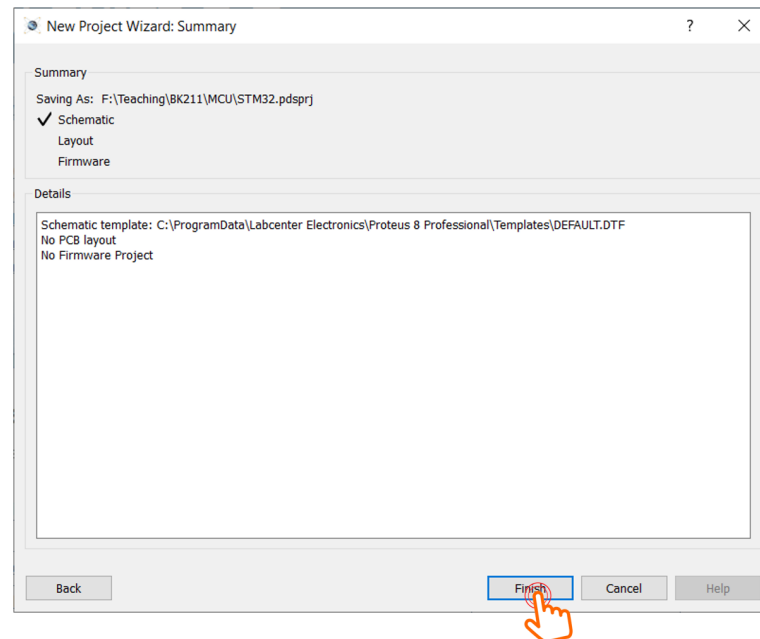


Figure 13: Finish the project wizard

Step 5: On the main page of the project, right click to select **Place, Components, From Libraries**, as follows:

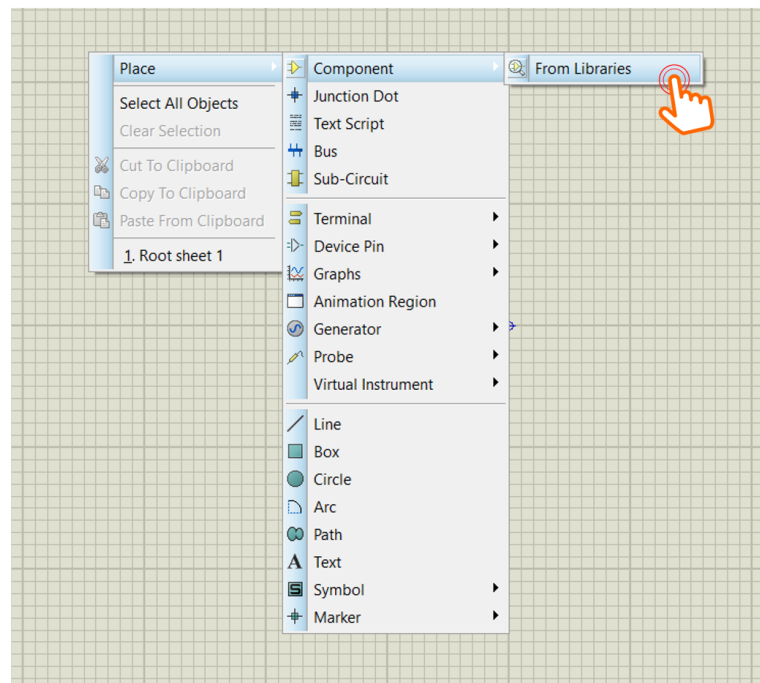


Figure 14: Select a component from the library

If there is an error with no library found, please restart the Proteus software with Run as administrator option.

Step 6: From the list of components in the library, select STM32F103C6, as follows:

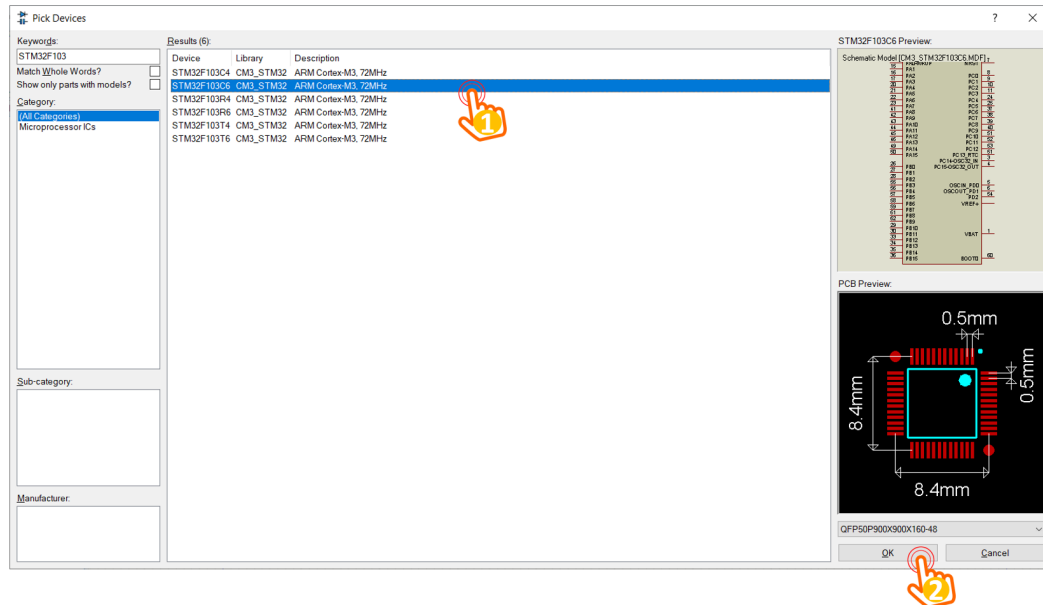


Figure 15: Select STM32F103C6

Repeat step 5 and 6 to select an LED, named **LED-RED** in Proteus. Finally, these components are appeared on the DEVICES windows, which is on left hand side as follows:

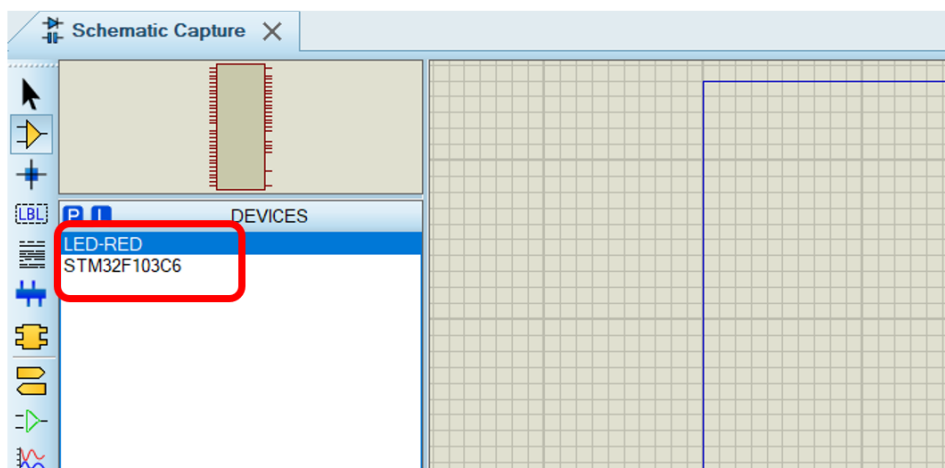


Figure 16: STM32 and an LED in the project

Step 7: Place the components to the project: right click on the main page, select on **Place, Component**, and select device added in Step 6. To add the Power and the Ground, right click on the main page, select on **Place, Terminal**. The result in this step is expected as follows:

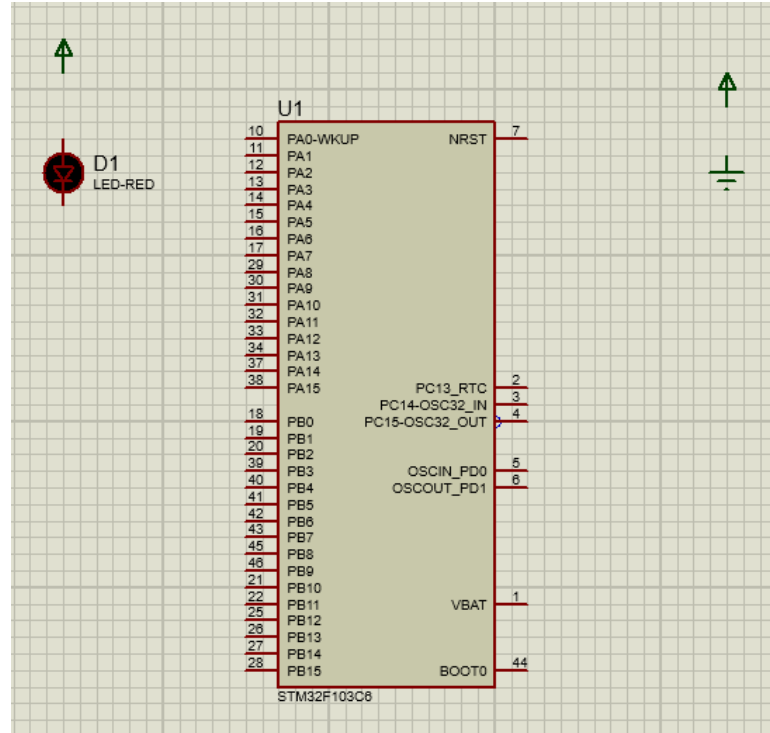


Figure 17: Place components to the project

Step 8: Start wiring the circuit. The negative pin of the LED is connected to PA5 while its positive pin is connected to the power supply. For the power and the ground on the right, just make a short wire, which will be labeled in the next step.

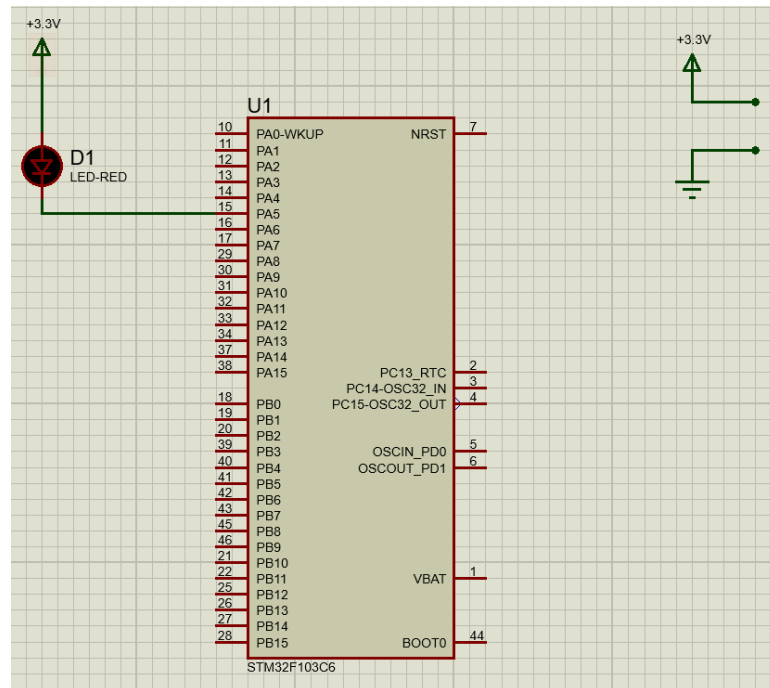


Figure 18: Connect components and set the power to 3.3V

In this step, also double click on the power supply in order to provide the String property to **+3.3V**.

Step 9: Right click on the wire of the power supply and the ground, and select **Place wire Label**

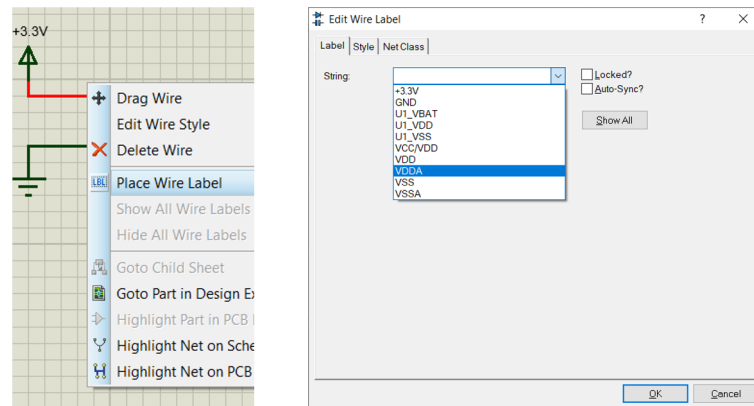


Figure 19: Place label for Power and Ground

This step is required as VDDA and VSSA of the STM32 must be connected to provide the reference voltage. Therefore, VDDA is connected to 3.3V, while the VSSA is connected to the Ground. Finally, the image of our schematic is shown below:

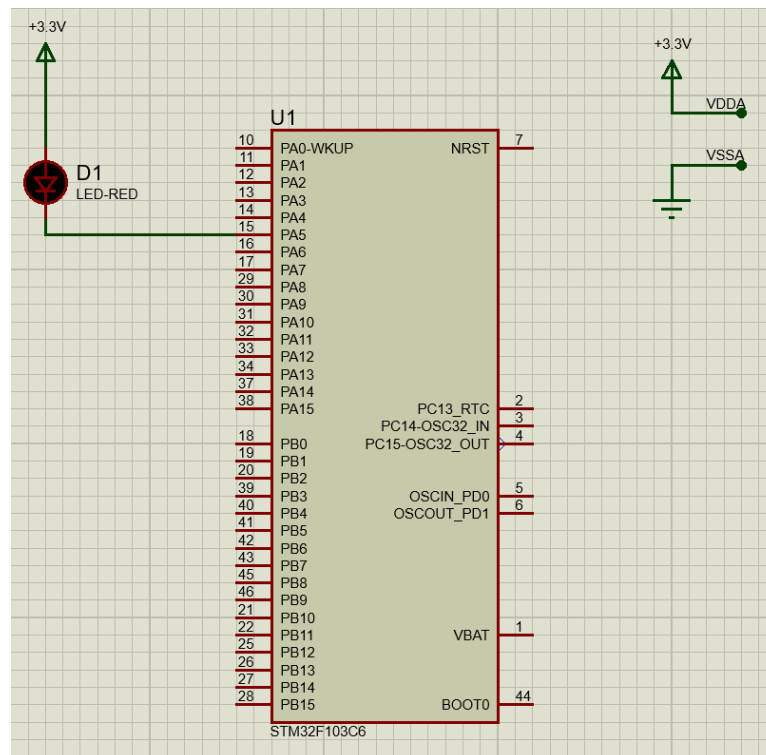


Figure 20: Finalize the schematic

Step 10: Double click on the STM32, and set the **Program File** to the Hex file, which is generated from Cube IDE, as following:

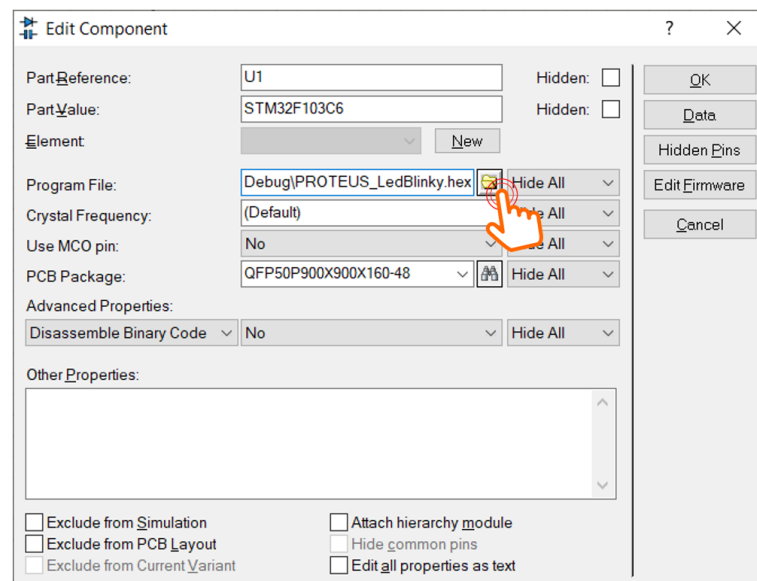


Figure 21: Set the program of the STM32 to the hex file from Cube IDE

From now, the simulation is ready to start by clicking on the menu **Debug**, and select on **Run simulation**. To stop the simulation, click on **Debug** and select **Stop VMS Debugging**. Moreover, there are some quick access bottom on the left corner of the Proteus to start or stop the simulation, as shown following:

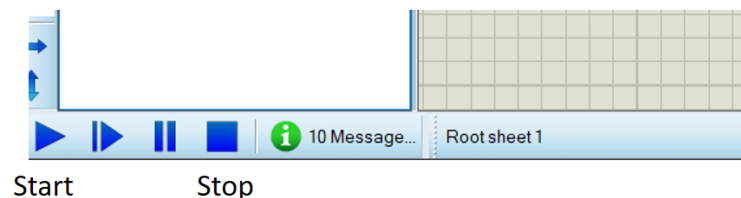


Figure 22: Quick access buttons to start and stop the simulation

If everything is success, students can see the LED is blinking every second. Please stop the simulation before updating the project, either in Proteus or STM32Cube IDE. However, the step 9 (set the program file for STM32 in Proteus) is required to do once. Beside the toggle instruction, student can set or reset a pin as following:

```
1 while (1){
2     HAL_GPIO_WritePin(LED_RED_GPIO_Port , LED_RED_Pin , GPIO_PIN_SET);
3     HAL_Delay(1000);
4     HAL_GPIO_WritePin(LED_RED_GPIO_Port , LED_RED_Pin , GPIO_PIN_RESET);
5     HAL_Delay(1000);
6 }
```

4 Exercise and Report

The practice source code is stored at: [Link GitHub Repository – STM32 Lab 1](#).

4.1 Exercise 1

From the simulation on Proteus, one more LED is connected to pin **PA6** of the STM32 (negative pin of the LED is connected to PA6). The component suggested in this exercise is **LED-YELLOW**, which can be found from the device list.

In this exercise, the status of two LEDs are switched every 2 seconds, as demonstrated in the figure bellow.

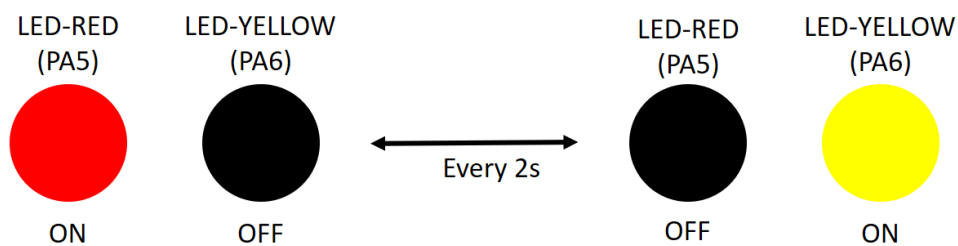


Figure 23: State transitions for 2 LEDs

Report 1: Depict the schematic from Proteus simulation in this report. The caption of the figure is a downloadable link to the Proteus project file (e.g. a github link).

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: Toggle LED](#).

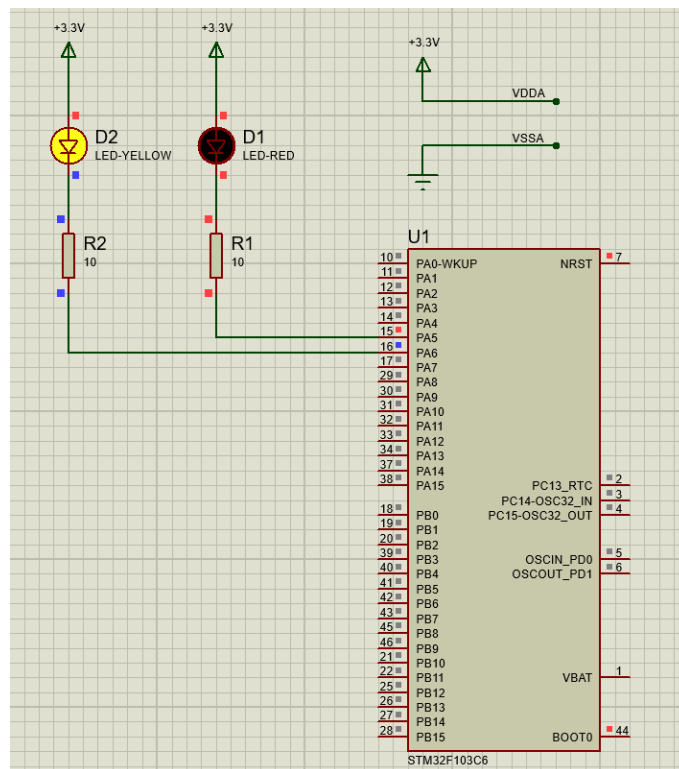


Figure 24: Proteus Ex1 ToggleLED

Report 2: Present the source code in the infinite loop while of your project. If a user-defined functions is used, it is required to present in this part. A brief description can be added for this function (e.g. using comments). A template to present your source code is presented below.

CASE 1:

```
1  while (1)
2      {
3          /* USER CODE END WHILE */
4          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
5          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
6          HAL_Delay(2000);
7          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
8          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
9          HAL_Delay(2000);
10         /* USER CODE BEGIN 3 */
11     }
```

CASE 2:

```
1  while (1)
2      {
3          /* USER CODE END WHILE */
4          HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
5          HAL_Delay(2000);
6          HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
7          /* USER CODE BEGIN 3 */
8      }
```

4.2 Exercise 2

Extend the first exercise to simulate the behavior of a traffic light. A third LED, named **LED-GREEN** is added to the system, which is connected to **PA7**. A cycle in this traffic light is 5 seconds for the RED, 2 seconds for the YELLOW and 3 seconds for the GREEN. The LED-GREEN is also controlled by its negative pin.

Similarly, the report in this exercise includes the schematic of your circuit and a your source code in the while loop.

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: TrafficLight](#).

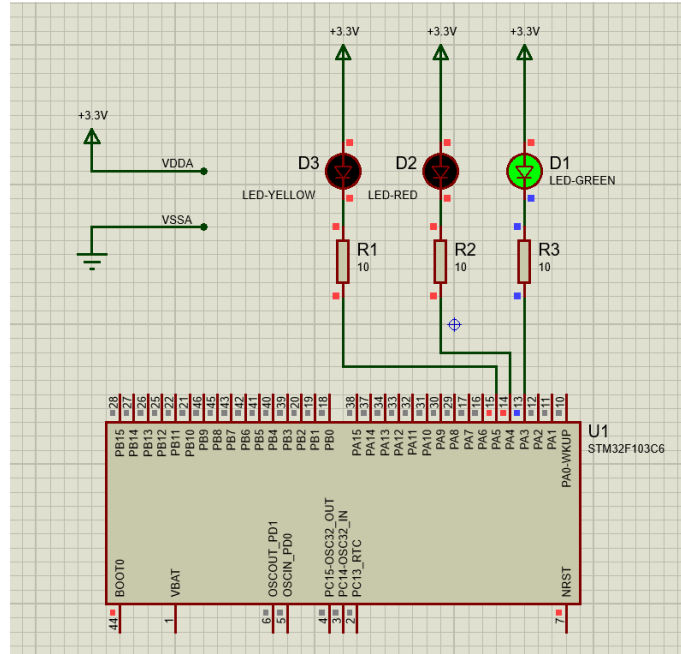


Figure 25: Proteus Ex2 TrafficLight

Report 2: Present the source code in while.

```

1  while (1)
2  {
3      // RED ON in 5s, others OFF
4      HAL_GPIO_WritePin(GPIOA, LED_RED_Pin, GPIO_PIN_RESET);
5      HAL_GPIO_WritePin(GPIOA, LED_YELLOW_Pin, GPIO_PIN_SET);
6      HAL_GPIO_WritePin(GPIOA, LED_GREEN_Pin, GPIO_PIN_SET);
7      HAL_Delay(5000);
8
9      // YELLOW ON in 2s, others OFF
10     HAL_GPIO_WritePin(GPIOA, LED_RED_Pin, GPIO_PIN_SET);
11     HAL_GPIO_WritePin(GPIOA, LED_YELLOW_Pin, GPIO_PIN_RESET);
12     HAL_GPIO_WritePin(GPIOA, LED_GREEN_Pin, GPIO_PIN_SET);
13     HAL_Delay(2000);
14
15     // GREEN ON in 3s, others OFF
16     HAL_GPIO_WritePin(GPIOA, LED_RED_Pin, GPIO_PIN_SET);
17     HAL_GPIO_WritePin(GPIOA, LED_YELLOW_Pin, GPIO_PIN_SET);
18     HAL_GPIO_WritePin(GPIOA, LED_GREEN_Pin, GPIO_PIN_RESET);
19     HAL_Delay(3000);
20 }

```

4.3 Exercise 3

Extend to the 4-way traffic light. Arrange 12 LEDs in a nice shape to simulate the behaviors of a traffic light. A reference design can be found in the figure bellow.

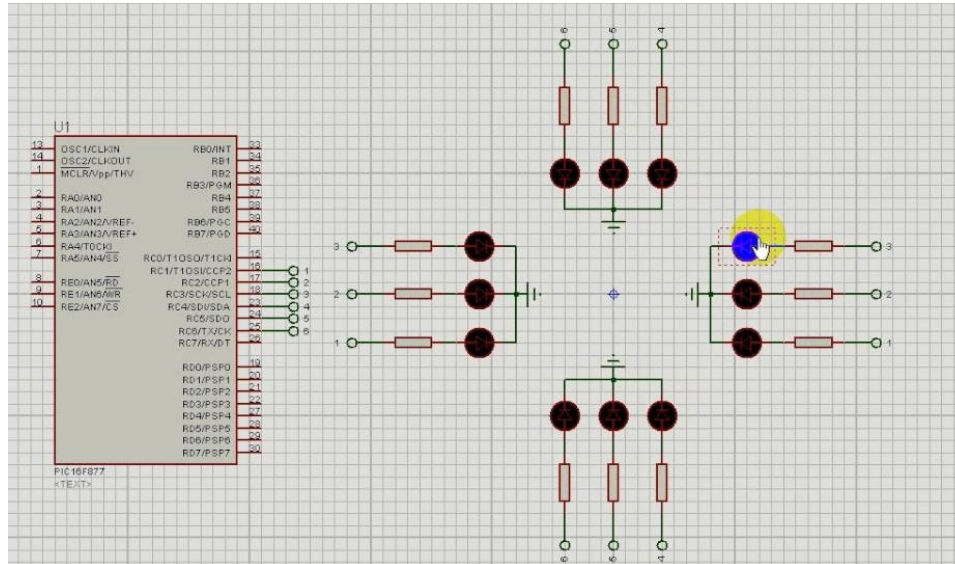


Figure 26: Reference design for a 4 way traffic light

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: 4WayTrafficLight](#).

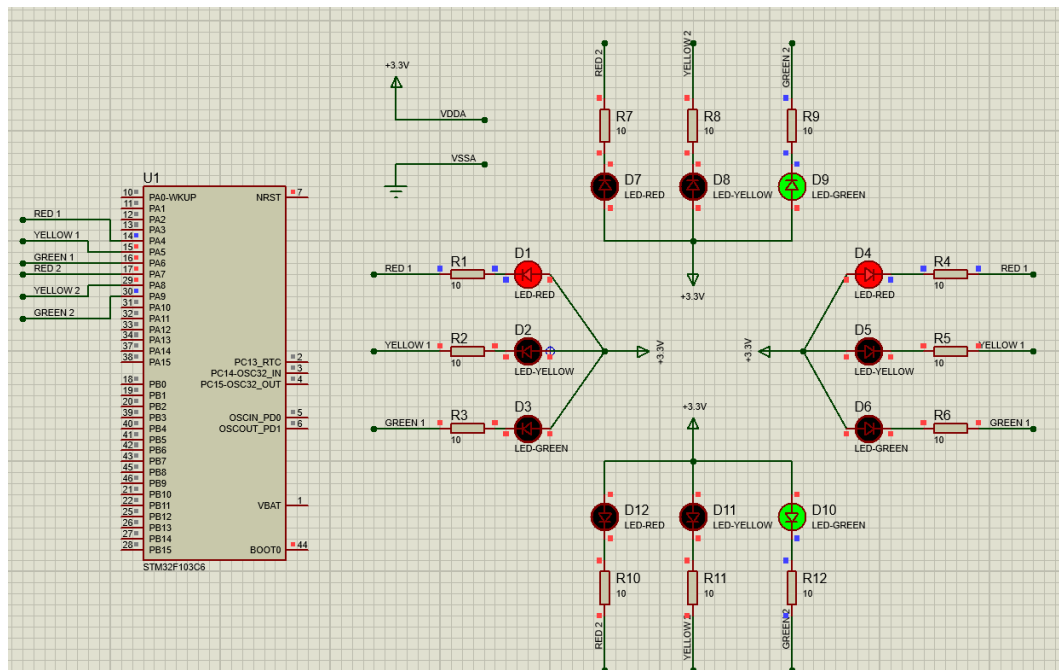


Figure 27: Proteus Ex3 Four Way TrafficLight

Report 2: Present the source code in while.

```
1 while (1)
2 {
```

```
3 // LED RED 1 ON, LED GREEN 2 ON AND OTHERS OFF
4 HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, RESET);
5 HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, SET);
6 HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, SET);
7 HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, SET);
8 HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, SET);
9 HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, RESET);
10 HAL_Delay(3000);
11
12 // LED RED 1 ON, LED YELLOW 2 ON, OTHERS LIGHT OFF
13 HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, RESET);
14 HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, SET);
15 HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, SET);
16 HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, SET);
17 HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, RESET);
18 HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, SET);
19 HAL_Delay(2000);
20
21 // LED GREEN 1 ON, LED RED 2 ON, OTHERS LIGHT OFF
22 HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, SET);
23 HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, SET);
24 HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, RESET);
25 HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, RESET);
26 HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, SET);
27 HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, SET);
28 HAL_Delay(3000);
29
30 // LED YELLOW 1 ON, LED RED 2 ON, OTHERS LIGHT OFF
31 HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, SET);
32 HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, SET);
33 HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, RESET);
34 HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, RESET);
35 HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, SET);
36 HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, SET);
37 HAL_Delay(2000);
38 }
```

4.4 Exercise 4

Add **only one 7 led segment** to the schematic in Exercise 3. This component can be found in Proteus by the keyword **7SEG-COM-ANODE**. For this device, the common pin should be connected to the power supply and other pins are supposed to be connected to PB0 to PB6. Therefore, to turn-on a segment in this 7SEG, the STM32 pin should be in logic 0 (0V). Implement a function named **display7SEG(int num)**. The input for this function is from 0 to 9 and the outputs are listed as following:

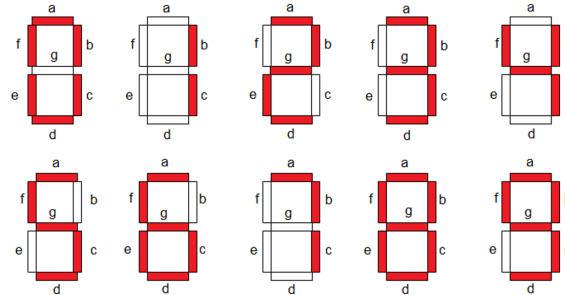


Figure 28: Display a number on 7 segment LED

This function is invoked in the while loop for testing as following:

```
1 int counter = 0;
2 while (1){
3     if(counter >= 10) counter = 0;
4     display7SEG(counter++);
5     HAL_Delay(1000);
6 }
```

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: 7SEG Display7SEG](#).

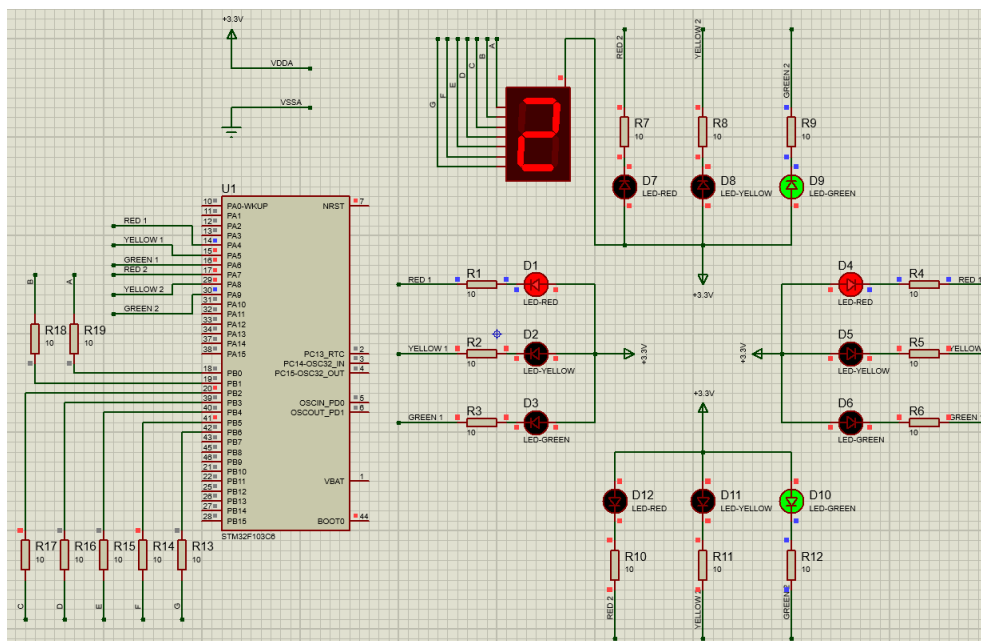


Figure 29: Proteus Ex4 7SEG Display7SEG

Report 2: Present the source code for display7SEG function.

```
1 void display7SEG(int num)
2 {
3     //Turn off all pins first
4     HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
5         LED_D_Pin | LED_E_Pin | LED_F_Pin | LED_G_Pin, SET);
6
7     switch (num)
8     {
9         case 0: // 0s
10            HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
11                LED_D_Pin | LED_E_Pin | LED_F_Pin, RESET);
12            break;
13         case 1: // 1s
14            HAL_GPIO_WritePin(GPIOB, LED_B_Pin | LED_C_Pin, RESET);
15            break;
16         case 2: // 2s
17            HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_D_Pin |
18                LED_E_Pin | LED_G_Pin, RESET);
19            break;
20         case 3: // 3s
21            HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
22                LED_D_Pin | LED_G_Pin, RESET);
23            break;
24         case 4: // 4s
25            HAL_GPIO_WritePin(GPIOB, LED_B_Pin | LED_C_Pin | LED_F_Pin |
26                LED_G_Pin, RESET);
27            break;
28         case 5: // 5s
29            HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_C_Pin | LED_D_Pin |
30                LED_F_Pin | LED_G_Pin, RESET);
31            break;
32         case 6: // 6s
33            HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_C_Pin | LED_D_Pin |
34                LED_E_Pin | LED_F_Pin | LED_G_Pin, RESET);
35            break;
```

```
29  case 7: // 7s
30      HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin,
31                          RESET);
31      break;
32  case 8: // 8s
33      HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
34                          LED_D_Pin | LED_E_Pin | LED_F_Pin | LED_G_Pin, RESET);
34      break;
35  case 9: // 9s
36      HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
37                          LED_D_Pin | LED_F_Pin | LED_G_Pin, RESET);
37      break;
38  default:
39      HAL_GPIO_WritePin(GPIOB, LED_G_Pin, RESET);
40      break;
41  }
42 }
```

4.5 Exercise 5

Integrate the 7SEG-LED to the 4 way traffic light. In this case, the 7SEG-LED is used to display countdown value.

In this exercise, only source code is required to present. The function `display7SEG` in previous exercise can be re-used.

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: Traffic with 7SEG](#).

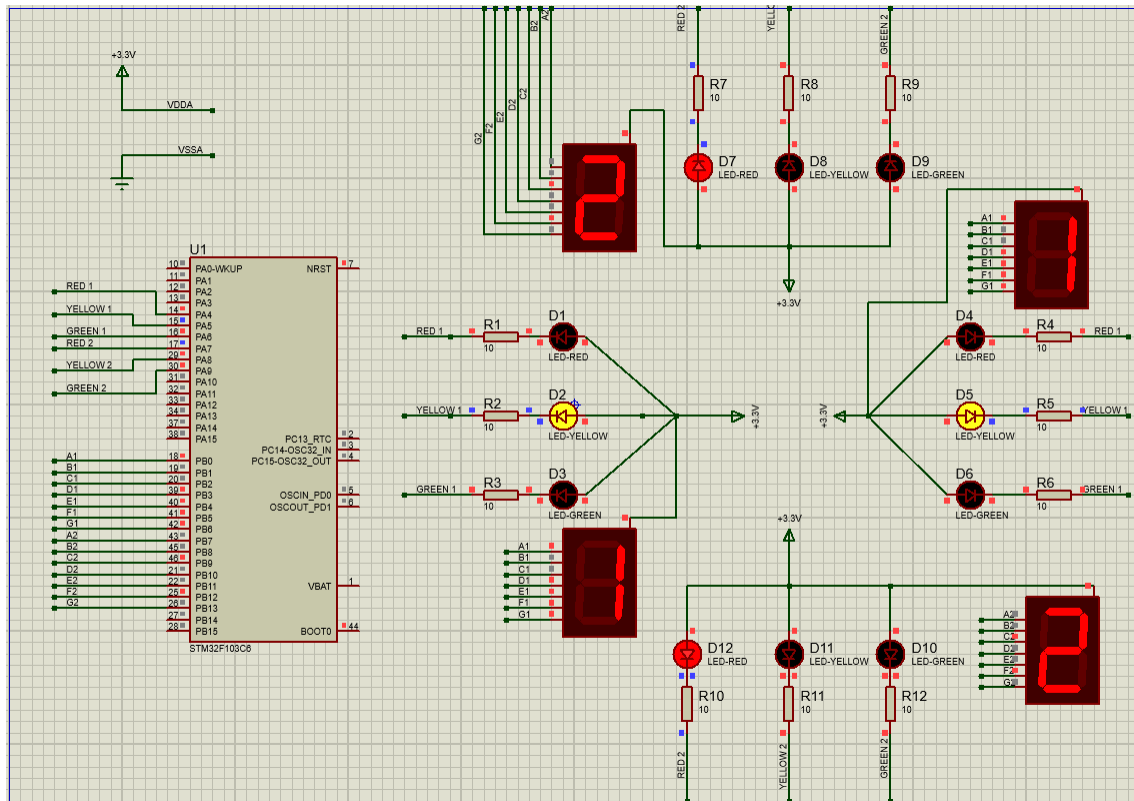


Figure 30: Proteus Ex5 Traffic with 7SEG

Report 2: Present the source code for display7SEG function.

```
1 // Define traffic light states
2 enum State
3 {
4     red,
5     green,
6     yellow
7 };
8 // Countdown timer variables for each intersection
9 int numDisplay1 = 4; // Countdown for intersection 1 (Start with red
    light: 4s)
10 int numDisplay2 = 2; // Countdown for intersection 2 (Start with
    green light: 3s)
11
12 // Current state of each intersection
13 // Intersection 1 starts with red
14 enum State currentStateVar1 = red;
15 // Intersection 2 starts with green
16 enum State currentStateVar2 = green;
```



```
17 /* Private function prototypes
    -----*/
18 void SystemClock_Config(void);
19 static void MX_GPIO_Init(void);
20 /* Private user code
    -----*/
21 /* USER CODE BEGIN 0 */
22 // Turn on the traffic light according to state
23 void setTrafficLight(GPIO_TypeDef *RED_Port, uint16_t RED_Pin,
24                      GPIO_TypeDef *YELLOW_Port, uint16_t YELLOW_Pin,
25                      GPIO_TypeDef *GREEN_Port, uint16_t GREEN_Pin,
26                      enum State state)
27 {
28     if (state == red)
29     {
30         HAL_GPIO_WritePin(RED_Port, RED_Pin, RESET);    // ON red
31         HAL_GPIO_WritePin(YELLOW_Port, YELLOW_Pin, SET); // OFF yellow
32         HAL_GPIO_WritePin(GREEN_Port, GREEN_Pin, SET);   // OFF green
33     }
34     else if (state == green)
35     {
36         HAL_GPIO_WritePin(RED_Port, RED_Pin, SET);      // OFF red
37         HAL_GPIO_WritePin(YELLOW_Port, YELLOW_Pin, SET); // OFF yellow
38         HAL_GPIO_WritePin(GREEN_Port, GREEN_Pin, RESET); // ON green
39     }
40     else if (state == yellow)
41     {
42         HAL_GPIO_WritePin(RED_Port, RED_Pin, SET);      // OFF red
43         HAL_GPIO_WritePin(YELLOW_Port, YELLOW_Pin, RESET); // ON yellow
44         HAL_GPIO_WritePin(GREEN_Port, GREEN_Pin, SET);   // OFF green
45     }
46 }
47
48 /**
49  * Display numbers on a 7-segment LED
50  * 7-segment LED layout:
```

```
51 *      A
52 *      -----
53 *      |      |
54 *  F|    G    |B
55 *  |-----|
56 *      |      |
57 *  E|          |C
58 *      -----
59 *      D
60 */
61 void display7SEG(int num, uint16_t A, uint16_t B, uint16_t C,
62     uint16_t D, uint16_t E, uint16_t F, uint16_t G)
63 {
64     if (num == 0)
65     {
66         // Display 0: turn on A,B,C,D,E,F - turn off G
67         HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
68         HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
69         HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
70         HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
71         HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
72         HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
73         HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_SET);   // g - OFF
74     }
75     else if (num == 1)
76     {
77         // Display 1: turn on B,C - turn off A,D,E,F,G
78         HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_SET);   // a - OFF
79         HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
80         HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
81         HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_SET);   // d - OFF
82         HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
83         HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);   // f - OFF
84         HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_SET);   // g - OFF
85     }
86     else if (num == 2)
```

```
86 {
87     // Display 2: turn on A,B,D,E,G - turn off C,F
88     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
89     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
90     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_SET);    // c - OFF
91     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
92     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
93     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);    // f - OFF
94     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
95 }
96 else if (num == 3)
97 {
98     // Display 3: turn on A,B,C,D,G - turn off E,F
99     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
100    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
101    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
102    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
103    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);    // e - OFF
104    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);    // f - OFF
105    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
106 }
107 else if (num == 4)
108 {
109     // Display 4: turn on B,C,F,G - turn off A,D,E
110     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_SET);    // a - OFF
111     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
112     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
113     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_SET);    // d - OFF
114     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);    // e - OFF
115     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
116     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
117 }
118 else if (num == 5)
119 {
120     // Display 5: turn on A,C,D,F,G - turn off B,E
121     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
```

```
122     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_SET);    // b - OFF
123     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET);  // c - ON
124     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET);  // d - ON
125     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);    // e - OFF
126     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET);  // f - ON
127     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET);  // g - ON
128 }
129 else if (num == 6)
130 {
131     // Display 6: turn on A,C,D,E,F,G - turn off B
132     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
133     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_SET);    // b - OFF
134     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET);  // c - ON
135     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET);  // d - ON
136     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET);  // e - ON
137     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET);  // f - ON
138     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET);  // g - ON
139 }
140 else if (num == 7)
141 {
142     // Display 7: turn on A,B,C - turn off D,E,F,G
143     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
144     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
145     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
146     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_SET);    // d - OFF
147     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);    // e - OFF
148     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);    // f - OFF
149     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_SET);    // g - OFF
150 }
151 else if (num == 8)
152 {
153     // Display 8: turn on all A,B,C,D,E,F,G
154     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
155     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
156     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
157     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
```

```
158     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
159     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
160     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
161 }
162 else if (num == 9)
163 {
164     // Display 9: turn on A,B,C,D,F,G - turn off E
165     HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
166     HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
167     HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
168     HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
169     HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET); // e - OFF
170     HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
171     HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
172 }
173 }
174
175 int main(void)
176 {
177     HAL_Init();
178     SystemClock_Config();
179     MX_GPIO_Init();
180
181     while (1)
182     {
183         //-----DISPLAY NUMBERS ON 7-SEGMENT-----//
184         // Show countdown on 7-segment LED of intersection 1
185         display7SEG(numDisplay1, LED_A1_Pin, LED_B1_Pin, LED_C1_Pin,
186                     LED_D1_Pin, LED_E1_Pin, LED_F1_Pin, LED_G1_Pin);
187         // Show countdown on 7-segment LED of intersection 2
188         display7SEG(numDisplay2, LED_A2_Pin, LED_B2_Pin, LED_C2_Pin,
189                     LED_D2_Pin, LED_E2_Pin, LED_F2_Pin, LED_G2_Pin);
190
191         // Turn on lights according to the current state
192         setTrafficLight(LED_RED1_GPIO_Port, LED_RED1_Pin,
193                         LED_YELLOW1_GPIO_Port, LED_YELLOW1_Pin,
```

```
192         LED_GREEN1_GPIO_Port, LED_GREEN1_Pin ,
193         currentStateVar1);
194
195     setTrafficLight(LED_RED2_GPIO_Port, LED_RED2_Pin ,
196                    LED_YELLOW2_GPIO_Port, LED_YELLOW2_Pin ,
197                    LED_GREEN2_GPIO_Port, LED_GREEN2_Pin ,
198                    currentStateVar2);
199     HAL_Delay(1000); // Delay 1s
200
201     //-----COUNTDOWN TIMER-----//
202     // Decrease countdown for both intersections
203     --numDisplay1;
204     --numDisplay2;
205
206     // CHANGE STATE OF INTERSECTION 1 (EAST-WEST)
207     // If red time ends -> switch to green (3s)
208     if (numDisplay1 < 0)
209     {
210         if (currentStateVar1 == red)
211         {
212             currentStateVar1 = green;
213             numDisplay1 = 2; // Green 3s
214         }
215         // If green time ends -> switch to yellow (2s)
216         else if (currentStateVar1 == green)
217         {
218             currentStateVar1 = yellow;
219             numDisplay1 = 1; // Yellow 2s
220         }
221         // If yellow time ends -> switch to red (4s)
222         else
223         { // yellow -> red
224             currentStateVar1 = red;
225             numDisplay1 = 4; // Red 4s
226         }
227     }
```

```
228
229
230 // CHANGE STATE OF INTERSECTION 2 (NORTH-SOUTH)
231 // If red time ends -> switch to green (3s)
232 if (numDisplay2 < 0)
233 {
234     if (currentStateVar2 == red)
235     {
236         currentStateVar2 = green;
237         numDisplay2 = 2; // Green 3s
238     }
239     // If green time ends -> switch to yellow (2s)
240     else if (currentStateVar2 == green)
241     {
242         currentStateVar2 = yellow;
243         numDisplay2 = 1; // Yellow 2s
244     }
245     // If yellow time ends -> switch to red (4s)
246     else
247     { // yellow -> red
248         currentStateVar2 = red;
249         numDisplay2 = 4; // Red 4s
250     }
251 }
252 }
253 }
```

4.6 Exercise 6

In this exercise, a new Proteus schematic is designed to simulate an analog clock, with 12 different number. The connections for 12 LEDs are supposed from PA4 to PA15 of the STM32. The arrangement of 12 LEDs is depicted as follows.

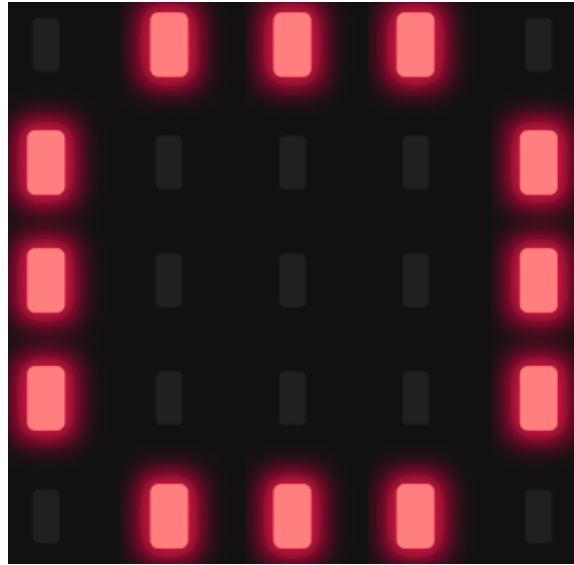


Figure 31: 12 LEDs for an analog clock

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: AnalogClock](#).

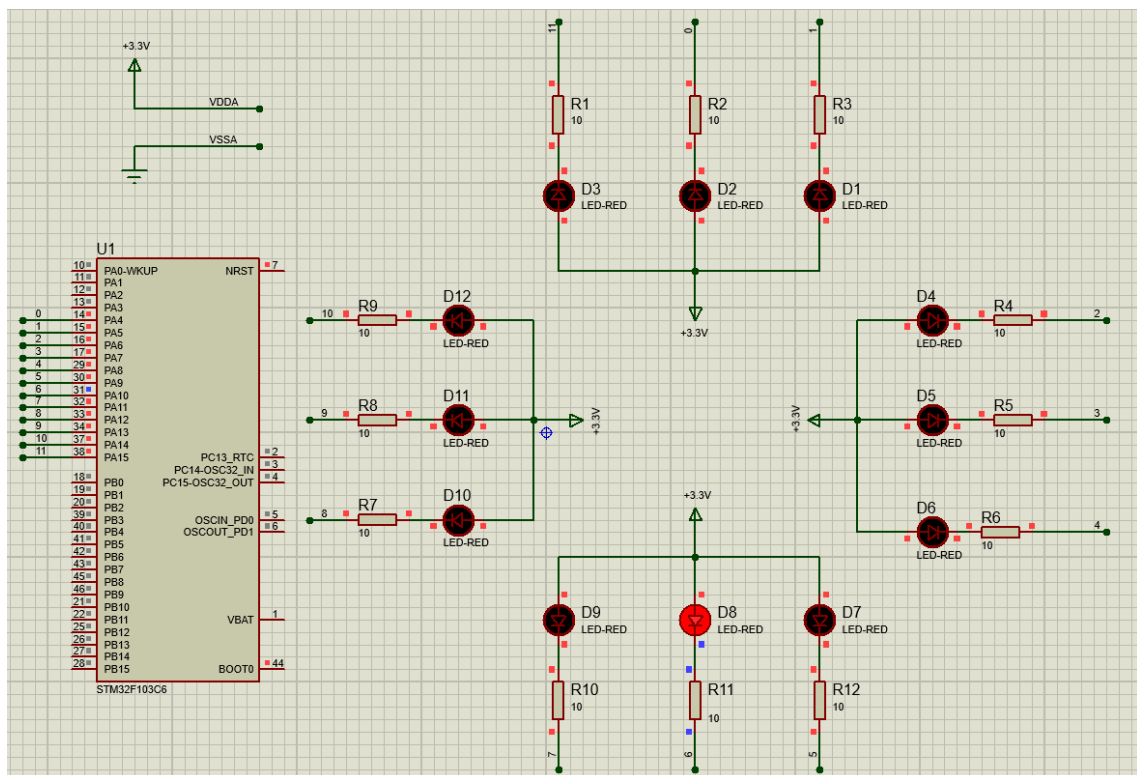


Figure 32: Proteus Ex6 AnalogClock 12LEDs

Report 2: Implement a simple program to test the connection of every single LED. This testing program should turn every LED in a sequence.

```
1 // Array of LED pins from LED0 - LED11
2 uint16_t LED_Pins[12] = {
3     LED0_Pin, LED1_Pin, LED2_Pin, LED3_Pin,
4     LED4_Pin, LED5_Pin, LED6_Pin, LED7_Pin,
5     LED8_Pin, LED9_Pin, LED10_Pin, LED11_Pin};
6
7 // Function to turn off all LEDs
8 void TurnOffAllLEDs(void)
9 {
10     for (int i = 0; i < 12; i++)
11     {
12         HAL_GPIO_WritePin(GPIOA, LED_Pins[i], SET); // Turn OFF LED
13     }
14 }
15
16 // Function to turn on an LED by index
17 void TurnOnLED(int index)
18 {
19     HAL_GPIO_WritePin(GPIOA, LED_Pins[index], RESET); // Turn ON LED
20 }
21
22 int main(void)
23 {
24     HAL_Init();
25     SystemClock_Config();
26     MX_GPIO_Init();
27
28     TurnOffAllLEDs(); // Ensure all LEDs are off at startup
29     int currentLED = 0; // LED0 = position 12 on the clock
30
31     while (1)
32     {
33         // Turn on the current LED
34         HAL_GPIO_WritePin(GPIOA, LED_Pins[currentLED], GPIO_PIN_RESET);
```

```
35     HAL_Delay(200);
36
37     // Turn off the LED after delay
38     HAL_GPIO_WritePin(GPIOA, LED_Pins[currentLED], GPIO_PIN_SET);
39
40     // Move to the next LED
41     currentLED++;
42     if (currentLED >= 12)
43     {
44         currentLED = 0; // Wrap around to LED0 after 12 LEDs
45     }
46 }
47 }
```

4.7 Exercise 7

Implement a function named **clearAllClock()** to turn off all 12 LEDs. Present the source code of this function.

CASE 1:

```
1 // Function to turn off all LEDs
2 void clearAllClock() {
3     HAL_GPIO_WritePin(GPIOA, LED0_Pin, SET);
4     HAL_GPIO_WritePin(GPIOA, LED1_Pin, SET);
5     HAL_GPIO_WritePin(GPIOA, LED2_Pin, SET);
6     HAL_GPIO_WritePin(GPIOA, LED3_Pin, SET);
7     HAL_GPIO_WritePin(GPIOA, LED4_Pin, SET);
8     HAL_GPIO_WritePin(GPIOA, LED5_Pin, SET);
9     HAL_GPIO_WritePin(GPIOA, LED6_Pin, SET);
10    HAL_GPIO_WritePin(GPIOA, LED7_Pin, SET);
11    HAL_GPIO_WritePin(GPIOA, LED8_Pin, SET);
12    HAL_GPIO_WritePin(GPIOA, LED9_Pin, SET);
13    HAL_GPIO_WritePin(GPIOA, LED10_Pin, SET);
14    HAL_GPIO_WritePin(GPIOA, LED11_Pin, SET);
15 }
```

CASE 2:

```
1 // Array of LED pins from LED0 - LED11
2 // Each LED corresponds to one position on the clock face (0 - 11)
3 uint16_t LED_Pins[12] = {
4     LED0_Pin, LED1_Pin, LED2_Pin, LED3_Pin,
5     LED4_Pin, LED5_Pin, LED6_Pin, LED7_Pin,
6     LED8_Pin, LED9_Pin, LED10_Pin, LED11_Pin};
7 // Function to turn off all LEDs (clear the entire clock face)
8 void clearAllClock()
9 {
10     for (int i = 0; i < 12; i++)
11     {
12         // Turn OFF LED: SET = high level - LED OFF (active-low)
13         HAL_GPIO_WritePin(GPIOA, LED_Pins[i], SET);
14     }
15 }
```

4.8 Exercise 8

Implement a function named **setNumberOnClock(int num)**. The input for this function is from **0** to **11** and an appropriate LED is turn on. Present the source code of this function.

CASE 1:

```
1 void setNumberOnClock(int index) {
2     switch(index) {
3         case 0: HAL_GPIO_WritePin(GPIOA, LED0_Pin, RESET); break;
4         case 1: HAL_GPIO_WritePin(GPIOA, LED1_Pin, RESET); break;
5         case 2: HAL_GPIO_WritePin(GPIOA, LED2_Pin, RESET); break;
6         case 3: HAL_GPIO_WritePin(GPIOA, LED3_Pin, RESET); break;
7         case 4: HAL_GPIO_WritePin(GPIOA, LED4_Pin, RESET); break;
8         case 5: HAL_GPIO_WritePin(GPIOA, LED5_Pin, RESET); break;
9         case 6: HAL_GPIO_WritePin(GPIOA, LED6_Pin, RESET); break;
10        case 7: HAL_GPIO_WritePin(GPIOA, LED7_Pin, RESET); break;
11        case 8: HAL_GPIO_WritePin(GPIOA, LED8_Pin, RESET); break;
12        case 9: HAL_GPIO_WritePin(GPIOA, LED9_Pin, RESET); break;
13        case 10: HAL_GPIO_WritePin(GPIOA, LED10_Pin, RESET); break;
14        case 11: HAL_GPIO_WritePin(GPIOA, LED11_Pin, RESET); break;
15        default: break;    } }
```

CASE 2:

```
1 // Array of LED pins from LED0 - LED11
2 // Each LED corresponds to one position on the clock face (0 - 11)
3 uint16_t LED_Pins[12] = {
4     LED0_Pin, LED1_Pin, LED2_Pin, LED3_Pin,
5     LED4_Pin, LED5_Pin, LED6_Pin, LED7_Pin,
6     LED8_Pin, LED9_Pin, LED10_Pin, LED11_Pin};
7
8 // Turn on the LED at the given clock position
9 void setNumberOnClock(int index)
10 {
11     HAL_GPIO_WritePin(GPIOA, LED_Pins[index], RESET);
12 }
```

4.9 Exercise 9

Implement a function named **clearNumberOnClock(int num)**. The input for this function is from **0 to 11** and an appropriate LED is turn off.

CASE 1:

```
1 void clearNumberOnClock(int index) {
2     switch(index) {
3         case 0: HAL_GPIO_WritePin(GPIOA, LED0_Pin, SET); break;
4         case 1: HAL_GPIO_WritePin(GPIOA, LED1_Pin, SET); break;
5         case 2: HAL_GPIO_WritePin(GPIOA, LED2_Pin, SET); break;
6         case 3: HAL_GPIO_WritePin(GPIOA, LED3_Pin, SET); break;
7         case 4: HAL_GPIO_WritePin(GPIOA, LED4_Pin, SET); break;
8         case 5: HAL_GPIO_WritePin(GPIOA, LED5_Pin, SET); break;
9         case 6: HAL_GPIO_WritePin(GPIOA, LED6_Pin, SET); break;
10        case 7: HAL_GPIO_WritePin(GPIOA, LED7_Pin, SET); break;
11        case 8: HAL_GPIO_WritePin(GPIOA, LED8_Pin, SET); break;
12        case 9: HAL_GPIO_WritePin(GPIOA, LED9_Pin, SET); break;
13        case 10: HAL_GPIO_WritePin(GPIOA, LED10_Pin, SET); break;
14        case 11: HAL_GPIO_WritePin(GPIOA, LED11_Pin, SET); break;
15        default: break;
16    }
17 }
```

CASE 2:

```
1 // Array of LED pins from LED0 - LED11
2 // Each LED corresponds to one position on the clock face (0 - 11)
3 uint16_t LED_Pins[12] = {
4     LED0_Pin, LED1_Pin, LED2_Pin, LED3_Pin,
5     LED4_Pin, LED5_Pin, LED6_Pin, LED7_Pin,
6     LED8_Pin, LED9_Pin, LED10_Pin, LED11_Pin};
7 // Turn off the LED at the given clock position
8 void clearNumberOnClock(int index)
9 {
10     HAL_GPIO_WritePin(GPIOA, LED_Pins[index], SET);
11 }
```

4.10 Exercise 10

Integrate the whole system and use 12 LEDs to display a clock. At a given time, there are only 3 LEDs are turn on for hour, minute and second information.

The source code and Proteus simulation schematic are provided at [STM32 Lab 1: ClockFunctions](#).

```
1 /* Turn off all LEDs at startup */
2 clearAllClock();
3 // Initialize starting values for hour, minute, second
4 int clkHr = 1; // Hour starts at LED1
5 int clkMin = 3; // Minute starts at LED3
6 int clkSec = 0; // Second starts at LED0
7
8 // Turn on initial LEDs for hour, minute, second
9 setNumberOnClock(clkHr);
10 setNumberOnClock(clkMin);
11 setNumberOnClock(clkSec);
12
13 while (1)
14 {
15     HAL_Delay(50);
16
17     // ==== STEP 1: CLEAR OLD SECOND LED ====
18     // If second position does not overlap with hour & minute - turn
19     // off old second LED
```

```
19  if (clkSec != clkMin && clkSec != clkHr)
20  {
21      clearNumberOnClock(clkSec);
22  }
23  // ==== STEP 2: INCREMENT SECOND ====
24  clkSec++;
25  if (clkSec > 11)
26  { // End of one cycle (12 positions) - reset to 0
27      clkSec = 0;
28      // ==== STEP 3: CLEAR OLD MINUTE LED ====
29      if (clkMin != clkHr && clkMin != clkSec)
30      {
31          clearNumberOnClock(clkMin);
32      }
33      // ==== STEP 4: INCREMENT MINUTE ====
34      clkMin++;
35      if (clkMin > 11)
36      { // End of one cycle - reset to 0
37          clkMin = 0;
38          // ==== STEP 5: CLEAR OLD HOUR LED ====
39          if (clkHr != clkMin && clkHr != clkSec)
40          {
41              clearNumberOnClock(clkHr);
42          }
43          // ==== STEP 6: INCREMENT HOUR ====
44          clkHr++;
45          if (clkHr > 11)
46              clkHr = 0;          // Wrap around to 0 after 12 hours
47          setNumberOnClock(clkHr); // Turn on new hour LED
48      }
49      setNumberOnClock(clkMin);   // Turn on new minute LED
50  }
51
52  // ==== STEP 7: TURN ON NEW SECOND LED ====
53  setNumberOnClock(clkSec);
54 }
```



Tài liệu