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: 30/09/2025

Instructor: Mr. Ton Huynh Long

Student Name	Student ID
Phạm Công Võ	2313946

Ho Chi Minh City, September 2025

Table of Contents

List of Figures

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**

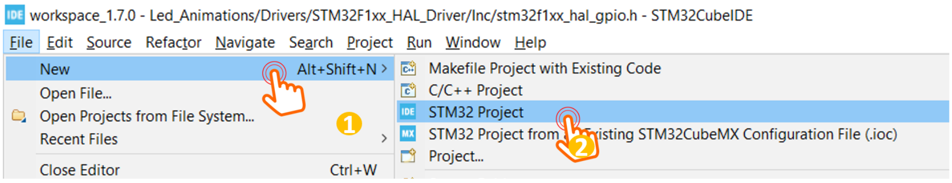

Figure 2: *Create a new project on STM32CubeIDE*

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**

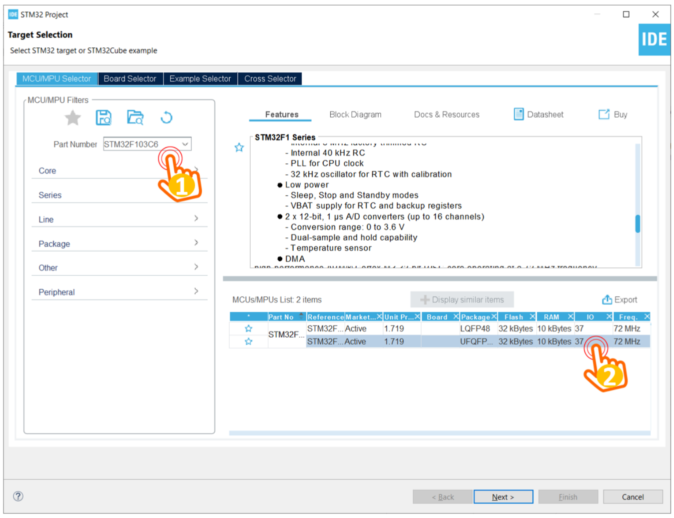

Figure 3: *Select the target device*

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

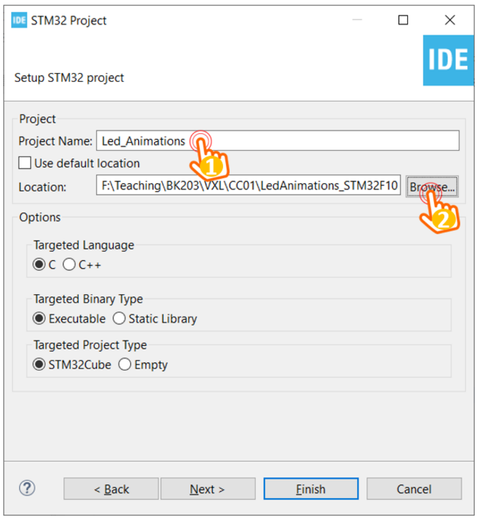

Figure 4: *Select the target device*

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.

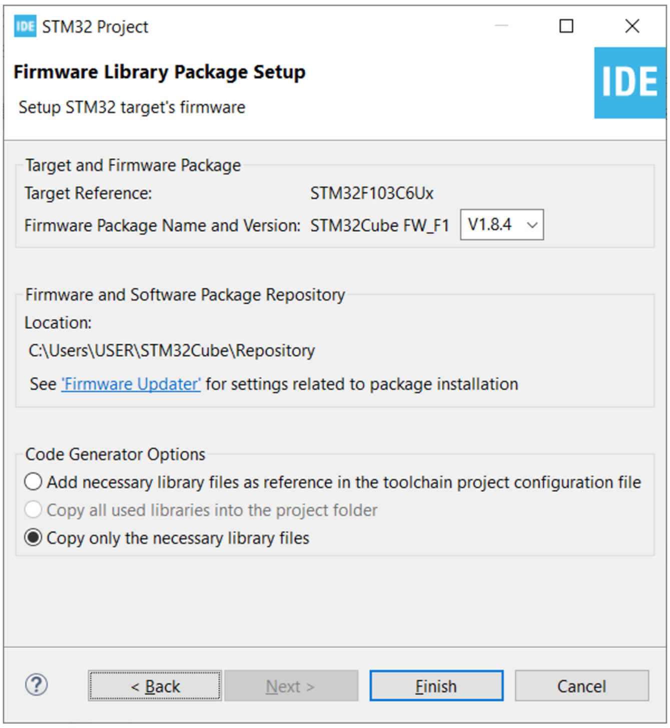

Figure 5: *Keep default firmware version*

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.

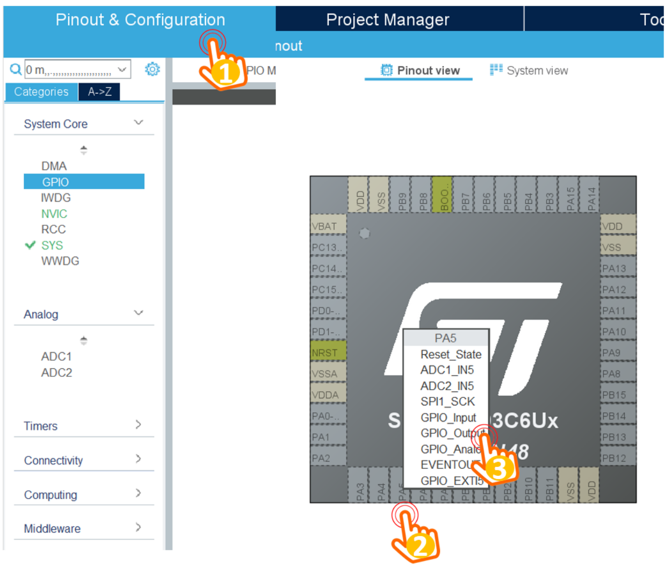

Figure 6: *Set PA5 to GPIO Output mode*

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 lable**, and provide the name for this pin (e.g. **LED_RED**). This step helps programming afterward more memorable.

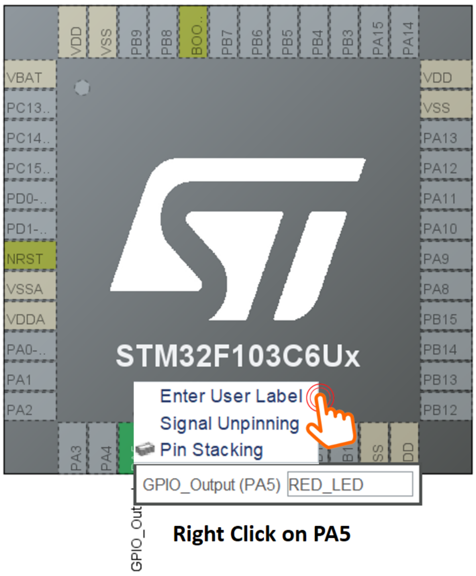

Figure 7: *Provide a name for PA5*

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:

```
int main(void)
{
  /* 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();
```

```
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 bellow:

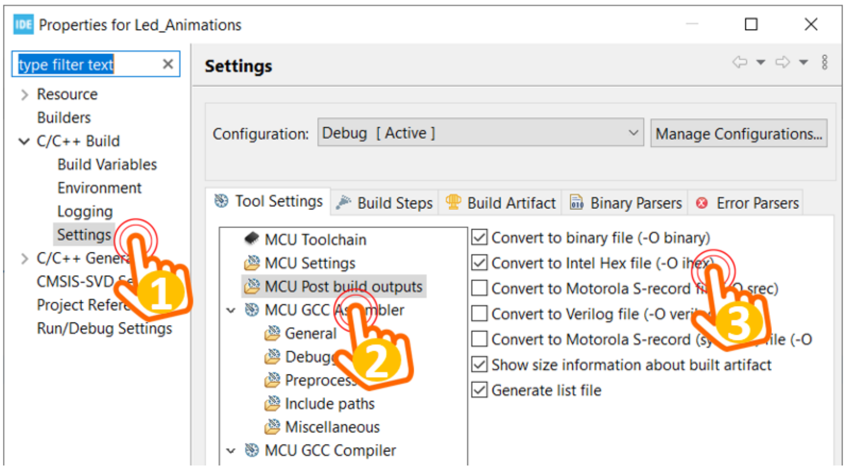

Figure 8: *Config for hex file output*

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**.

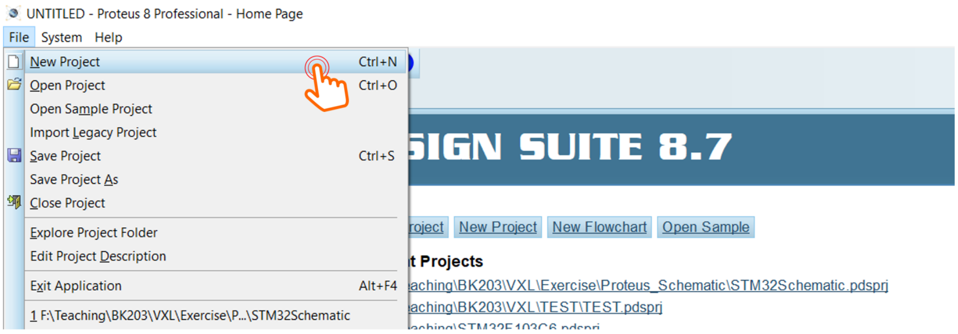

Figure 10: *Create a new project on Proteus*

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

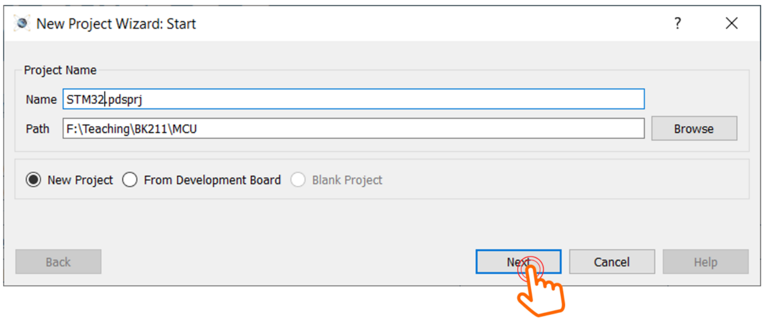

Figure 11: *Provide project name and location*

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

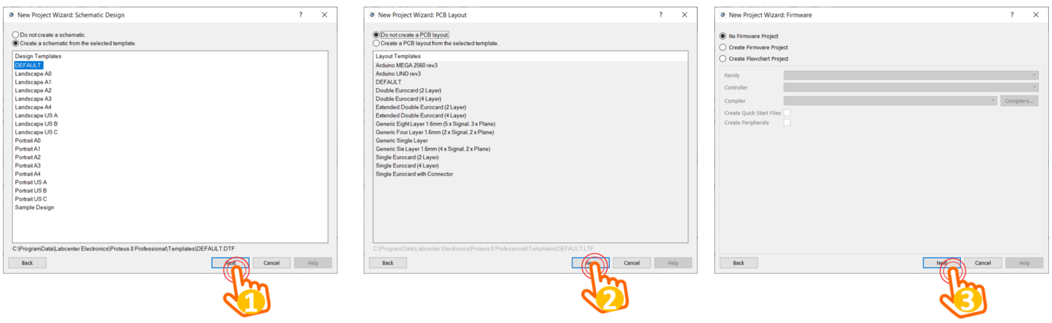

Figure 12: *Keep the default options by clicking on Next*

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

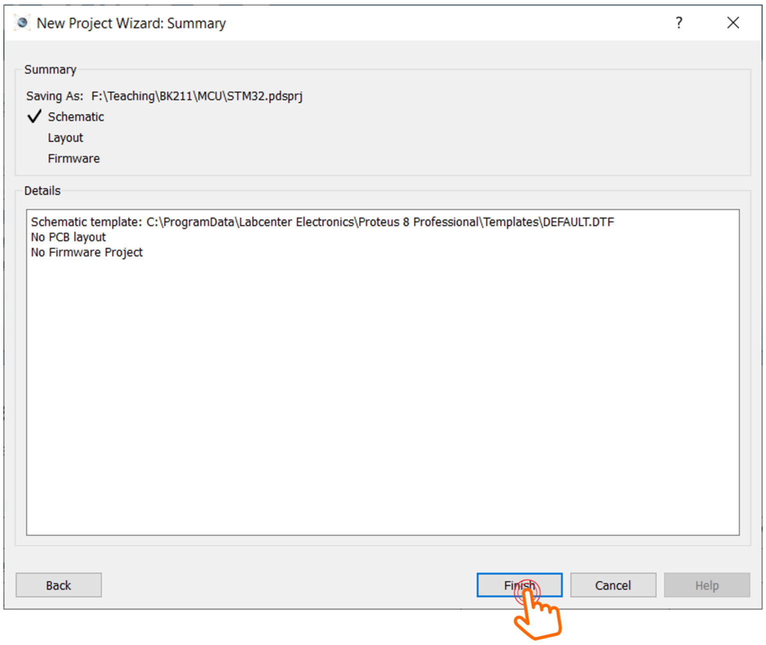

Figure 13: *Finish the project wizard*

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

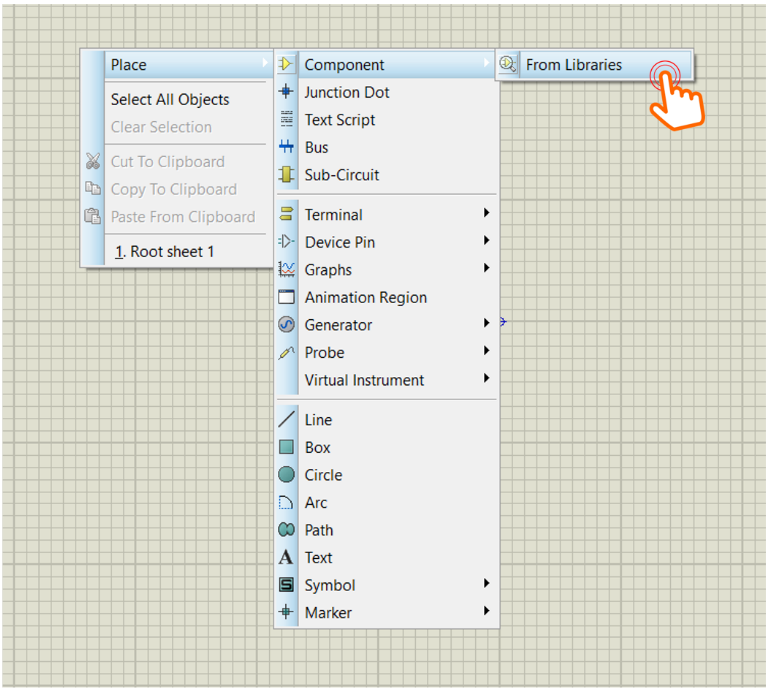

Figure 14: *Select a component from the library*

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:

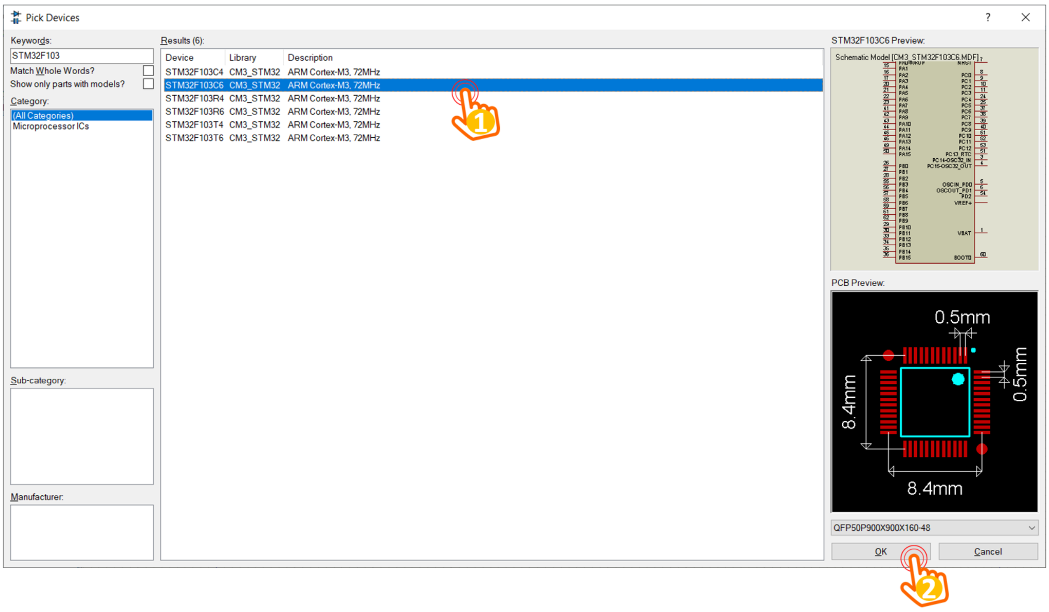

Figure 15: *Select STM32F103C6*

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:

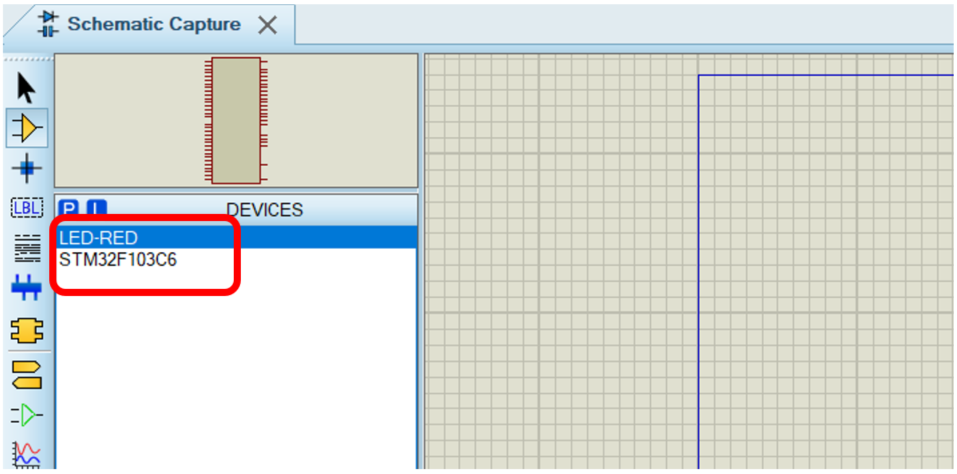

Figure 16: *STM32 and an LED in the project*

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:

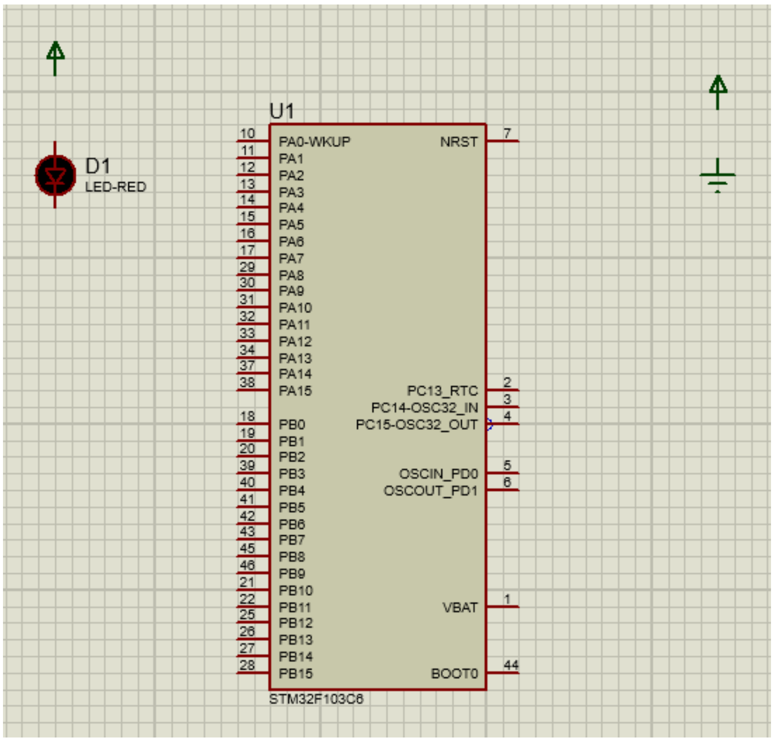

Figure 17: *Place components to the project*

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 labeled in the next step.

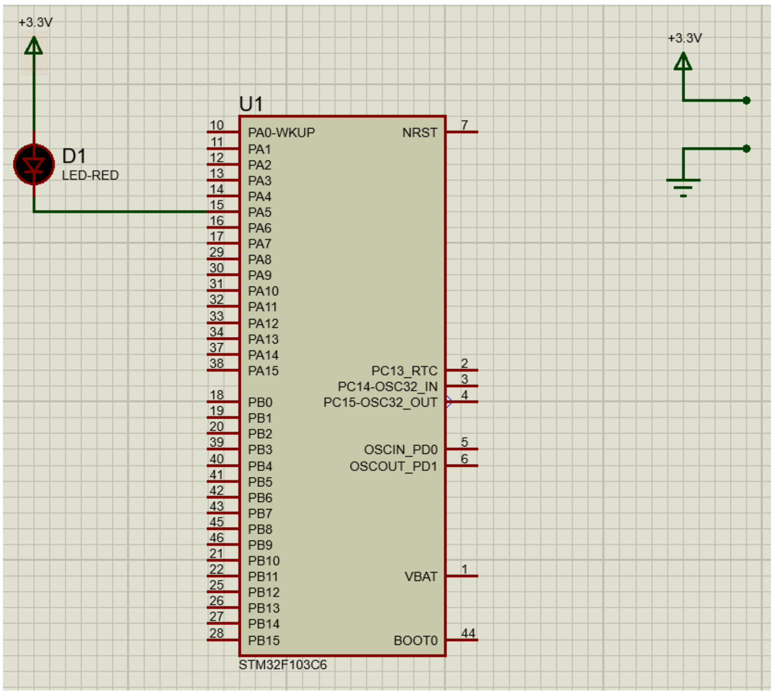

Figure 18: *Connect components and set the power to 3.3V*

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**

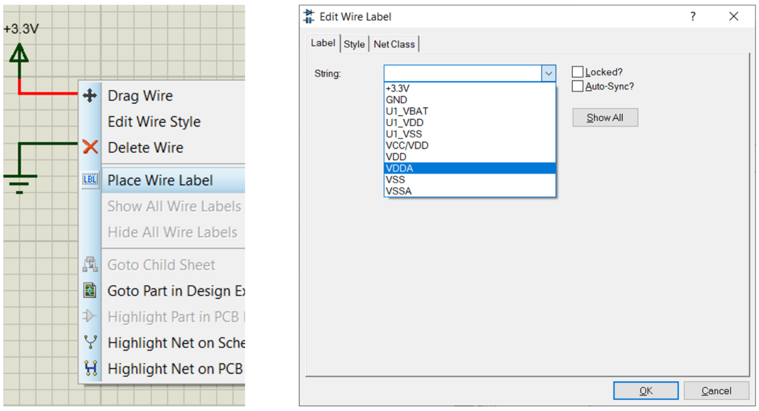

Figure 19: *Place label for Power and Ground*

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 bellow:

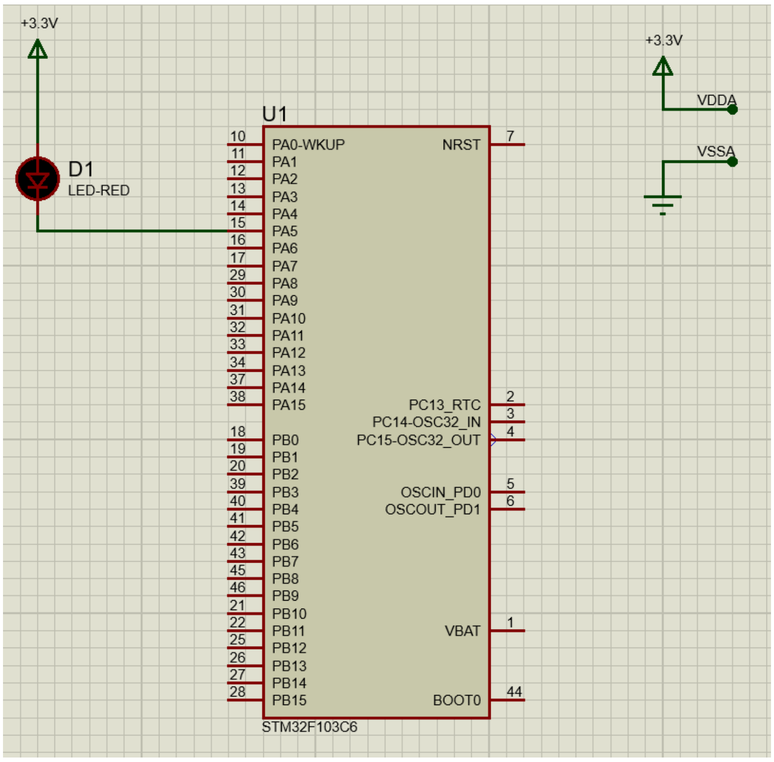

Figure 20: *Finalize the schematic*

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

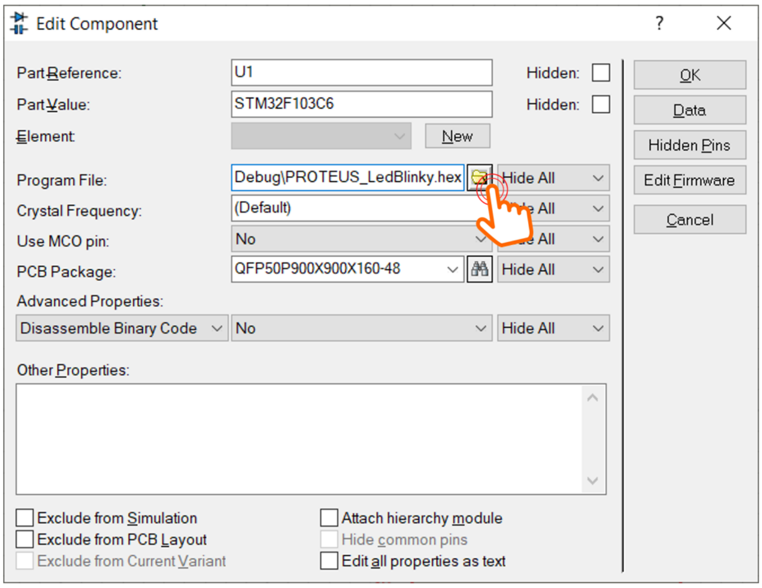

Figure 21: *Set the program of the STM32 to the hex file from Cube IDE*

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:

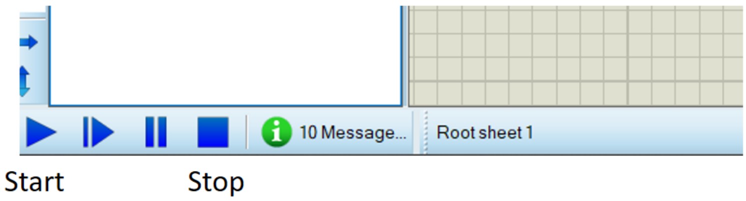

Figure 22: *Quick access buttons to start and stop the simulation*

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:

```
while (1){
  HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, GPIO_PIN_SET);
  HAL_Delay(1000);
  HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, GPIO_PIN_RESET);
  HAL_Delay(1000);
}
```

4 Exercise and Report

The source code and demo video for LAB1 are 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.

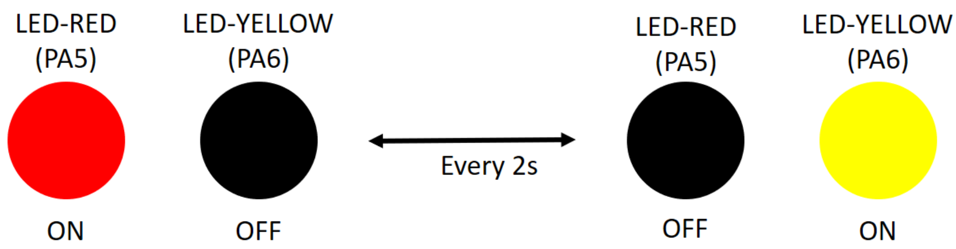

Figure 23: *State transitions for 2 LEDs*

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.

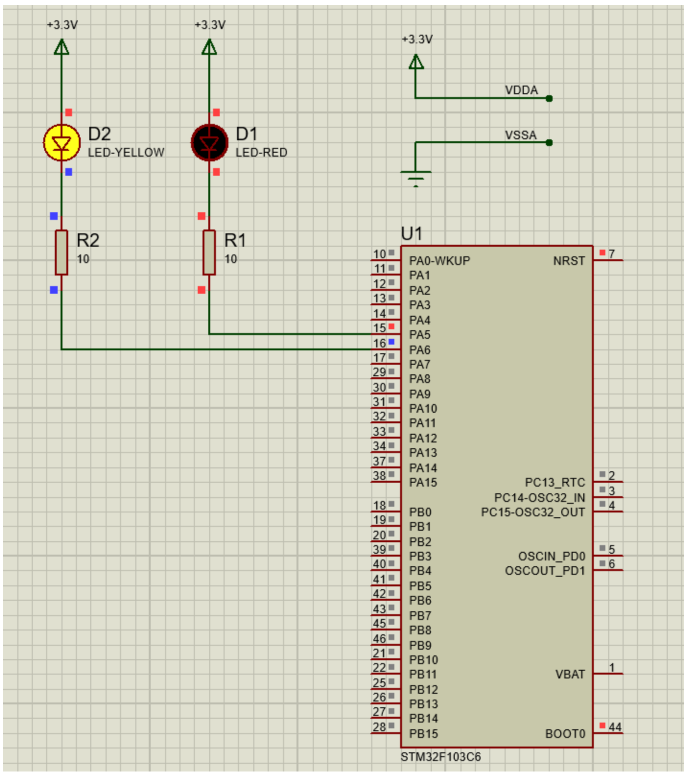

Figure 24: *Proteus Ex1 ToggleLED*

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 bellow.

CASE 1:

```
while (1)
  {
   /* USER CODE END WHILE */
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
   HAL_Delay(2000);
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
   HAL_Delay(2000);
    /* USER CODE BEGIN 3 */
  }
```

CASE 2:

```
while (1)
  {
   /* USER CODE END WHILE */
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
  HAL_Delay(2000);
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
    /* USER CODE BEGIN 3 */
  }
```

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.

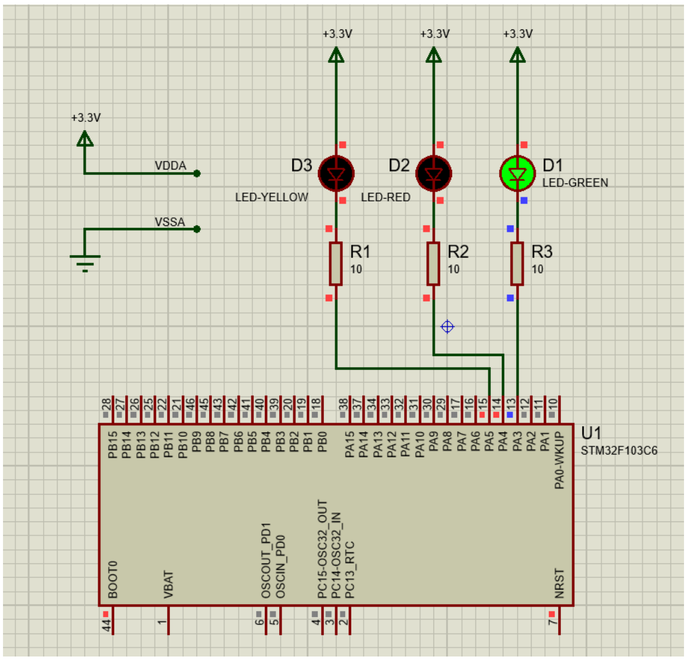

Figure 25: *Proteus Ex2 TrafficLight*

Report 2: Present the source code in while.

```c
while (1)
{
  // RED ON in 5s, others OFF
  HAL_GPIO_WritePin(GPIOA, LED_RED_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, LED_YELLOW_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, LED_GREEN_Pin, GPIO_PIN_SET);
  HAL_Delay(5000);

  // YELLOW ON in 2s, others OFF
  HAL_GPIO_WritePin(GPIOA, LED_RED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, LED_YELLOW_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, LED_GREEN_Pin, GPIO_PIN_SET);
  HAL_Delay(2000);

  // GREEN ON in 3s, others OFF
  HAL_GPIO_WritePin(GPIOA, LED_RED_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, LED_YELLOW_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(GPIOA, LED_GREEN_Pin, GPIO_PIN_RESET);
  HAL_Delay(3000);
}
```

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.

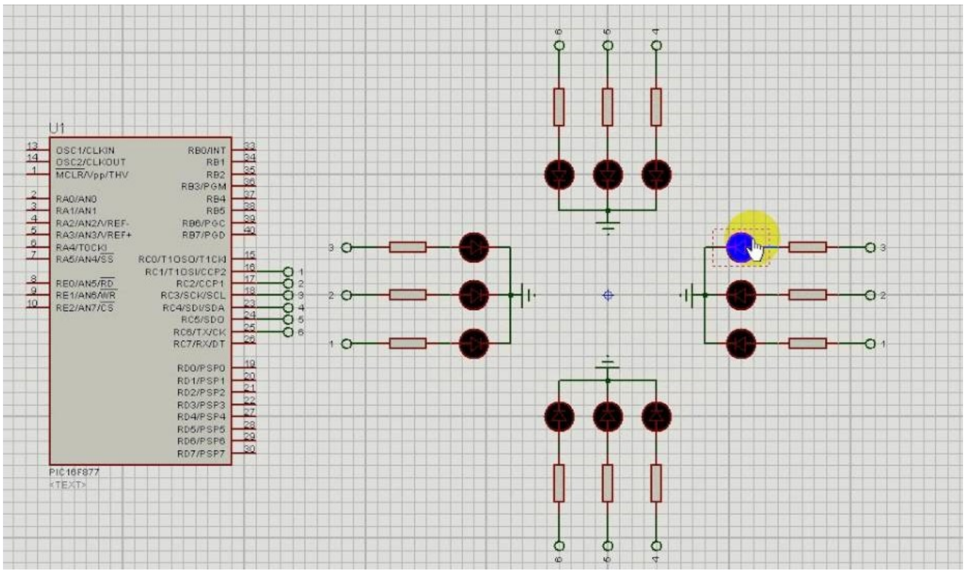

Figure 26: *Reference design for a 4 way traffic light*

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at STM32 Lab 1: 4WayTrafficLight.

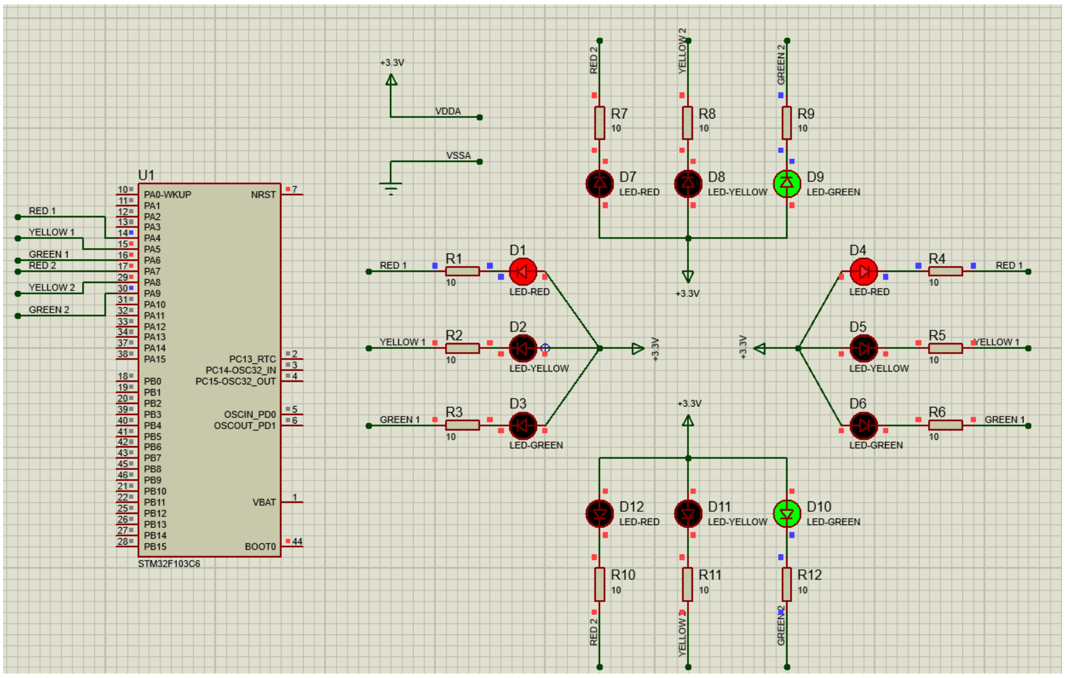

Figure 27: *Proteus Ex3 Four Way TrafficLight*

Report 2: Present the source code in while.

```
 while (1)
{
```

```c
    // LED RED 1 ON, LED GREEN 2 ON AND OTHERS OFF
    HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, RESET);
    HAL_Delay(3000);

    // LED RED 1 ON, LED YELLOW 2 ON, OTHERS LIGHT OFF
    HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, SET);
    HAL_Delay(2000);

    // LED GREEN 1 ON, LED RED 2 ON, OTHERS LIGHT OFF
    HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, SET);
    HAL_Delay(3000);

    // LED YELLOW 1 ON, LED RED 2 ON, OTHERS LIGHT OFF
    HAL_GPIO_WritePin(GPIOA, LED_GREEN1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_RED1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW1_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_RED2_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_YELLOW2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_GREEN2_Pin, SET);
    HAL_Delay(2000);
  }
```

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 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:

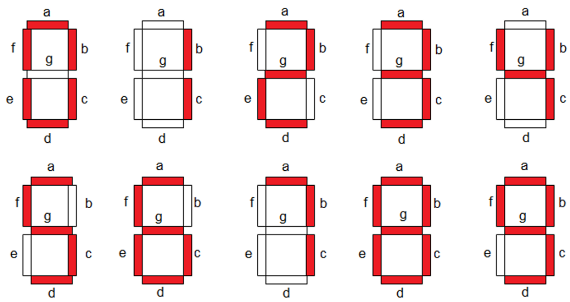

Figure 28: *Display a number on 7 segment LED*

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

```
int counter = 0;
while (1){
    if(counter >= 10) counter = 0;
    display7SEG(counter++);
    HAL_Delay(1000);
}
```

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at STM32 Lab 1: 7SEG Display7SEG.

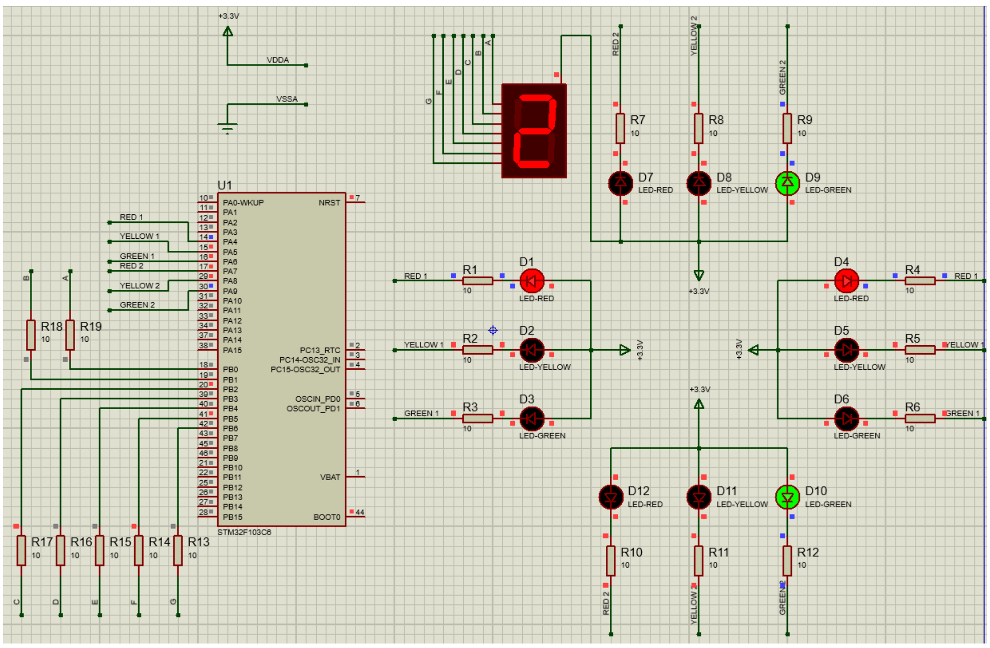

Figure 29: *Proteus Ex4 7SEG Display7SEG*

Report 2: Present the source code for display7SEG function.

```
void display7SEG(int num)
{
  //Turn off all pins first
  HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
   LED_D_Pin|LED_E_Pin | LED_F_Pin | LED_G_Pin, SET);

  switch (num)
  {
  case 0: // 0s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
   LED_D_Pin | LED_E_Pin | LED_F_Pin, RESET);
    break;
  case 1: // 1s
    HAL_GPIO_WritePin(GPIOB, LED_B_Pin | LED_C_Pin, RESET);
    break;
  case 2: // 2s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_D_Pin |
   LED_E_Pin | LED_G_Pin, RESET);
    break;
  case 3: // 3s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin |
   LED_D_Pin | LED_G_Pin, RESET);
    break;
  case 4: // 4s
    HAL_GPIO_WritePin(GPIOB, LED_B_Pin | LED_C_Pin | LED_F_Pin |
   LED_G_Pin, RESET);
    break;
  case 5: // 5s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_C_Pin | LED_D_Pin |
   LED_F_Pin | LED_G_Pin, RESET);
    break;
  case 6: // 6s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_C_Pin | LED_D_Pin |
   LED_E_Pin | LED_F_Pin | LED_G_Pin, RESET);
    break;
```

```c
  case 7: // 7s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin, RESET);
    break;
  case 8: // 8s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin | LED_D_Pin | LED_E_Pin | LED_F_Pin | LED_G_Pin, RESET);
    break;
  case 9: // 9s
    HAL_GPIO_WritePin(GPIOB, LED_A_Pin | LED_B_Pin | LED_C_Pin | LED_D_Pin | LED_F_Pin | LED_G_Pin, RESET);
    break;
  default:
    HAL_GPIO_WritePin(GPIOB, LED_G_Pin, RESET);
    break;
  }
}
```

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.

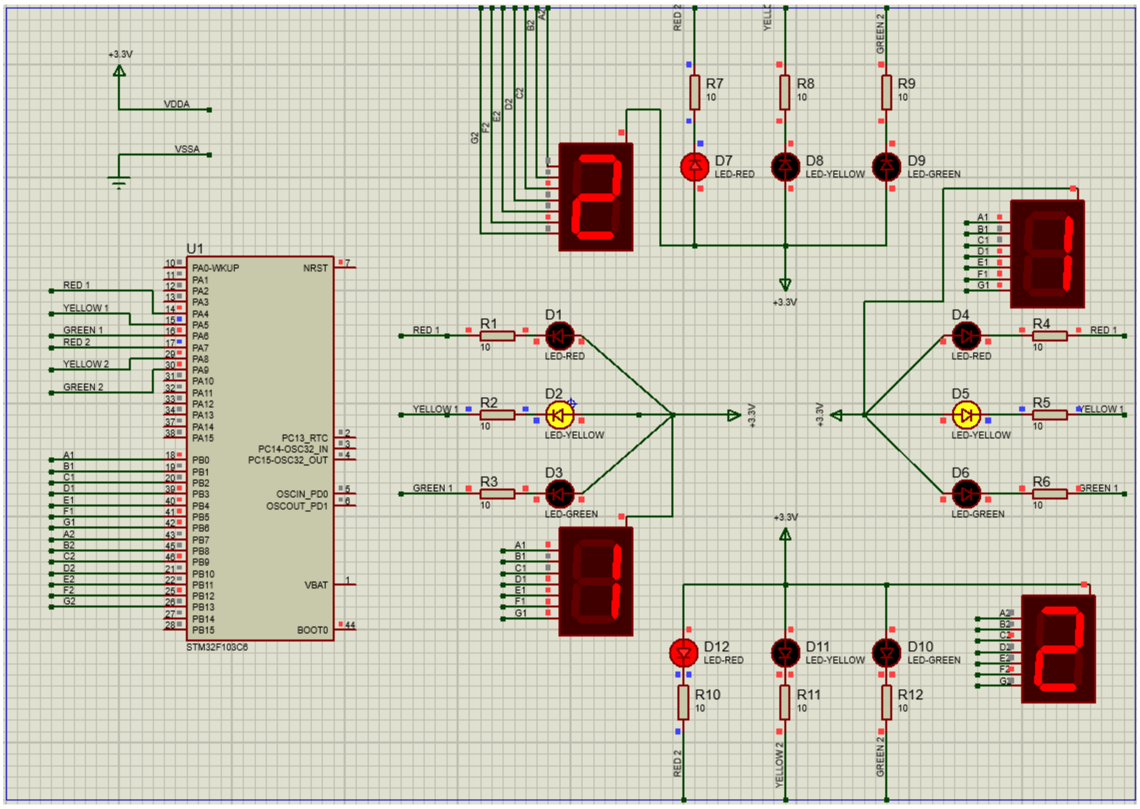

Figure 30: *Proteus Ex5 Traffic with 7SEG*

Report 2: Present the source code for display7SEG function.

```c
// Define traffic light states
enum State
{
  red,
  green,
  yellow
};
// Countdown timer variables for each intersection
int numDisplay1 = 4; // Countdown for intersection 1 (Start with red
    light: 4s)
int numDisplay2 = 2; // Countdown for intersection 2 (Start with
    green light: 3s)

// Current state of each intersection
// Intersection 1 starts with red
enum State currentStateVar1 = red;
// Intersection 2 starts with green
enum State currentStateVar2 = green;
```

```
/* Private function prototypes
   -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* Private user code
   ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
// Turn on the traffic light according to state
void setTrafficLight(GPIO_TypeDef *RED_Port, uint16_t RED_Pin,
                     GPIO_TypeDef *YELLOW_Port, uint16_t YELLOW_Pin,
                     GPIO_TypeDef *GREEN_Port, uint16_t GREEN_Pin,
                     enum State state)
{
  if (state == red)
  {
    HAL_GPIO_WritePin(RED_Port, RED_Pin, RESET);      // ON red
    HAL_GPIO_WritePin(YELLOW_Port, YELLOW_Pin, SET); // OFF yellow
    HAL_GPIO_WritePin(GREEN_Port, GREEN_Pin, SET);   // OFF green
  }
  else if (state == green)
  {
    HAL_GPIO_WritePin(RED_Port, RED_Pin, SET);        // OFF red
    HAL_GPIO_WritePin(YELLOW_Port, YELLOW_Pin, SET); // OFF yellow
    HAL_GPIO_WritePin(GREEN_Port, GREEN_Pin, RESET); // ON green
  }
  else if (state == yellow)
  {
    HAL_GPIO_WritePin(RED_Port, RED_Pin, SET);          // OFF red
    HAL_GPIO_WritePin(YELLOW_Port, YELLOW_Pin, RESET); // ON yellow
    HAL_GPIO_WritePin(GREEN_Port, GREEN_Pin, SET);     // OFF green
  }
}

/**
 *  Display numbers on a 7-segment LED
 * 7-segment LED layout:
```

```
 *      A
 *    -----
 *   |     |
 * F|   G  |B
 *   |-----|
 *   |     |
 * E|      |C
 *    -----
 *     D
 */
void display7SEG(int num, uint16_t A, uint16_t B, uint16_t C,
   uint16_t D, uint16_t E, uint16_t F, uint16_t G)
{
  if (num == 0)
  {
    // Display 0: turn on A,B,C,D,E,F - turn off G
    HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_SET);   // g - OFF
  }
  else if (num == 1)
  {
    // Display 1: turn on B,C - turn off A,D,E,F,G
    HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_SET);   // a - OFF
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_SET);   // d - OFF
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);   // f - OFF
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_SET);   // g - OFF
  }
  else if (num == 2)
```

```
{
  // Display 2: turn on A,B,D,E,G - turn off C,F
  HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
  HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
  HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_SET);   // c - OFF
  HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
  HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
  HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);   // f - OFF
  HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
}
else if (num == 3)
{
  // Display 3: turn on A,B,C,D,G - turn off E,F
  HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
  HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
  HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
  HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
  HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
  HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);   // f - OFF
  HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
}
else if (num == 4)
{
  // Display 4: turn on B,C,F,G - turn off A,D,E
  HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_SET);   // a - OFF
  HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
  HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
  HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_SET);   // d - OFF
  HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
  HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
  HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
}
else if (num == 5)
{
  // Display 5: turn on A,C,D,F,G - turn off B,E
  HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
```

```
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_SET);   // b - OFF
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
  }
  else if (num == 6)
  {
    // Display 6: turn on A,C,D,E,F,G - turn off B
    HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_SET);   // b - OFF
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
  }
  else if (num == 7)
  {
    // Display 7: turn on A,B,C - turn off D,E,F,G
    HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_SET);   // d - OFF
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_SET);   // f - OFF
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_SET);   // g - OFF
  }
  else if (num == 8)
  {
    // Display 8: turn on all A,B,C,D,E,F,G
    HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
```

```
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_RESET); // e - ON
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
  }
  else if (num == 9)
  {
    // Display 9: turn on A,B,C,D,F,G - turn off E
    HAL_GPIO_WritePin(GPIOB, A, GPIO_PIN_RESET); // a - ON
    HAL_GPIO_WritePin(GPIOB, B, GPIO_PIN_RESET); // b - ON
    HAL_GPIO_WritePin(GPIOB, C, GPIO_PIN_RESET); // c - ON
    HAL_GPIO_WritePin(GPIOB, D, GPIO_PIN_RESET); // d - ON
    HAL_GPIO_WritePin(GPIOB, E, GPIO_PIN_SET);   // e - OFF
    HAL_GPIO_WritePin(GPIOB, F, GPIO_PIN_RESET); // f - ON
    HAL_GPIO_WritePin(GPIOB, G, GPIO_PIN_RESET); // g - ON
  }
}

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  while (1)
  {
    //-----------DISPLAY NUMBERS ON 7-SEGMENT--------------//
    // Show countdown on 7-segment LED of intersection 1
    display7SEG(numDisplay1, LED_A1_Pin, LED_B1_Pin, LED_C1_Pin,
   LED_D1_Pin, LED_E1_Pin, LED_F1_Pin, LED_G1_Pin);
    // Show countdown on 7-segment LED of intersection 2
    display7SEG(numDisplay2, LED_A2_Pin, LED_B2_Pin, LED_C2_Pin,
   LED_D2_Pin, LED_E2_Pin, LED_F2_Pin, LED_G2_Pin);

    // Turn on lights according to the current state
    setTrafficLight(LED_RED1_GPIO_Port, LED_RED1_Pin,
                    LED_YELLOW1_GPIO_Port, LED_YELLOW1_Pin,
```

```
                    LED_GREEN1_GPIO_Port, LED_GREEN1_Pin,
                    currentStateVar1);

    setTrafficLight(LED_RED2_GPIO_Port, LED_RED2_Pin,
                    LED_YELLOW2_GPIO_Port, LED_YELLOW2_Pin,
                    LED_GREEN2_GPIO_Port, LED_GREEN2_Pin,
                    currentStateVar2);
    HAL_Delay(1000); // Delay 1s

    //-------------------COUNTDOWN TIMER------------------------//
     // Decrease countdown for both intersections
    --numDisplay1;
    --numDisplay2;

    // CHANGE STATE OF INTERSECTION 1 (EAST-WEST)
    // If red time ends -> switch to green (3s)
    if (numDisplay1 < 0)
    {
      if (currentStateVar1 == red)
      {
        currentStateVar1 = green;
        numDisplay1 = 2; // Green 3s
      }
      // If green time ends -> switch to yellow (2s)
      else if (currentStateVar1 == green)
      {
        currentStateVar1 = yellow;
        numDisplay1 = 1; // Yellow 2s
      }
      // If yellow time ends -> switch to red (4s)
      else
      { // yellow -> red
        currentStateVar1 = red;
        numDisplay1 = 4; // Red 4s
      }
    }
```

```


    // CHANGE STATE OF INTERSECTION 2 (NORTH-SOUTH)
    // If red time ends -> switch to green (3s)
    if (numDisplay2 < 0)
    {
      if (currentStateVar2 == red)
      {
        currentStateVar2 = green;
        numDisplay2 = 2; // Green 3s
      }
      // If green time ends -> switch to yellow (2s)
      else if (currentStateVar2 == green)
      {
        currentStateVar2 = yellow;
        numDisplay2 = 1; // Yellow 2s
      }
      // If yellow time ends -> switch to red (4s)
      else
      { // yellow -> red
        currentStateVar2 = red;
        numDisplay2 = 4; // Red 4s
      }
    }
  }
}
```

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.

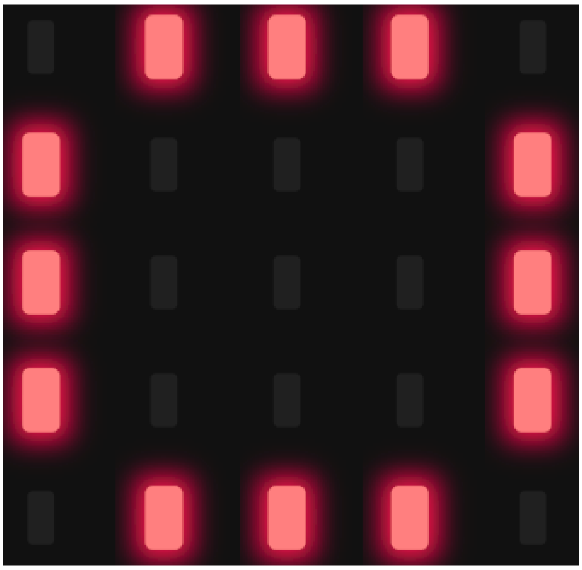

Figure 31: *12 LEDs for an analog clock*

Report 1: Present the schematic.

The source code and Proteus simulation schematic are provided at STM32 Lab 1: AnalogClock.

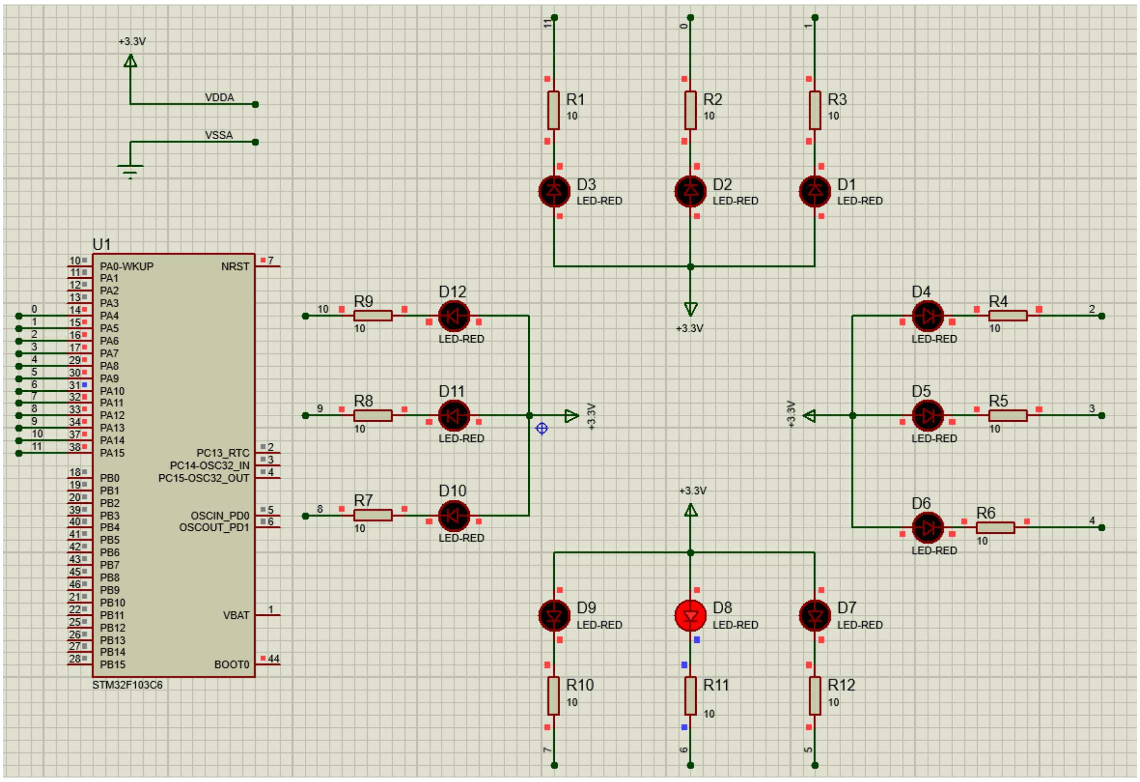

Figure 32: *Proteus Ex6 AnalogClock 12LEDs*

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

```
// Array of LED pins from LED0 - LED11
uint16_t LED_Pins[12] = {
    LED0_Pin, LED1_Pin, LED2_Pin, LED3_Pin,
    LED4_Pin, LED5_Pin, LED6_Pin, LED7_Pin,
    LED8_Pin, LED9_Pin, LED10_Pin, LED11_Pin};

// Function to turn off all LEDs
void TurnOffAllLEDs(void)
{
  for (int i = 0; i < 12; i++)
  {
    HAL_GPIO_WritePin(GPIOA, LED_Pins[i], SET); // Turn OFF LED
  }
}

// Function to turn on an LED by index
void TurnOnLED(int index)
{
  HAL_GPIO_WritePin(GPIOA, LED_Pins[index], RESET); // Turn ON LED
}

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  TurnOffAllLEDs(); // Ensure all LEDs are off at startup
  int currentLED = 0; // LED0 = position 12 on the clock

  while (1)
  {
    // Turn on the current LED
    HAL_GPIO_WritePin(GPIOA, LED_Pins[currentLED], GPIO_PIN_RESET);
```

```
    HAL_Delay(200);

    // Turn off the LED after delay
    HAL_GPIO_WritePin(GPIOA, LED_Pins[currentLED], GPIO_PIN_SET);

    // Move to the next LED
    currentLED++;
    if (currentLED >= 12)
    {
      currentLED = 0; // Wrap around to LED0 after 12 LEDs
    }
  }
}
```

4.7 Exercise 7

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

CASE 1:

```
// Function to turn off all LEDs
void clearAllClock() {
    HAL_GPIO_WritePin(GPIOA, LED0_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED1_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED2_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED3_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED4_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED5_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED6_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED7_Pin, SET);
    HALGPIO_WritePin(GPIOA, LED8_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED9_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED10_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED11_Pin, SET);
}
```

CASE 2:

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

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:

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

CASE 2:

```
// Array of LED pins from LED0 - LED11
// Each LED corresponds to one position on the clock face (0 - 11)
uint16_t LED_Pins[12] = {
    LED0_Pin, LED1_Pin, LED2_Pin, LED3_Pin,
    LED4_Pin, LED5_Pin, LED6_Pin, LED7_Pin,
    LED8_Pin, LED9_Pin, LED10_Pin, LED11_Pin};

// Turn on the LED at the given clock position
void setNumberOnClock(int index)
{
  HAL_GPIO_WritePin(GPIOA, LED_Pins[index], RESET);
}
```

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:

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

CASE 2:

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

LED Test Program

To verify the operation of the LED system, a testing function named `testAllLed()` was implemented. This function simulates different lighting sequences to ensure that the LEDs are controlled correctly. The testing procedure includes:

- **Step 1:** Sequentially turning on LEDs from position 0 to 11 and keeping them lit.
- **Step 2:** Sequentially turning off LEDs from position 0 to 11.
- **Step 3:** Turning on LEDs in reverse order, from position 11 down to 0.
- **Step 4:** Finally, turning off all LEDs to prepare for the next cycle.

In the `main()` function, the system is initialized and `testAllLed()` is called in an infinite loop, allowing the lighting effect to repeat continuously.

The source code and Proteus simulation schematic are provided at STM32 Lab 1: testAllLed().

```
// ===== LED Test Function =====
void testAllLed(void) {
    // 1. Turn on LEDs gradually from 0 -> 11 (keep them on)
    for (int i = 0; i < 12; i++) {
        setNumberOnClock(i);
        HAL_Delay(200);
    }
    // 2. Turn off LEDs gradually from 0 -> 11
    for (int i = 0; i < 12; i++) {
        clearNumberOnClock(i);
        HAL_Delay(200);
```

```c
    }
    // 3. Turn on LEDs gradually from 11 -> 0
    for (int i = 11; i >= 0; i--) {
        setNumberOnClock(i);
        HAL_Delay(200);
    }
    // 4. Turn off all LEDs
    clearAllClock();
    HAL_Delay(500);
}
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  // Main loop
  while (1)
  {
    clearAllClock();
    testAllLed();   // repeatedly call the LED test function
  }
```

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.

```c
/* Turn off all LEDs at the beginning */
clearAllClock();

// Set initial values for hour, minute, and second
int clkHr = 1;   // Hour starts at LED1
int clkMin = 3;  // Minute starts at LED3
int clkSec = 0;  // Second starts at LED0

// Turn on initial LEDs for hour, minute, and second
setNumberOnClock(clkHr);
```

```
setNumberOnClock(clkMin);
setNumberOnClock(clkSec);

int secRound = 0; // Count the number of second cycles (1 cycle = 60
    seconds)

while (1)
{
  HAL_Delay(50); // 1 real second = 1 step in seconds

  // ==== STEP 1: CLEAR OLD SECOND LED ==== //
  if (clkSec != clkMin && clkSec != clkHr)
  {
    clearNumberOnClock(clkSec);
  }

  // ==== STEP 2: INCREASE SECONDS ====
  clkSec++;
  if (clkSec > 11)
  {
    clkSec = 0;
    secRound++; // Count one full second cycle

 // ==== EVERY 5 SECOND CYCLES - MINUTE MOVES 1 STEP ==== //
    if (secRound >= 5)
    {
      secRound = 0; // Reset second cycle counter

      // CLEAR OLD MINUTE LED
      if (clkMin != clkHr && clkMin != clkSec)
      {
        clearNumberOnClock(clkMin);
      }

      // INCREASE MINUTES
      clkMin++;
```

```
      if (clkMin > 11)
      {
        clkMin = 0;

    // ==== EVERY FULL MINUTE CYCLE - HOUR MOVES 1 STEP ==== //
        if (clkHr != clkMin && clkHr != clkSec)
        {
          clearNumberOnClock(clkHr);
        }

        clkHr++;
        if (clkHr > 11)
          clkHr = 0;

        setNumberOnClock(clkHr); // Turn on new hour LED
      }
      setNumberOnClock(clkMin); // Turn on new minute LED
    }
  }

  // ==== TURN ON NEW SECOND LED ====
  setNumberOnClock(clkSec);
}
```

Tài liệu