

Lab #0 Introduction to Micro-Controllers and Lab Tools

David Quintanilla

CpE 185

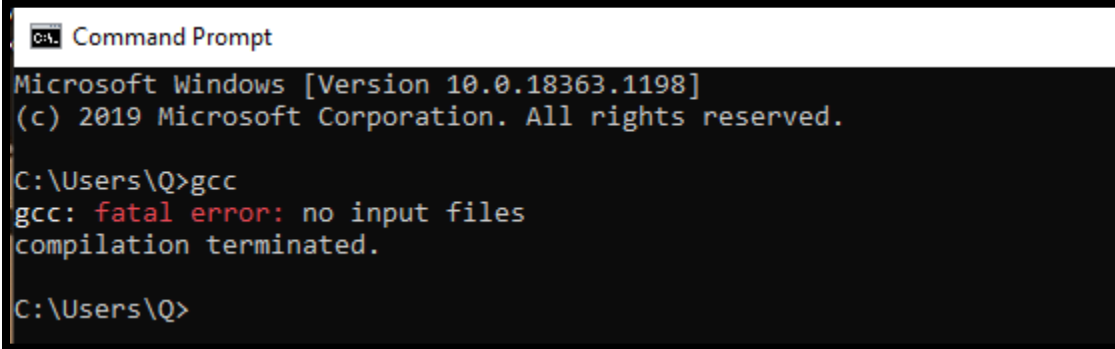
Prof. Sean Kennedy

Introduction

In Lab 0, I learned the basic understanding of basic IO operations, and development of microcontroller devices. Using the STM32CubeIDE created an application, edited IO pins and constructed a circuit on the breadboard that ran from the STM32Cube.

Part 0 : Installing Software

In order to follow the steps in this lab I had to install the correct software which could be found from <http://www.mingw.org/>. After installing the MinGW application I completed Part 0 by typing “gcc” into the command prompt and comparing to the instructions provided.



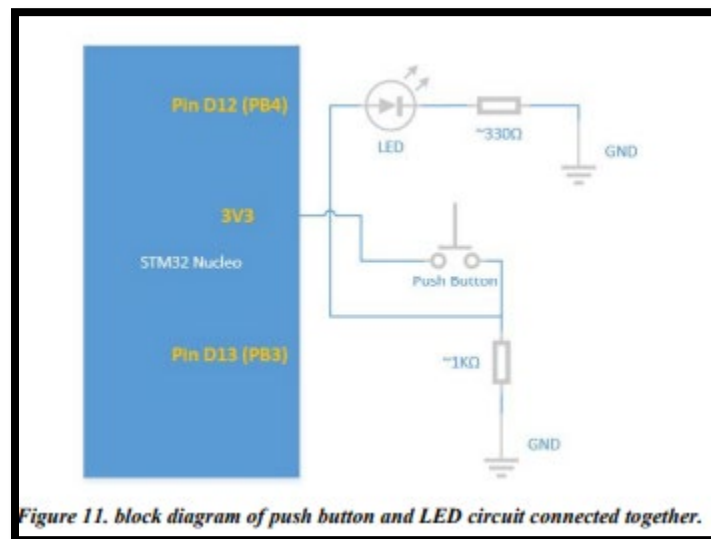
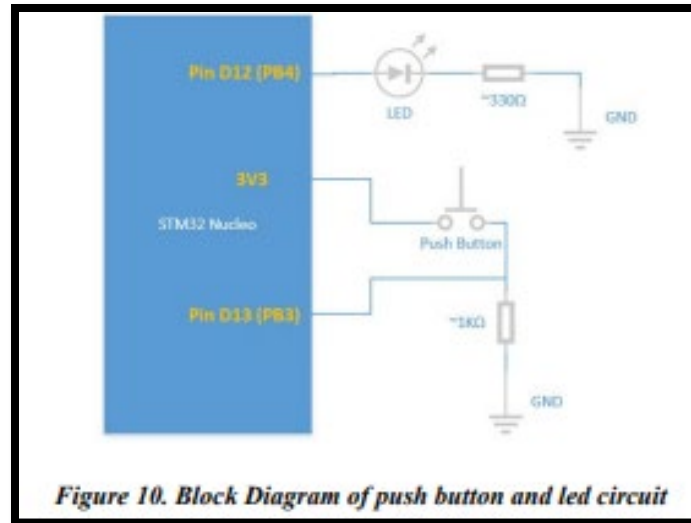
```
Command Prompt
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Q>gcc
gcc: fatal error: no input files
compilation terminated.

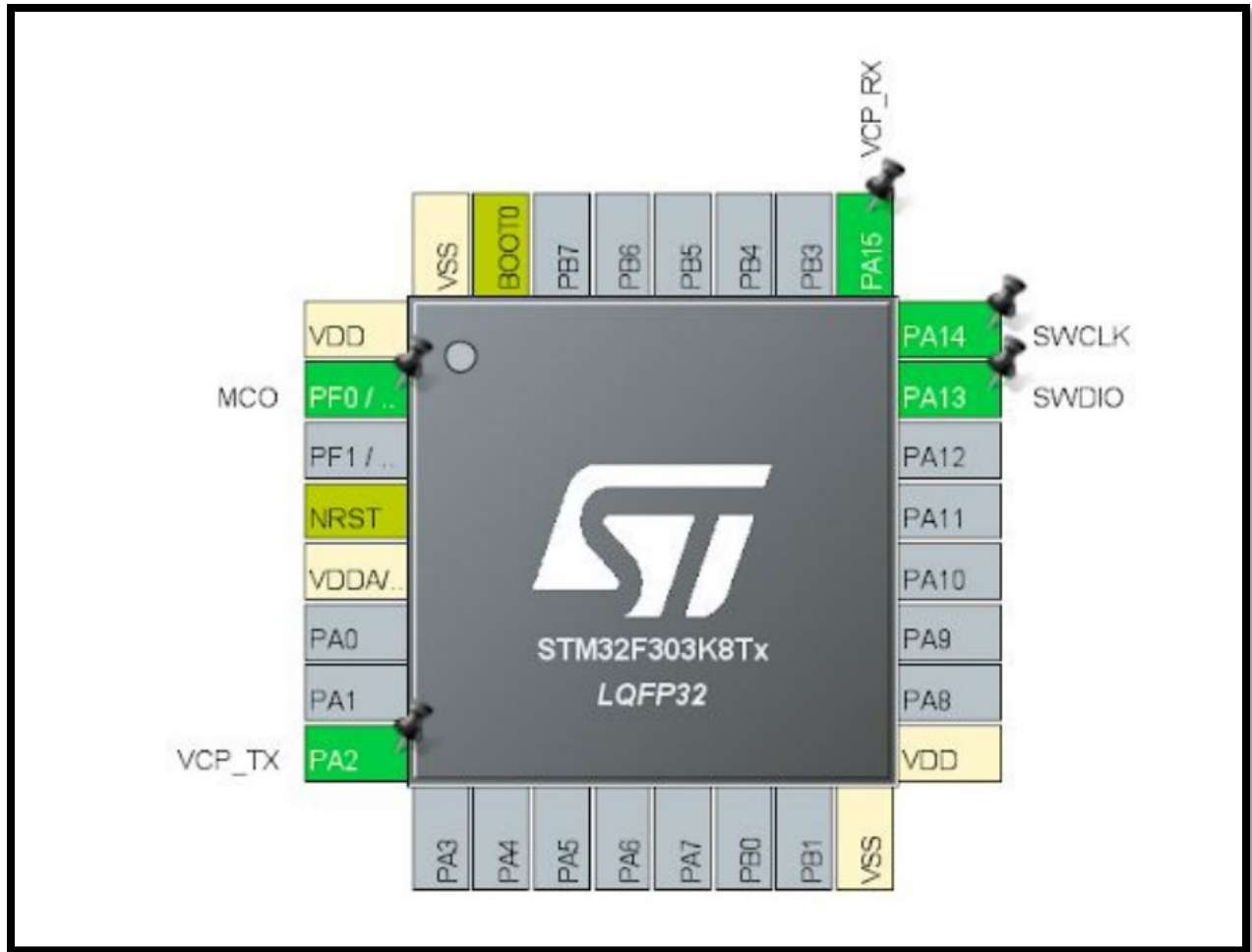
C:\Users\Q>
```

Part 1 : General Purpose Input / Output

Part 1 had to connect the STM32 to the breadboard and create a simple push button with a LED. Following the block diagrams in Figure 10 and Figure 1, I created a configure the STM32CubeIDE to match the following parameters of the lab.

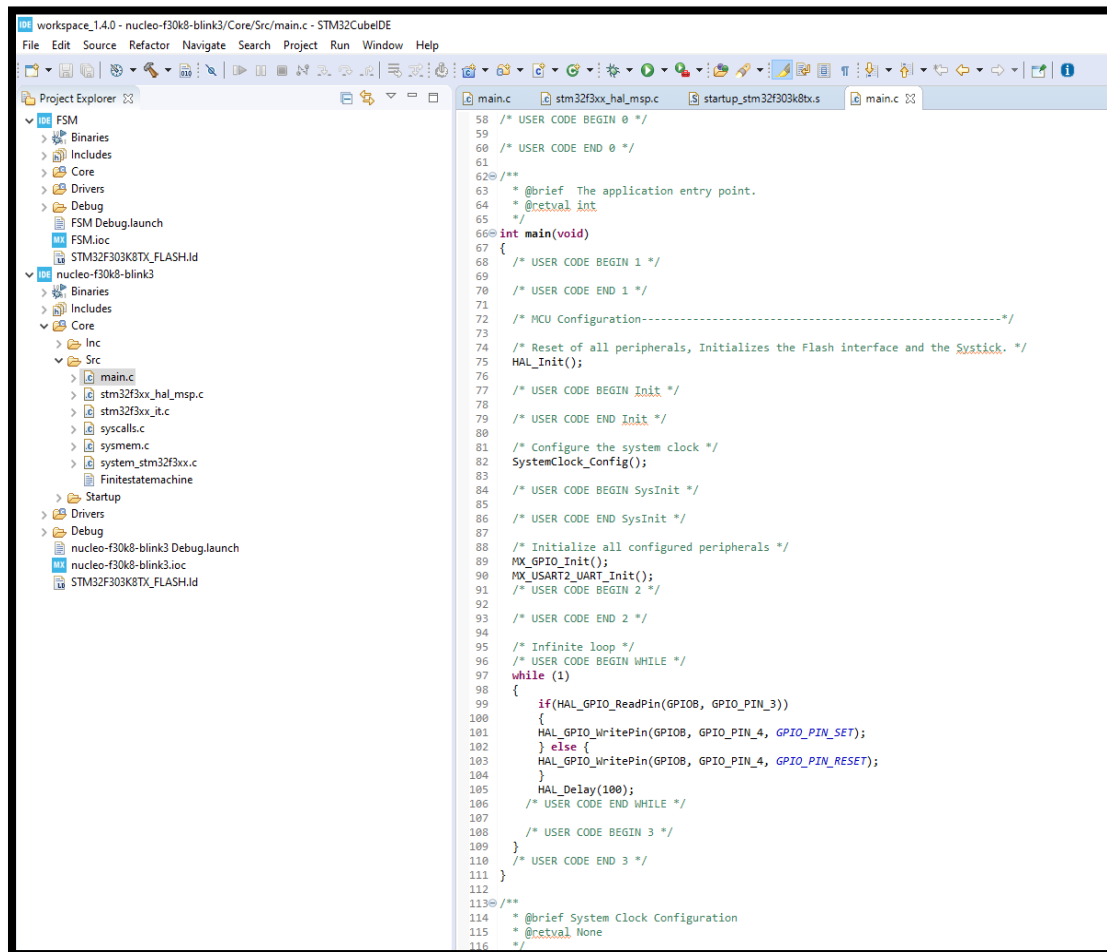


The first step of any STM32Cube project is to configure the pins for basic GPIO.



One of the most important part of this labs was paying attention to the pins that are assigned. Several times throughout this lab I found myself running in circles trying to figure out where my bug was just to find out that a pin was assigned to either the wrong output or name.

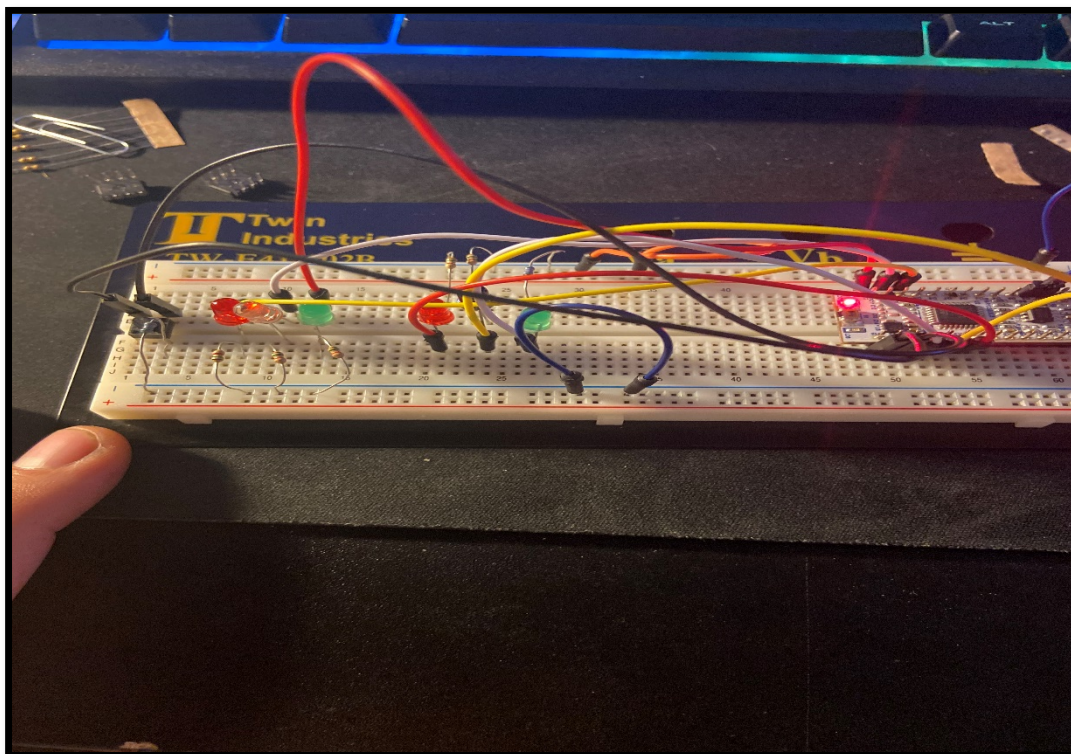
Moving onto the debugger, STM32CubeIDE require one to set the debug options before actually debugging. This was also a very important part of the lab. Debug would be very useful in the rest of the lab.



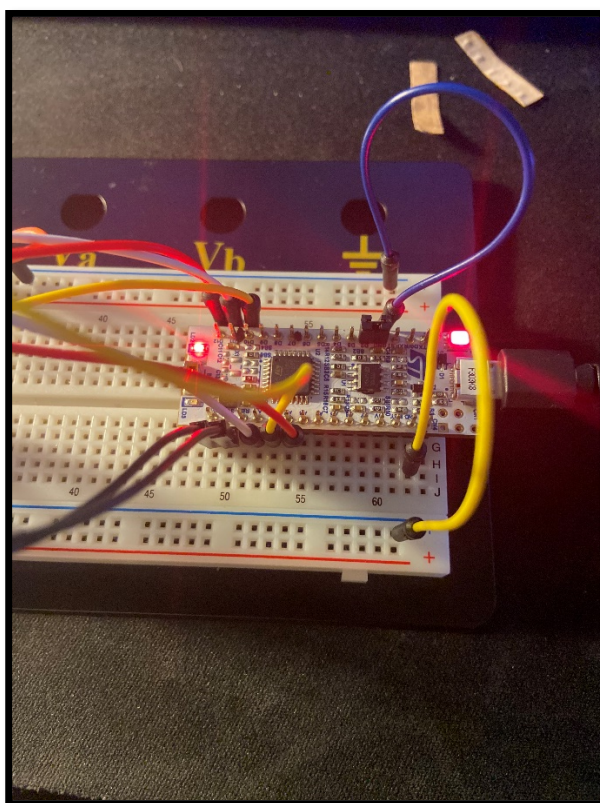
Part 2: Finite State Machines Software Design Pattern

In part 2, we are going to build a Finite State Machines (FSM) that represent a traffic light. To do this part we are going to modify the software and hardware in Section D of Part 1.

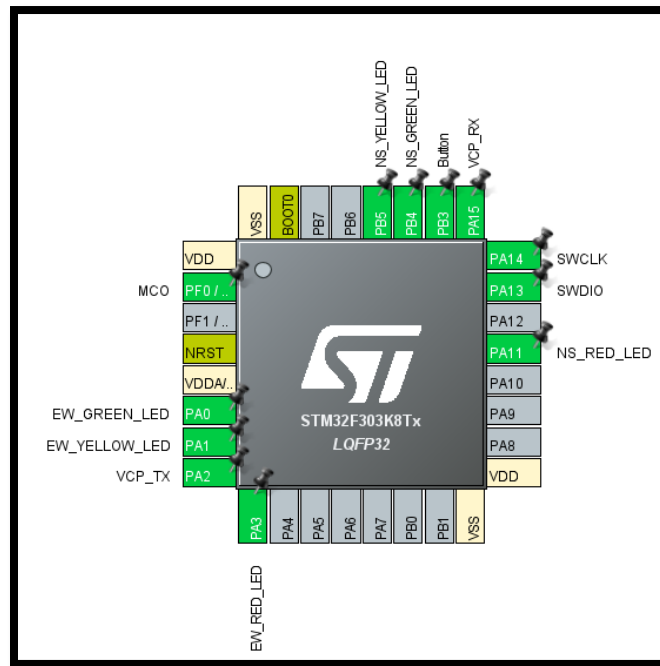
Now that we design of our finite state machine we are going to implement it with our microcontroller and 6 LEDs. For the next part we will need 3 more LEDs combined with Section D of Part 1. Total of 6 LEDs.



Wiring of FSM State machine



Pin Layout with (Button) output



Although the instructions were given to us, I had a very challenging part on the FSM. First was the incorrect color of LEDs. Even though the instructions specified green, yellow and red LED, I found myself having to change the names of several pins in order to match my FSM I built. Second was the problem of the push button. Although never specified that the push button needed to be on the pinout, it made sense that the push pin would be a pin due to the fact that it is controlling the main part of the FSM.

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
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;
    }

    // if(HAL_GPIO_ReadPin(GPIOB, Button_Pin)){
    //     eNextState = TransitionEastWestHandler();
    // }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSISState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }
}

```


Conclusion

This lab was a great introduction on using microcontrollers software. Although several steps of the lab were difficult, I learned a good understanding on pin layouts and configuring a microcontroller on a breadboard.