Lab 1: Introduction to the STM32F407G Discovery Board with GPIO and Rotary Encoders

ECE 4240 B01

October 1, 2025

Dayaveer Gill – 7969885

4.0 - LABORATORY EXPERIMENTS

Part 1:

For part one of this lab we started off with creating our first ever project by following the steps provided in the lab manual. After starting up our first project we used the HAL_GPIO_WritePin function to set and reset the LED3 on our STM board while also adding HAL_Delay in-between both functions and after the reset so we can see it change states, and it becomes a blinking red LED on the board. After observing the red LED blink, we showed the TA. Figure 1 in the appendix shows the code used for this part.

Part 2:

For part 2, we started off by supplying the rotary encoder with 3V and GND via a breadboard using the STM board. Then the Saleae Logic Analyzer was connected to the rotary encoder on the breadboard, and the Saleae Logic Analyzer was also connected to the PC via a USB cable. After connecting all the required probes, the Logic application was opened, and all the parameters were set as required. After everything was set up rotating the rotary encoder produced a waveforms from which we obtained the gray code for each direction. Figure 2 shows the waveforms for clockwise direction.

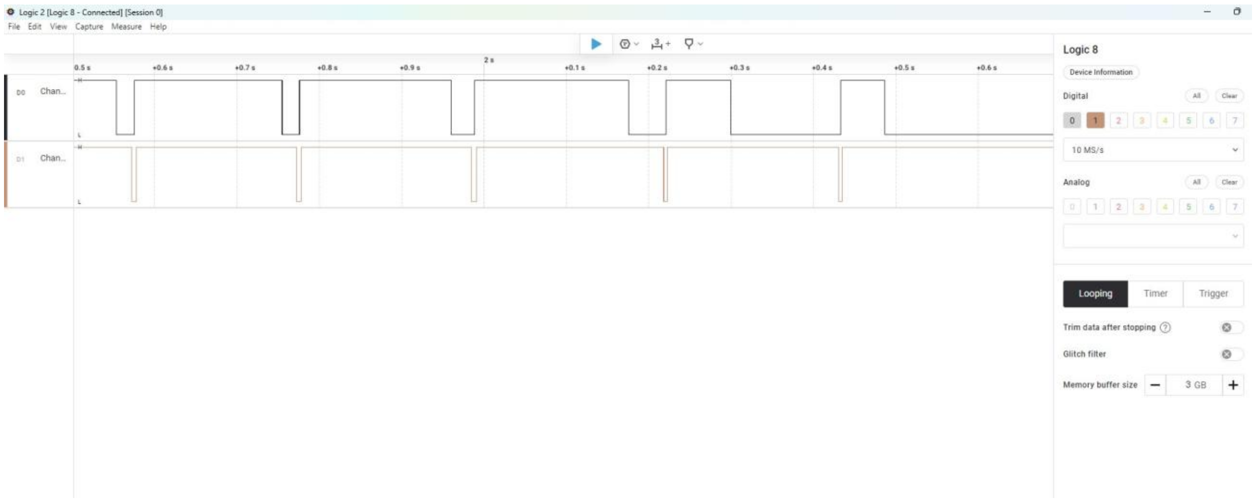


Figure 2. Clockwise direction waveform of rotary encoder.

Part 3:

For part 3 we it does an initial read of the rotary encoder and then it reads it again comparing the previous state with the next one read until the clockwise or counterclockwise pattern is seen, when one is seen then currentLED counter increases or decreases and based on the value of the currentLED it toggles the correct LED.

Figure 3a. 3b. 3c. in the appendix shows the code for this part.

Additional Questions:

1. HAL_Delay prevents multitasking, meaning none of the other code can run during the delay.

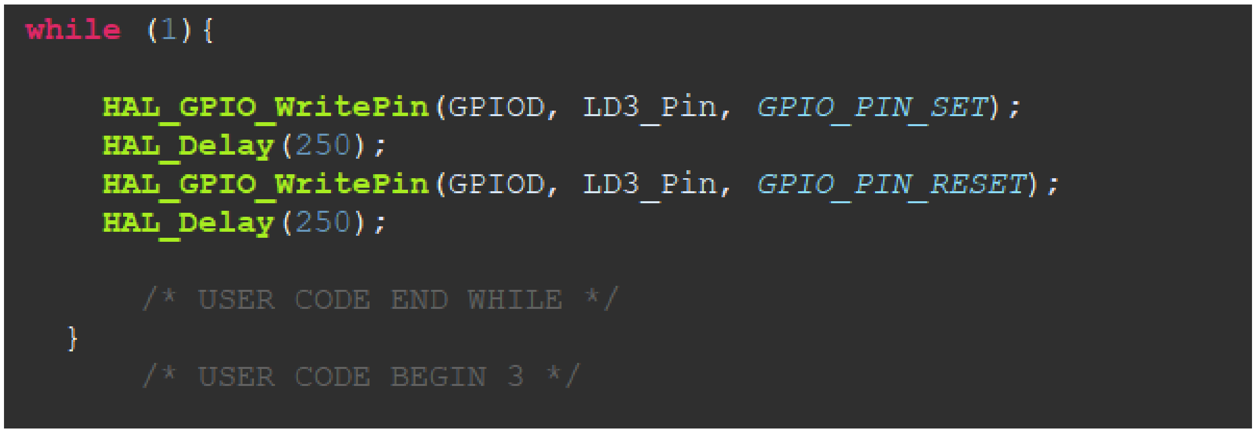
2. a. The glitchy output come from the switch bounce when the rotary encoder doesn't switch cleanly to the next notch.

b. You can make a debounce circuit using a resistor and capacitor, with the rotary encoder output, resistor and STM board input in series and a capacitor connected between the STM board input and ground.

c. One way to debounce in software would be to have sampling at fixed intervals and only stable changes are accepted by checking if the new state stays changed for a fixed number of samples.

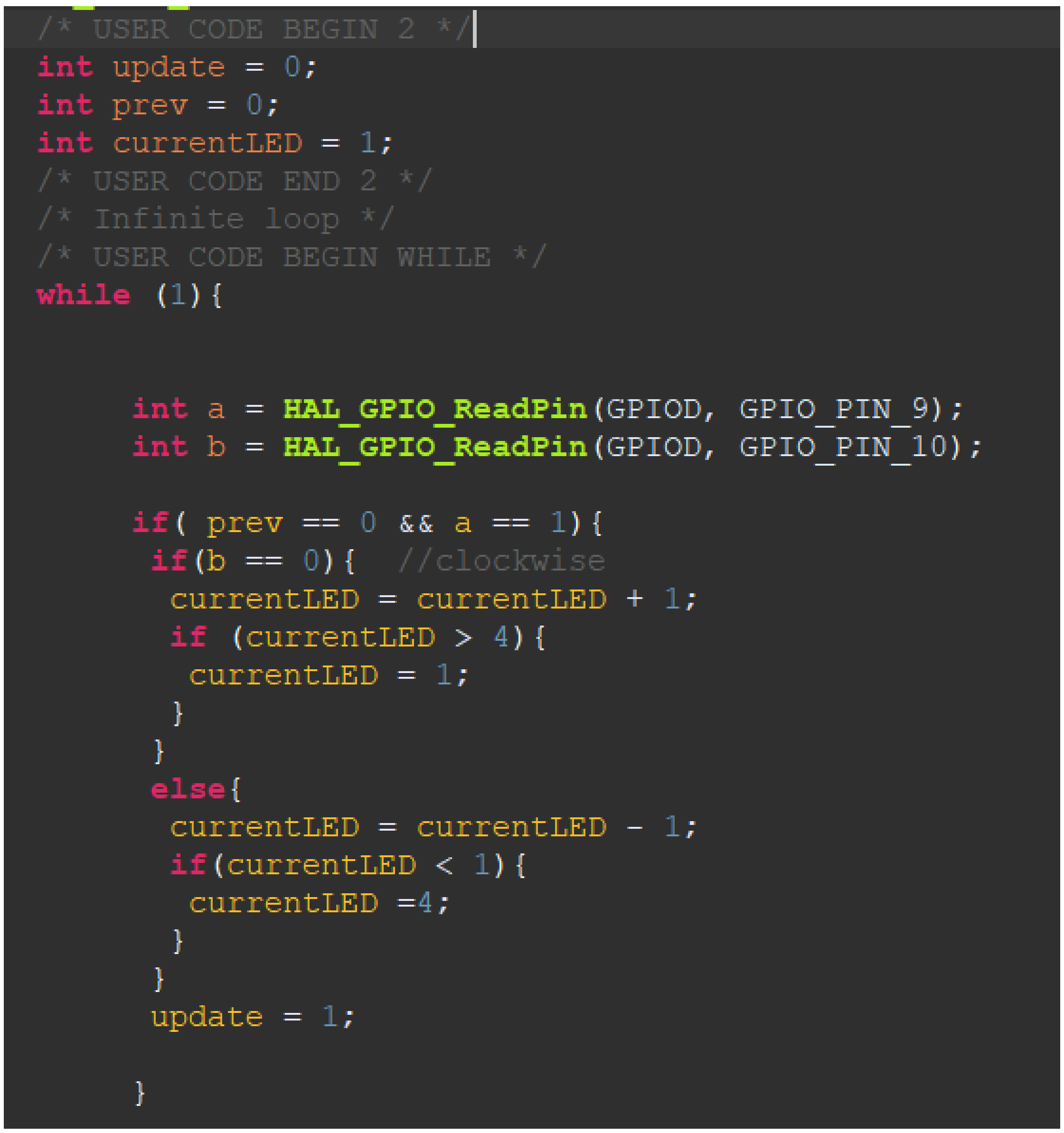
3. General purpose timer would be the right choice for this function to delay in execution in microseconds. To configure the general-purpose timer, we set the prescaler so that 1 timer tick = 1 microsecond. The 1 microsecond is written into auto reload register, and then timer is run in periodic mode.

Appendix:



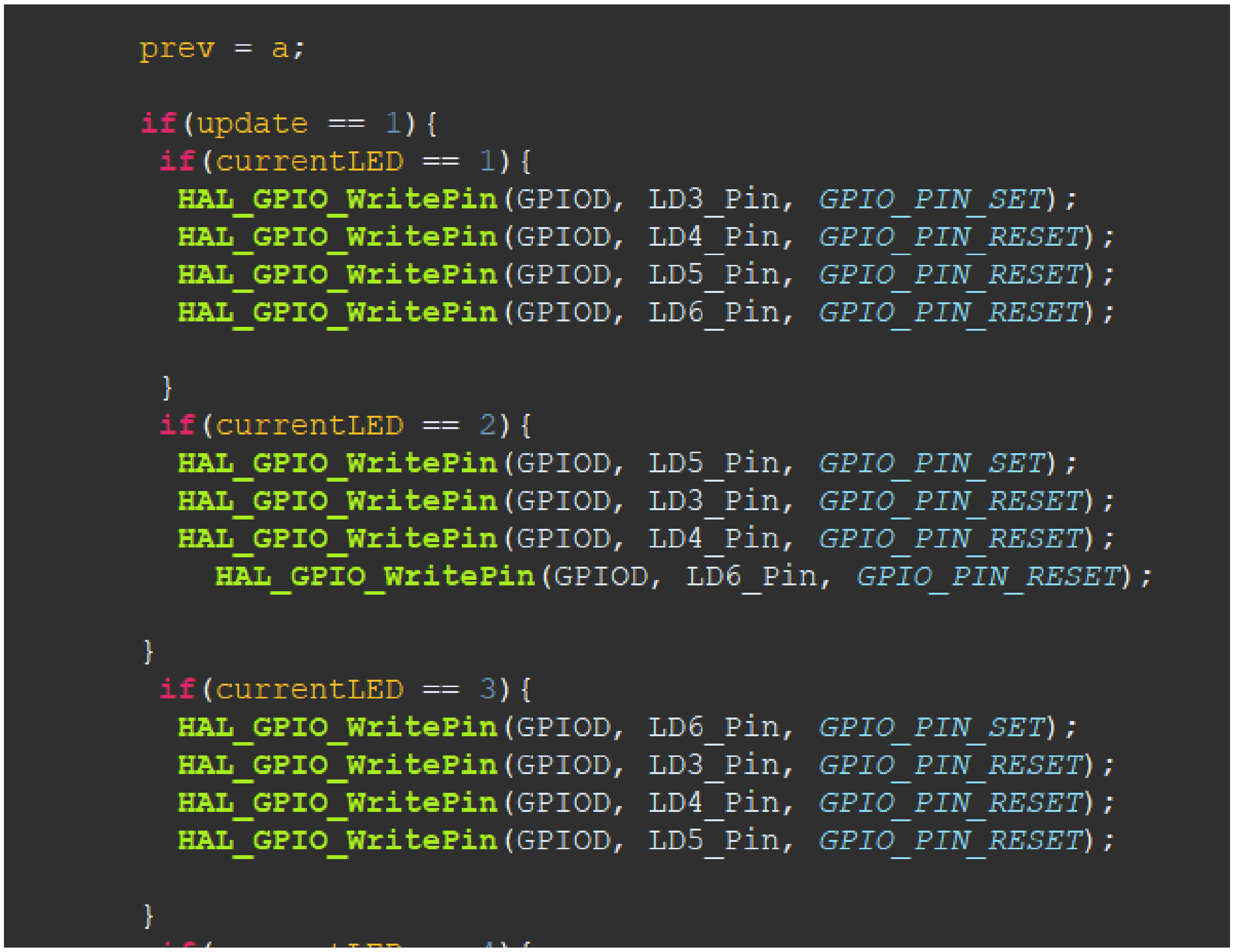
```
while (1){

    HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_SET);
    HAL_Delay(250);
    HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
    HAL_Delay(250);

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

Figure 1. Code for part 1.



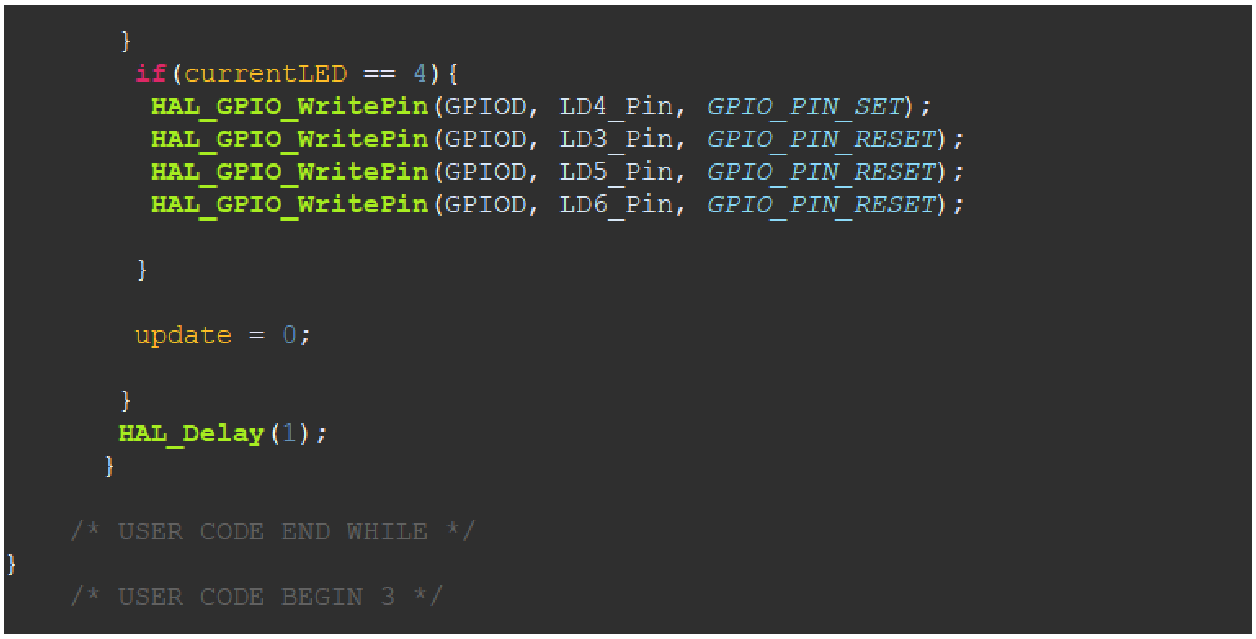
```
/* USER CODE BEGIN 2 */
int update = 0;
int prev = 0;
int currentLED = 1;
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1){


     int a = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_9);
     int b = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_10);

     if( prev == 0 && a == 1){
      if(b == 0){  //clockwise
       currentLED = currentLED + 1;
       if (currentLED > 4){
        currentLED = 1;
       }
      }
      else{
       currentLED = currentLED - 1;
       if(currentLED < 1){
        currentLED =4;
       }
      }
      update = 1;

     }
```

Figure 3a. Code for part 3.



```
      prev = a;

      if(update == 1){
       if(currentLED == 1){
        HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_RESET);

       }
       if(currentLED == 2){
        HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);
          HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_RESET);

      }
       if(currentLED == 3){
        HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);

      }
```

Figure 3b. Code for part 3.



```
       }
        if(currentLED == 4){
         HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_RESET);

        }

        update = 0;

       }
       HAL_Delay(1);
      }

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

Figure 3c. Code for part 3.