Lab2: Joysticks and LEDs with STM32CubeMX

Lab Date: 1/30/19

Report Date: 2/2/19

Cameron Stark & Jacob Gattuso

Introduction:

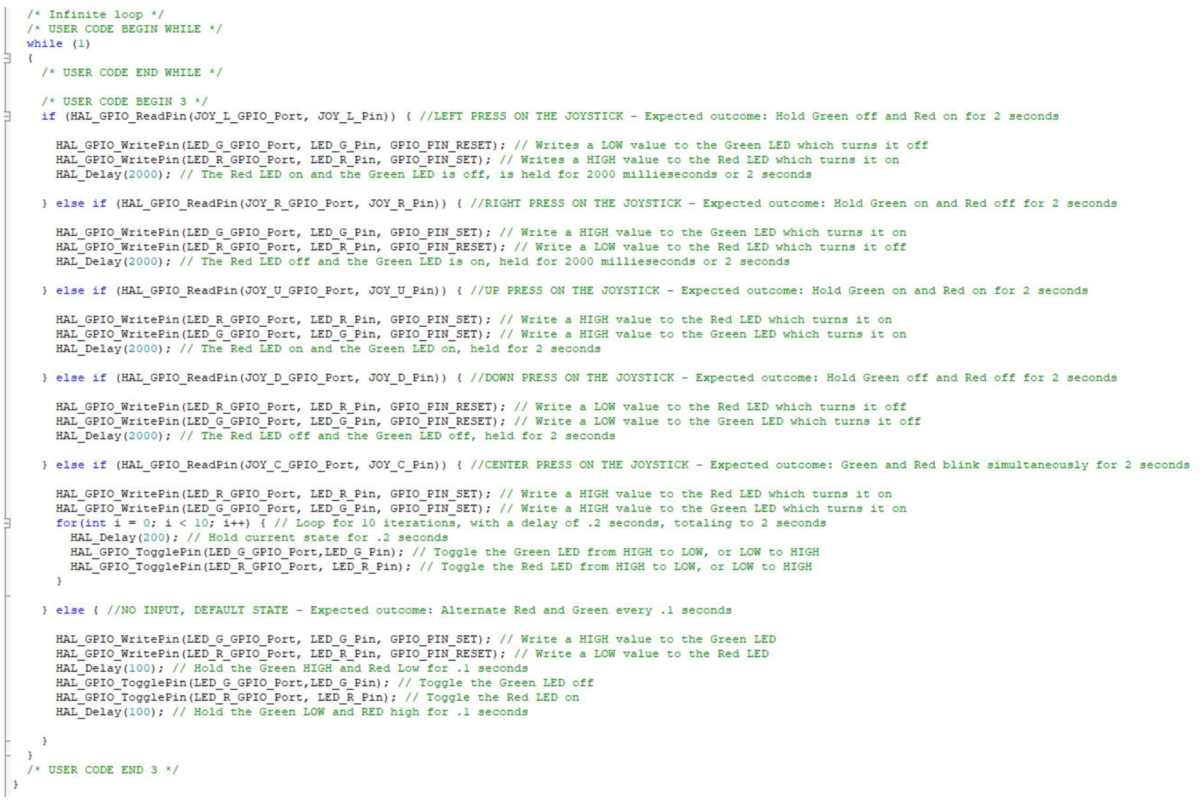
Lab 2 consisted of using the CubeMX software to allocate and assign the necessary pins to take input from the directional joystick for the 5 different inputs being up, down, left, right and center, also to provide output to the red LED and green LED. After selecting, renaming and allocating the pins, their characteristics were modified such as changing the LEDs to pull down and the inputs to neither. These changes allow the pins to work as they are needed for the lab.

Code:

The custom code for the actual functionality of the lab must put in specific sections of the auto generated code provided by the CubeMX program. In the Main.c file there are several commented sections that are defined by the following.

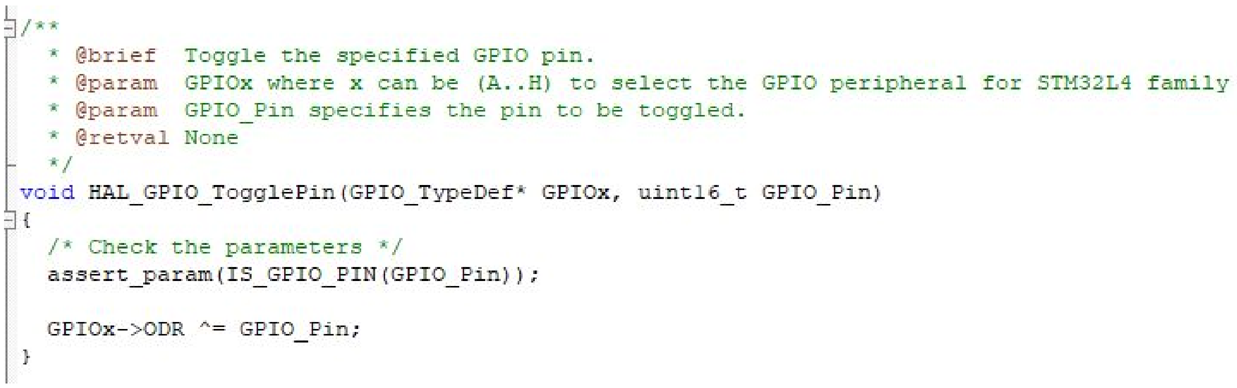
```
/* USER CODE BEGIN # */

/* USER CODE END # */
```

The code for this lab is inserted between the user code section #3, which is contained inside of a while loop which keeps the program running until the device is powered off or reloaded and hasn't started yet. Below is custom code to get the LED and Joystick working as expected.



```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if (HAL_GPIO_ReadPin(JOY_L_GPIO_Port, JOY_L_Pin)) { //LEFT PRESS ON THE JOYSTICK - Expected outcome: Hold Green off and Red on for 2 seconds

      HAL_GPIO_WritePin(LED_G_GPIO_Port, LED_G_Pin, GPIO_PIN_RESET); // Writes a LOW value to the Green LED which turns it off
      HAL_GPIO_WritePin(LED_R_GPIO_Port, LED_R_Pin, GPIO_PIN_SET); // Writes a HIGH value to the Red LED which turns it on
      HAL_Delay(2000); // The Red LED on and the Green LED is off, is held for 2000 millieseconds or 2 seconds

    } else if (HAL_GPIO_ReadPin(JOY_R_GPIO_Port, JOY_R_Pin)) { //RIGHT PRESS ON THE JOYSTICK - Expected outcome: Hold Green on and Red off for 2 seconds

      HAL_GPIO_WritePin(LED_G_GPIO_Port, LED_G_Pin, GPIO_PIN_SET); // Write a HIGH value to the Green LED which turns it on
      HAL_GPIO_WritePin(LED_R_GPIO_Port, LED_R_Pin, GPIO_PIN_RESET); // Write a LOW value to the Red LED which turns it off
      HAL_Delay(2000); // The Red LED off and the Green LED is on, held for 2000 millieseconds or 2 seconds

    } else if (HAL_GPIO_ReadPin(JOY_U_GPIO_Port, JOY_U_Pin)) { //UP PRESS ON THE JOYSTICK - Expected outcome: Hold Green on and Red on for 2 seconds

      HAL_GPIO_WritePin(LED_R_GPIO_Port, LED_R_Pin, GPIO_PIN_SET); // Write a HIGH value to the Red LED which turns it on
      HAL_GPIO_WritePin(LED_G_GPIO_Port, LED_G_Pin, GPIO_PIN_SET); // Write a HIGH value to the Green LED which turns it on
      HAL_Delay(2000); // The Red LED on and the Green LED on, held for 2 seconds

    } else if (HAL_GPIO_ReadPin(JOY_D_GPIO_Port, JOY_D_Pin)) { //DOWN PRESS ON THE JOYSTICK - Expected outcome: Hold Green off and Red off for 2 seconds

      HAL_GPIO_WritePin(LED_R_GPIO_Port, LED_R_Pin, GPIO_PIN_RESET); // Write a LOW value to the Red LED which turns it off
      HAL_GPIO_WritePin(LED_G_GPIO_Port, LED_G_Pin, GPIO_PIN_RESET); // Write a LOW value to the Green LED which turns it off
      HAL_Delay(2000); // The Red LED off and the Green LED off, held for 2 seconds

    } else if (HAL_GPIO_ReadPin(JOY_C_GPIO_Port, JOY_C_Pin)) { //CENTER PRESS ON THE JOYSTICK - Expected outcome: Green and Red blink simultaneously for 2 seconds

      HAL_GPIO_WritePin(LED_R_GPIO_Port, LED_R_Pin, GPIO_PIN_SET); // Write a HIGH value to the Red LED which turns it on
      HAL_GPIO_WritePin(LED_G_GPIO_Port, LED_G_Pin, GPIO_PIN_SET); // Write a HIGH value to the Green LED which turns it on
      for(int i = 0; i < 10; i++) { // Loop for 10 iterations, with a delay of .2 seconds, totaling to 2 seconds
        HAL_Delay(200); // Hold current state for .2 seconds
        HAL_GPIO_TogglePin(LED_G_GPIO_Port,LED_G_Pin); // Toggle the Green LED from HIGH to LOW, or LOW to HIGH
        HAL_GPIO_TogglePin(LED_R_GPIO_Port, LED_R_Pin); // Toggle the Red LED from HIGH to LOW, or LOW to HIGH
      }

    } else { //NO INPUT, DEFAULT STATE - Expected outcome: Alternate Red and Green every .1 seconds

      HAL_GPIO_WritePin(LED_G_GPIO_Port, LED_G_Pin, GPIO_PIN_SET); // Write a HIGH value to the Green LED
      HAL_GPIO_WritePin(LED_R_GPIO_Port, LED_R_Pin, GPIO_PIN_RESET); // Write a LOW value to the Red LED
      HAL_Delay(100); // Hold the Green HIGH and Red Low for .1 seconds
      HAL_GPIO_TogglePin(LED_G_GPIO_Port,LED_G_Pin); // Toggle the Green LED off
      HAL_GPIO_TogglePin(LED_R_GPIO_Port, LED_R_Pin); // Toggle the Red LED on
      HAL_Delay(100); // Hold the Green LOW and RED high for .1 seconds

    }
  }
  /* USER CODE END 3 */
}
```

The above code makes use of some predefined and generated functions from the cubeMX program that are designed to work with our STM board. For example the below code is for the HAL_GPIO_TogglePin function which takes in the current pin and current pin state and flips it.



```
/**
  * @brief  Toggle the specified GPIO pin.
  * @param  GPIOx where x can be (A..H) to select the GPIO peripheral for STM32L4 family
  * @param  GPIO_Pin specifies the pin to be toggled.
  * @retval None
  */
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  /* Check the parameters */
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  GPIOx->ODR ^= GPIO_Pin;
}
```

The above code works by making use of bitwise operators, specifically the XOR operator which outputs a 1 if the values are different and 0 if they are the same

Narrative:

1.) Why do we need to have the “Push-pull” mode for the LEDs? Can we use the “Open-drain” mode?

We use push-pull for the LED because it needs to be turned from high to low, to toggle the LED on and off. No we cannot use Open-Drain because it can go from high to low, but can’t go back to high without a pull up, so it can’t be used in this case.

2.) Why do we need to have the “Pull-down” resistors for the input keys? Can we ignore this resistor for the Center key? Why or why not?

We set to pull-down so that the default value is 0 when not pressed and 1 when pressed. No we can’t ignore the resistor because it makes the output as a high or low, in the case of a pull down the output/input is low.

Results:

From the lab, it was learned how the push-pulls can be used to control LEDs and how to handle input from the board in c program to make the board do actions as a result, also learned how to set up the pins on the board to perform specific and needed tasks.