Anthony Chavez

EEE 174-CpE 185 Summer 2020

Monday, Wednesday

Lab 0

Introduction to Microcontrollers and Lab Tools

Dennis Dahlquist

Introduction:

The objective of this lab is to gain knowledge of the basic IO, operations, and development of a microcontroller device. Instead of using the STM32 Nucleo F303K8 microcontroller, the STM32 Nucleo L432KC will be used. In addition, the software we will be using will be the STM32CubeMX and the Atollic TrueStudio to code and debug.

Part 1: General Purpose Input / Output

A GPIO is a uncommitted pin on a chip that can be configured into a digital input or output. This configuration process is setup during runtime and there are 3 important built in hardware to the structure of a STM32 GPIO. They are the pull up and pull down resistors, the Schmitt trigger to help avoid input signal and contact bounce, and the n-mos and p-mos help manage the pin states in push-pull mode and in open-drain mode. In addition, the STM32 I/O pins allow for 4 different types of modes: Digital input, Digital output, Alternate which allows the GPIO pin to be accessed by other peripherals such as UART, SPI, etc., and Analog I/O. Often digital IO is represented as a binary value with 1 being High and 0 being Low, but actually it is in analog. It is important to note that the STM32 board's IO pins are 3.3V, but most pins are 5V tolerant. However, it's best to check if the pin is 5V tolerant before connecting a 5V input to it to prevent any damage to the microcontroller.

For this portion of the lab, we learned how to configure the STM32CubeMX software to generate the code template that will then be modified in the Atollic TrueStudio IDE and debugged on the breadboard. For the first project, we connect the push button and led circuit as shown in Figure 1.1. Then, we match the configuration of the pin out as shown in Figure 1.2, the GPIO configuration in Figure 1.3, and the clock configuration in Figure 1.4. Finally we insert the code in Figure 1.5 into the main.c file of the program in the Atollic TrueStudio IDE, run the debugger and make notes of the behavior of the LED when stepping through the program (see Figures 1.6 and 1.7). For the second project, the same breadboard and configuration was used with the exception of using a pin to act as a external interrupt and only adding code to the stm32l4xx_it.c file (see Figure 1.8 and 1.10). While running the debugger we made note of the behavior of the LED (see Figure 1.11).

For the demo project, we added two more LEDs (see Figure 1.14) and configured the code to create a unique pattern when the push button is depressed (see Figure 1.13) and returns the LEDs to static on position when the push button is open.

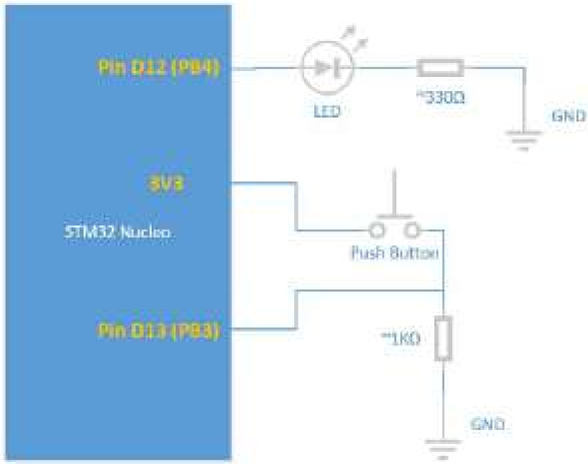


Figure 1.1: Block Diagram of push button and led circuit

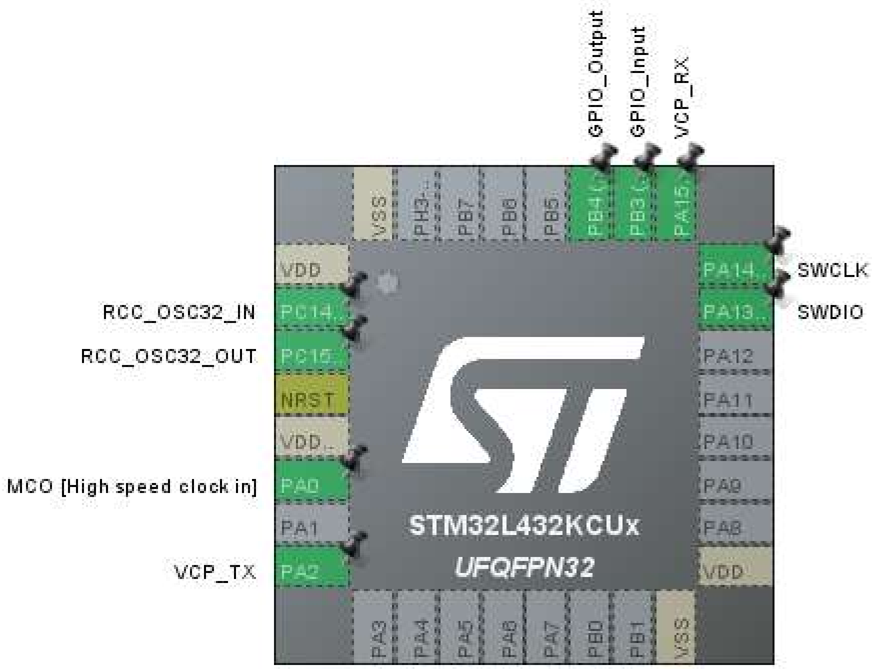


Figure 1.2: Pin Out for Project 1

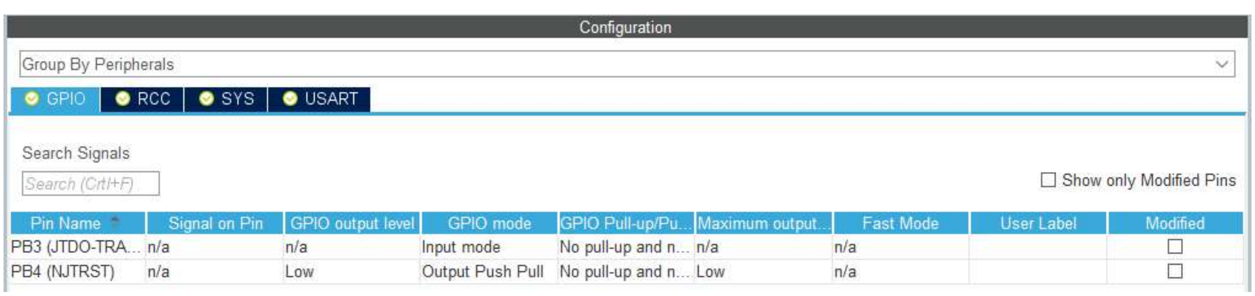


Figure 1.3: GPIO Configuration

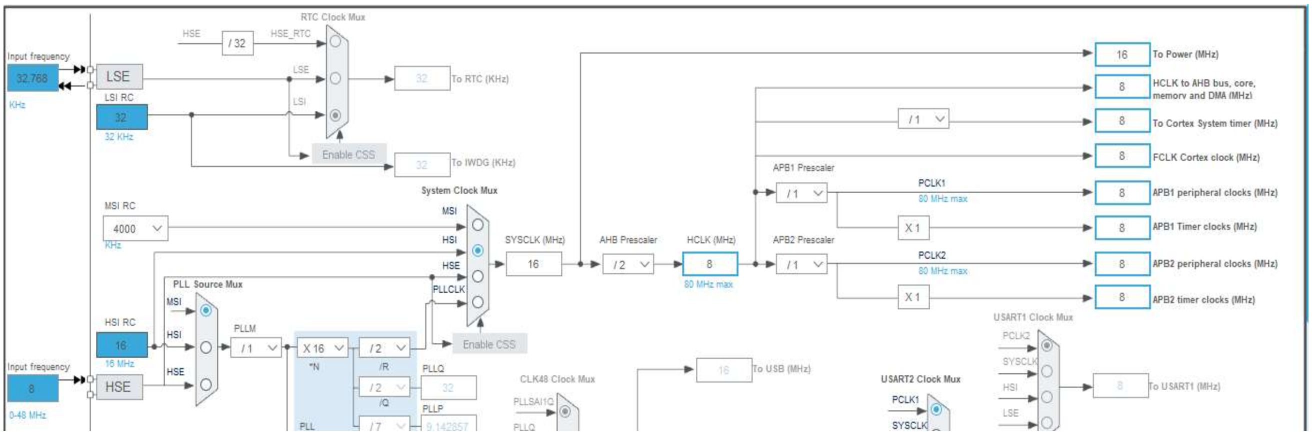


Figure 1.4: Clock Configuration

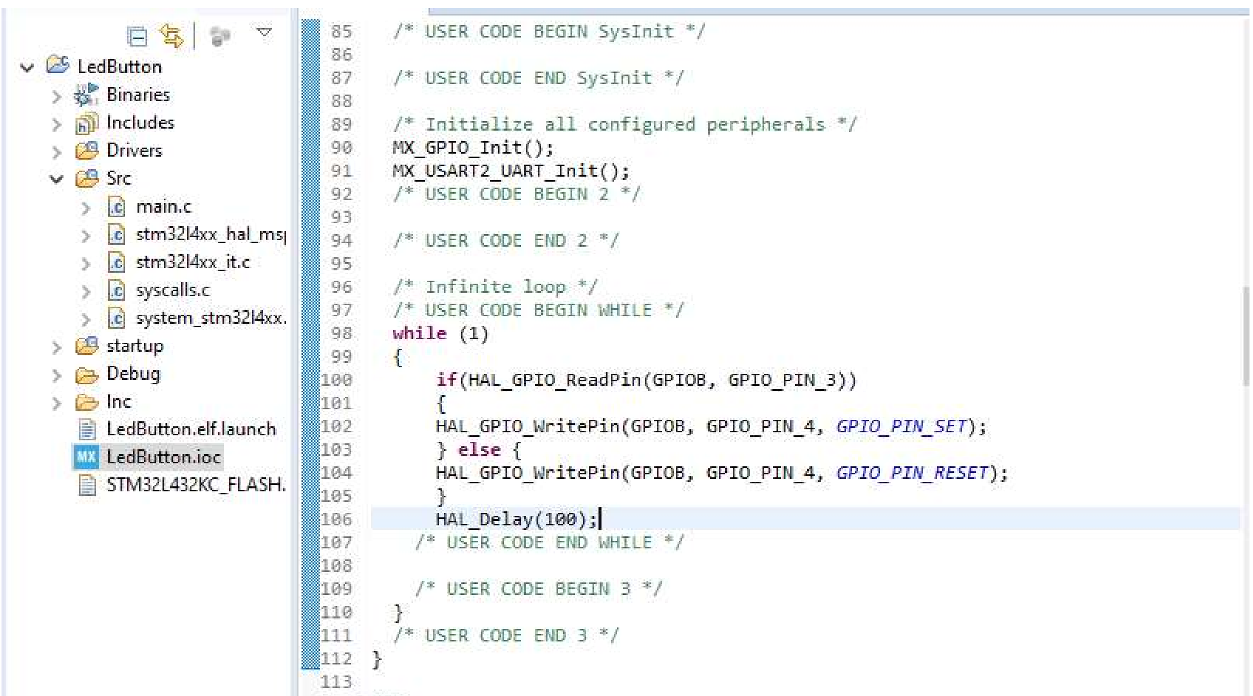
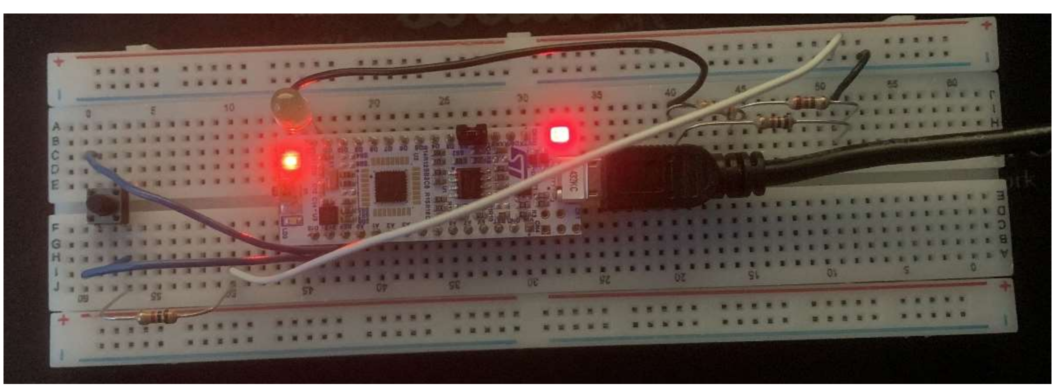


Figure 1.5: Code added to main.c in Atollic TrueStudio



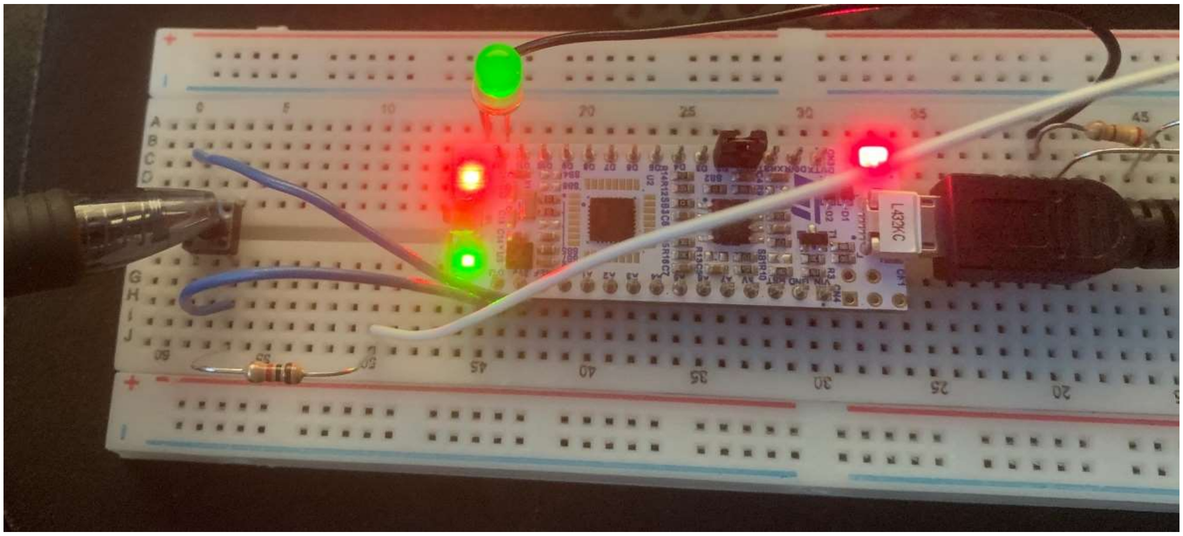


Figure 1.6: Bread Board During Debug Runs

```
Using the BreakPoint; First Run part b
Loop 1:
    If statement    1
    else statement  1
    delay           0
Loop 2:
    If statement    0
    else statement  0
    delay           1
Loop 3:
    If statement    1
    else statement  1
    delay           1
Loop 4:
    If statement    1
    else statement  1
    delay           1
Loop 5:
    If statement    1
    else statement  1
    delay           1
Loop 6:
    If statement    1
    else statement  1
    delay           1
```

Figure 1.7: LED Circuit Debugger Results

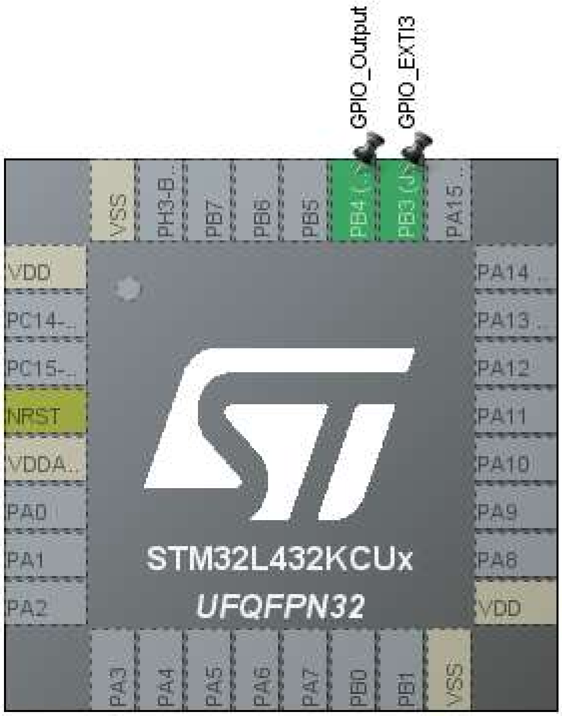
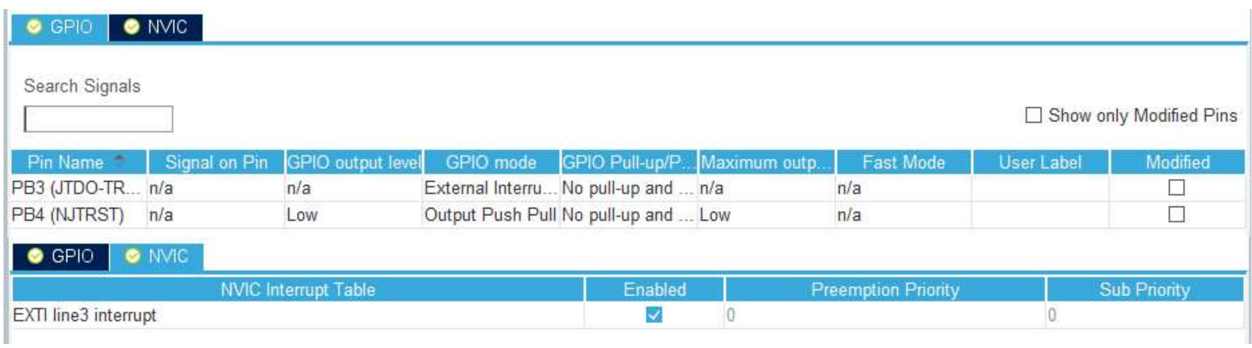


Figure 1.8: Pin Out for Project 2



Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/P...	Maximum outp...	Fast Mode	User Label	Modified
PB3 (JTDO-TR...	n/a	n/a	External Interru...	No pull-up and ...	n/a	n/a		☐
PB4 (NJTRST)	n/a	Low	Output Push Pull	No pull-up and ...	Low	n/a		☐

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line3 interrupt	☑	0	0

Figure 1.9: GPIO and NVIC Configuration

```
void EXTI3_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI3_IRQn 0 */
  HAL_GPIO_TogglePin(GPIOB,GPIO_PIN_4);//Toggle the state of pin PC9
  /* USER CODE END EXTI3_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_3);
  /* USER CODE BEGIN EXTI3_IRQn 1 */

  /* USER CODE END EXTI3_IRQn 1 */
}
```
Figure 1.10: Code added to stm32l4xx.it.c file in Atollic TrueStudio

```
Second Run part c
Loop 1:
    1st push        1
    2nd push        0
Same sequence repeats
```

Figure 1.11: LED Circuit Debugger Results

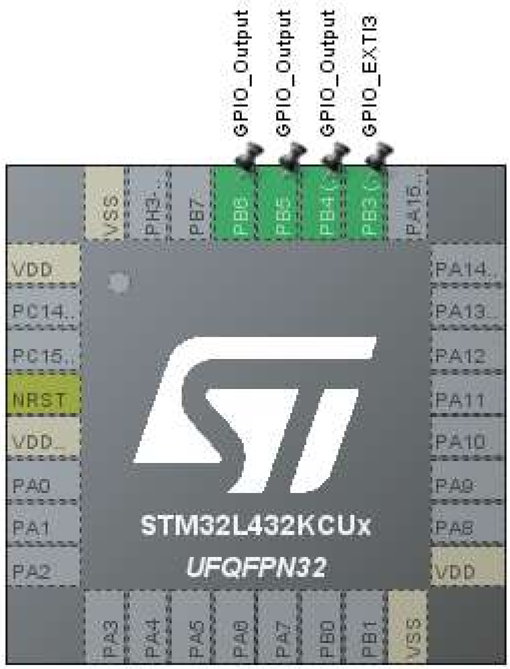


Figure 1.12: LED Circuit Demo Pin Out

```c
while (1)
{
    if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)) {
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET);
        HAL_Delay(500);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
        HAL_Delay(500);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_SET);
        HAL_Delay(500);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET);
      } else {
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);
      }
    HAL_Delay(500);
  /* USER CODE END WHILE */
```
Figure 1.13: LED Circuit Demo Code Pattern

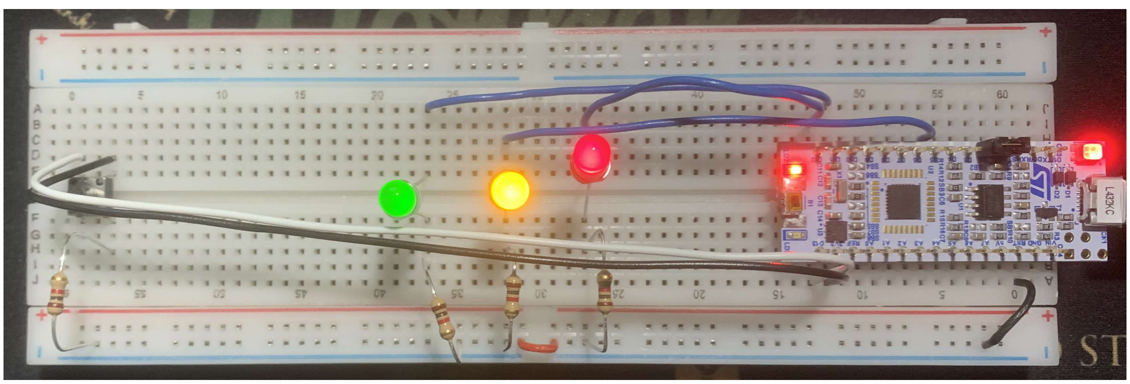


Figure 1.14: LED Circuit For Demo

Part 2: Finite State Machines Software Design Pattern

For this portion of the lab, we will be exploring Finite State Machines (FSM). These are a mathematical model that can be in exactly one state at a given time, one example we will investigate is a traffic light system (see Figure 2.1). As seen in the figure, there is a 2-way intersection with perpendicular roads going North-South and East-West. A timed light will allow one light to remain on for some period of time and transition to allow the other road to pass. To describe the functionality of this traffic light controller, see Figure 2.2 and 2.3. We will be implementing this project using our microcontroller and 6 LEDs with at least 3 different colors. First, we configure the pin out configuration for our board in the STM32CubeMX software to match Figure 2.4 and then wire the bread board to look something like Figure 2.5. Second, we copy the provided code in Figure 2.6 into the project file in the Atollic TrueStudio IDE under the appropriate sections. Finally, we run the code in the debugger and see if the LEDs follow the State Machine Diagrams.

Now for the demo portion of the lab, we simply had to modify the code to make the North-South road traffic light turn green only when the push button is pressed. This means the East-West road will remain green, allow traffic, until a car arrives to the North-South road. One state that can be removed is the All Stop EW or NS procedure since both set the Red LED, but I left the state for readability (see Figures 2.8 and 2.9). As for the code, I created a simple if-else statement to wait until the button is push (see Figure 2.10). I used the original breadboard created in the beginning of part 2 and tested the code using the debugger (see Figure 2.11).

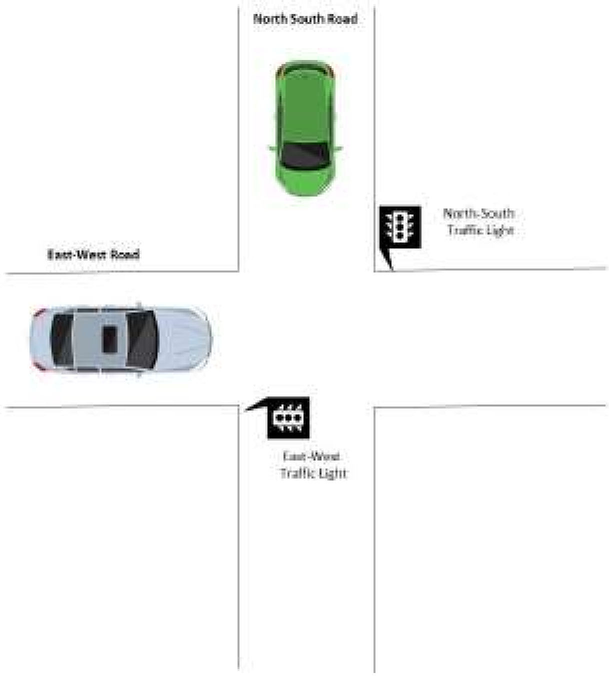


Figure 2.1: Traffic Light Controller Application

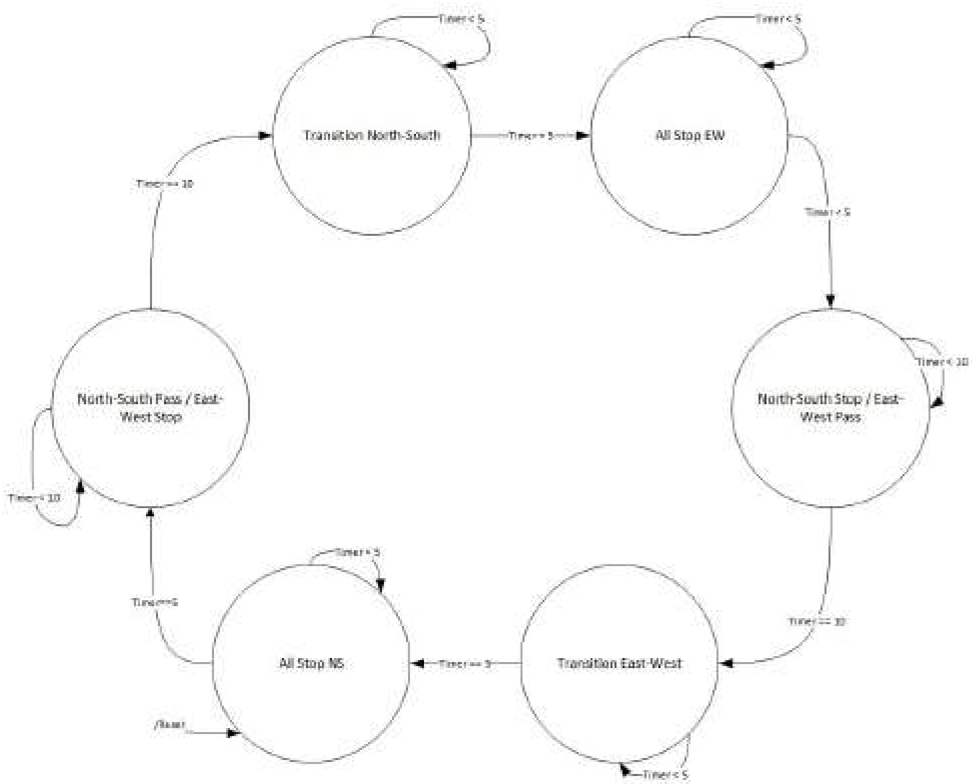


Figure 2.2: Traffic Light Controller Finite State Machine

Curren State	Input	Next State	Output
North-South Pass/ East-West Stop	Timer delay < 10s	North-South Pass/ East-West Stop	NS_Green_led = 1 NS_Yellow_led=0 NS_Red_led=0 EW_Green_led =0 EW_Yellow_led=0 EW_Red_led=1
	Timer delay == 10s	Transition North-South	
Transition North-South	Timer delay < 5s	Transition North-South	NS_Green_led = 0 NS_Yellow_led=1 NS_Red_led=0 EW_Green_led =0 EW_Yellow_led=0 EW_Red_led=1
	Timer delay == 5s	All Stop EW	
All Stop EW	Timer delay < 5s	All Stop EW	NS_Green_led = 0 NS_Yellow_led=0 NS_Red_led=1 EW_Green_led =0 EW_Yellow_led=0 EW_Red_led=1
	Timer delay == 5s	North-South Stop/ East-West Pass	
North-South Stop/ East-West Pass	Timer delay < 10s	North-South Stop/ East-West Pass	NS_Green_led = 0 NS_Yellow_led=0 NS_Red_led=1 EW_Green_led =1 EW_Yellow_led=0 EW_Red_led=0
	Timer delay == 10s	Transition East-West	
Transition East-West	Timer delay < 5s	Transition East-West	NS_Green_led = 0 NS_Yellow_led=0 NS_Red_led=1 EW_Green_led =0 EW_Yellow_led=1 EW_Red_led=0
	Timer delay == 5s	All Stop NS	
All Stop NS	Timer delay < 5s	All Stop NS	NS_Green_led = 0 NS_Yellow_led=0 NS_Red_led=1 EW_Green_led =0 EW_Yellow_led=0 EW_Red_led=1
	Timer delay == 5s	North-South Pass/ East-West Stop	

Figure 2.3: Traffic Light Controller Finite State Machine I/O

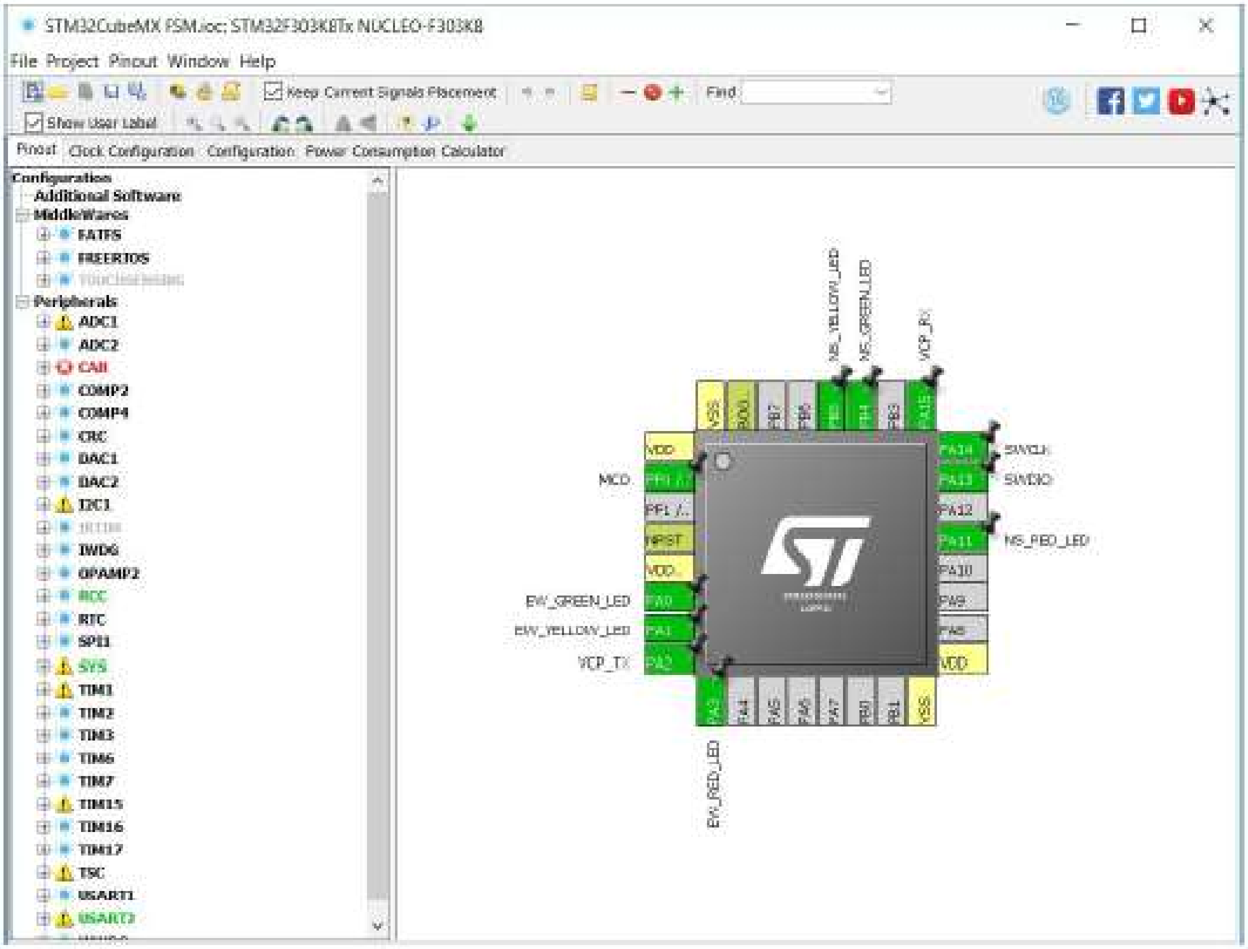


Figure 2.4: Traffic Light Controller STM32CubeMX pin configuration

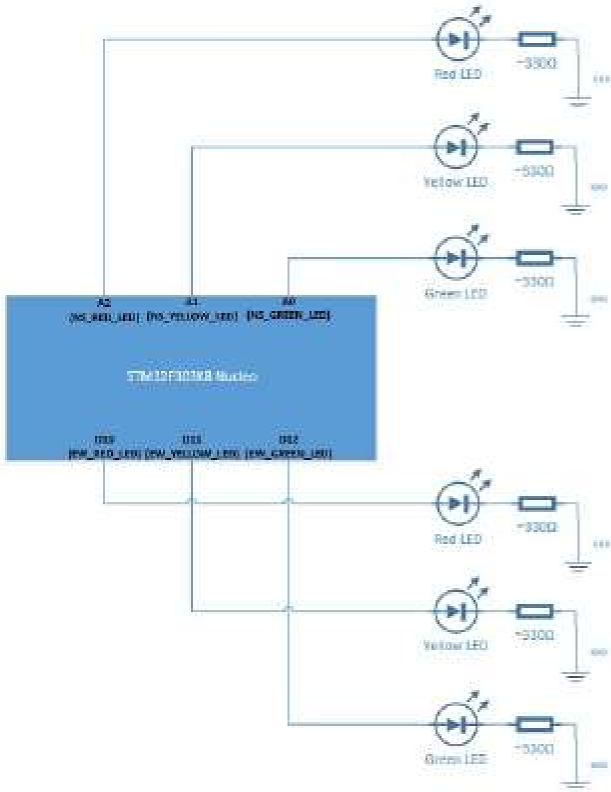


Figure 2.5: Traffic Light Controller STM32CubeMX Wiring Diagram

```
typedef enum
{
            Transition_NS_State,
            NS_Pass_EW_Stop_State,
            All_Stop_EW_State,
            NS_Stop_EW_Pass_State,
            Transition_EW_State,
            All_Stop_NS_State
}eSystemState;

/* Prototype Event Handlers */
eSystemState NorthSouthPassHandler(void)
{
            HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);
            HAL_Delay(10*1000); //10 seconds
            return Transition_NS_State;
}
```

```
eSystemState TransitionNorthSouthHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);
	HAL_Delay(5*1000); //5 seconds
	return All_Stop_EW_State;
}
eSystemState AllStopEastWestHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);
	HAL_Delay(5*1000); //5 seconds
	return NS_Stop_EW_Pass_State;
}
eSystemState EastWestPassHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_RESET);
	HAL_Delay(10*1000); //10 seconds
	return Transition_EW_State;
}
eSystemState TransitionEastWestHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_RESET);
	HAL_Delay(5*1000); //5 seconds
	return All_Stop_NS_State;
}
eSystemState AllStopNorthSouthHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);
```

```
        HAL_Delay(5*1000); //5 seconds
        return NS_Pass_EW_Stop_State;
}
/**
* @brief The application entry point.
*
* @retval None
*/
int main(void)
{
/* MCU Configuration----------------------------------------------------------*/
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();
/* Configure the system clock */
SystemClock_Config();
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
/*Declare eNextState and initialize it to All Stop North South */
eSystemState eNextState = Transition_NS_State;
/* Infinite loop */
while (1)
{
        switch(eNextState)
        {
                case Transition_NS_State:
                        eNextState = TransitionNorthSouthHandler();
                        break;
                case NS_Pass_EW_Stop_State:
                        eNextState = NorthSouthPassHandler();
                        break;
                case All_Stop_EW_State:
                        eNextState = AllStopEastWestHandler();
                        break;
                case NS_Stop_EW_Pass_State:
                        eNextState = EastWestPassHandler();
                        break;
                case Transition_EW_State:
                        eNextState = TransitionEastWestHandler();
                        break;
                case All_Stop_NS_State:
                        eNextState = AllStopNorthSouthHandler();
                        break;
                default:
                        eNextState = AllStopNorthSouthHandler();
                        break;
        }
}
/* USER CODE END 3 */
}
```

Figure 2.6: Traffic Light Controller Provided Code

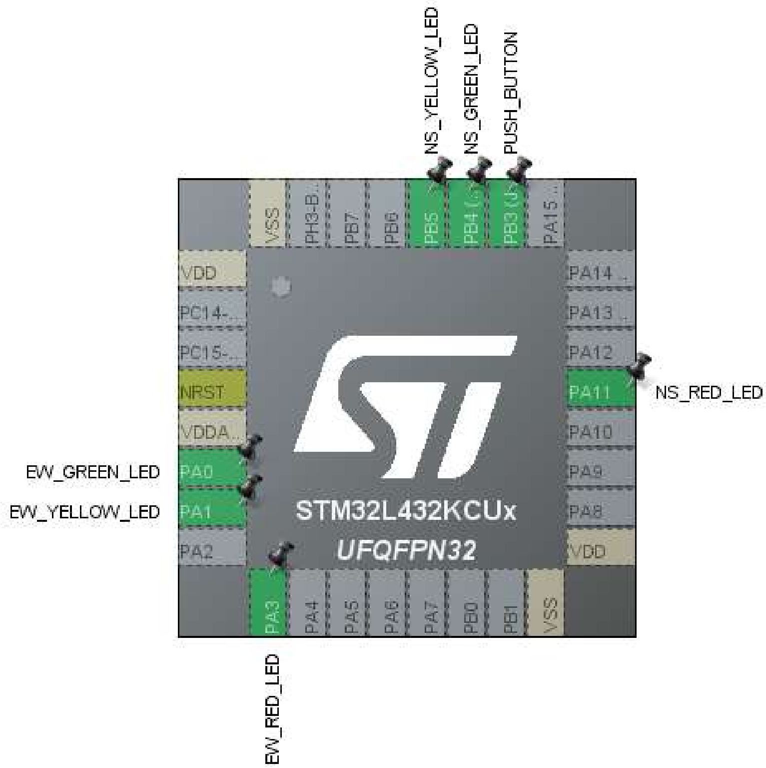


Figure 2.7: Traffic Light Controller Demo Pin Out

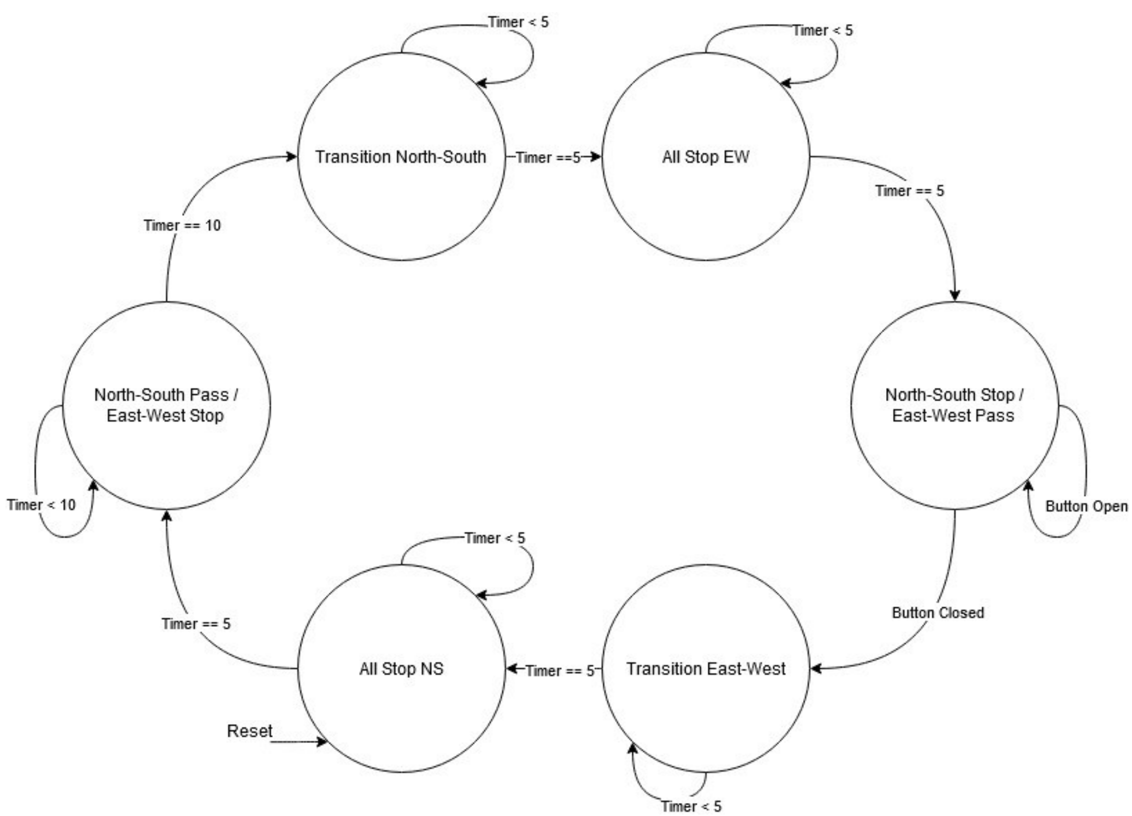


Figure 2.8: Traffic Light Controller Demo Finite State Machine

Current State	Input	Next State	Output
North-South Pass / East-West Stop	Timer delay < 10s	North-South Pass / East-West Stop	NS_RED_LED = 0 NS_YELLOW_LED = 0 NS_GREEN_LED = 1 EW_RED_LED = 1 EW_YELLOW_LED = 0 EW_GREEN_LED = 0
	Timer delay == 10s	Transition North-South	
Transition North-South	Timer delay < 5s	Transition North-South	NS_RED_LED = 0 NS_YELLOW_LED = 1 NS_GREEN_LED = 0 EW_RED_LED = 1 EW_YELLOW_LED = 0 EW_GREEN_LED = 0
	Timer delay == 5s	All Stop EW	
All Stop EW	Timer delay < 5s	All Stop EW	NS_RED_LED = 1 NS_YELLOW_LED = 0 NS_GREEN_LED = 0 EW_RED_LED = 1 EW_YELLOW_LED = 0 EW_GREEN_LED = 0
	Timer delay == 5s	North-South Stop / East-West Pass	
North-South Stop / East-West Pass	Button Open	North-South Stop / East-West Pass	NS_RED_LED = 1 NS_YELLOW_LED = 0 NS_GREEN_LED = 0 EW_RED_LED = 0 EW_YELLOW_LED = 0 EW_GREEN_LED = 1
	Button Closed	Transition East-West	
Transition East-West	Timer delay < 5s	Transition East-West	NS_RED_LED = 1 NS_YELLOW_LED = 0 NS_GREEN_LED = 0 EW_RED_LED = 0 EW_YELLOW_LED = 1 EW_GREEN_LED = 0
	Timer delay == 5s	All Stop NS	
All Stop NS	Timer delay < 5s	All Stop NS	NS_RED_LED = 1 NS_YELLOW_LED = 0 NS_GREEN_LED = 0 EW_RED_LED = 1 EW_YELLOW_LED = 0 EW_GREEN_LED = 0
	Timer delay == 5s	North-South Pass / East-West Stop	

Figure 2.9: Traffic Light Controller Demo Finite State Machine I/O

```
eSystemState NorthSouthPassHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_RESET);

	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);

	HAL_Delay(10*1000); //10 seconds
	return Transition_NS_State;
}

eSystemState TransitionNorthSouthHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_RESET);

	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);

	HAL_Delay(5*1000); //5 seconds
	return All_Stop_EW_State;
}

eSystemState AllStopEastWestHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);

	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);

	HAL_Delay(5*1000); //5 seconds
	return NS_Stop_EW_Pass_State;
}
```

```
eSystemState EastWestPassHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);

	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_RESET);

	HAL_Delay(10*1000); //10 seconds
	return NS_Stop_EW_Pass_State;
}

eSystemState TransitionEastWestHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);

	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_SET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_RESET);

	HAL_Delay(5*1000); //5 seconds
	return All_Stop_NS_State;
}

eSystemState AllStopNorthSouthHandler(void)
{
	HAL_GPIO_WritePin(GPIOB, NS_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOB, NS_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, NS_RED_LED_Pin, GPIO_PIN_SET);

	HAL_GPIO_WritePin(GPIOA, EW_GREEN_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_YELLOW_LED_Pin, GPIO_PIN_RESET);
	HAL_GPIO_WritePin(GPIOA, EW_RED_LED_Pin, GPIO_PIN_SET);

	HAL_Delay(5*1000); //5 seconds
	return NS_Pass_EW_Stop_State;
}
```

```
while (1)
{
        switch(eNextState)
        {
              case NS_Stop_EW_Pass_State:
                    eNextState = EastWestPassHandler();
                    if(HAL_GPIO_ReadPin(GPIOB, PUSH_BUTTON_Pin))
                          eNextState = TransitionEastWestHandler();
                    break;
              case Transition_EW_State:
                    eNextState = TransitionEastWestHandler();
                    break;
              case All_Stop_NS_State:
                    eNextState = AllStopNorthSouthHandler();
                    break;
              case NS_Pass_EW_Stop_State:
                    eNextState = NorthSouthPassHandler();
                    break;
              case Transition_NS_State:
                    eNextState = TransitionNorthSouthHandler();
                    break;
              case All_Stop_EW_State:
                    eNextState = AllStopEastWestHandler();
                    break;
              default:
                    eNextState = AllStopNorthSouthHandler();
                    break;
        }
}
```

Figure 2.10: Traffic Light Controller Demo Code

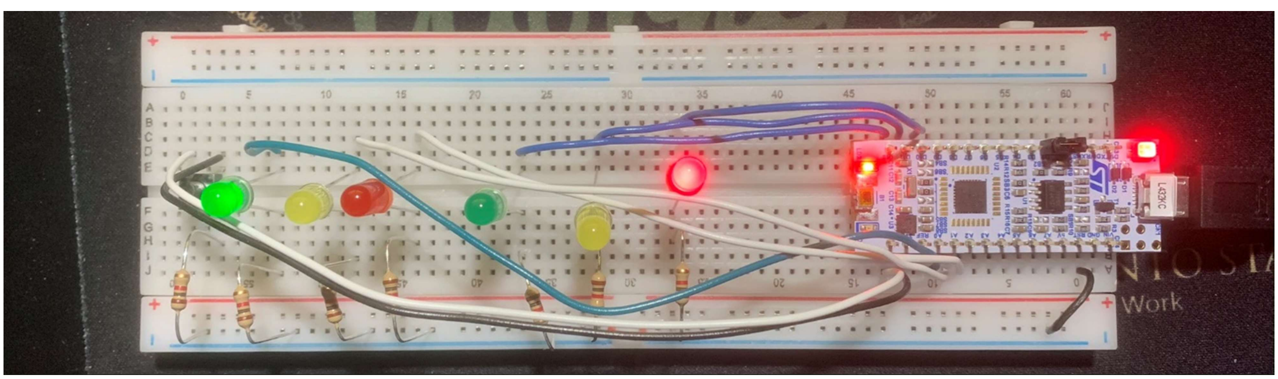


Figure 2.11: Traffic Light Controller Bread Board Configuration

Conclusion:

At first, I was struggling with understanding the code we were provided since no explanation was provided for what the code syntax says, only what it's supposed to achieve. I watched a few videos on YouTube to get a better understanding of the coding syntax and was successful in finding answers to many of my questions. In addition, I downloaded a pin out diagram from the microcontroller's website to understand which pin on the STM32MX software matched on the physical microcontroller. Once I understood where which pins connecting in the software and what functions to add to them, the lab was fairly simple. However, I feel that part 2 of the lab helped the most with understanding the coding process of building a project. After completing this lab, I feel a little more comfortable with working with the microcontroller and understand what steps to take to configure and test a project.