Instrumentation*

*Note: Instrumentation — Homework 3

1st Asra Mehrolhassani
Electrical Engineering Dept.
University of Tehran
StudentID: 810800024
Asra.Mehr@ut.ac.ir

***Abstract*—The current document contains the solutions to the third assignment prepared for the class Instrumentation, 8106126-01 . Knowledge of Instrumentation is required for deep understanding of the materials presented in this report.**

***Index Terms*—Instrumentation,LS, RLS**

I. Introduction

A. List of Problems

1) Traffic Light
2) Timer
3) ADC
4) PWM Generation

The STM32 board used for this assignment is STM32F103T6 and clock frequency is set to be 20MHz.

II. Problem 1— Traffic Light

A. Configuration

1) - Description: 3 LEDs are used with the corresponding pins defined as GPIO inputs, along with a button, the pin for which is defined as GPIO output. The LED pins are stored in an array for better access, a variable (i) is defined which increases whenever a button is pressed, the remainder of which is used to choose the appropriate LED to light up.

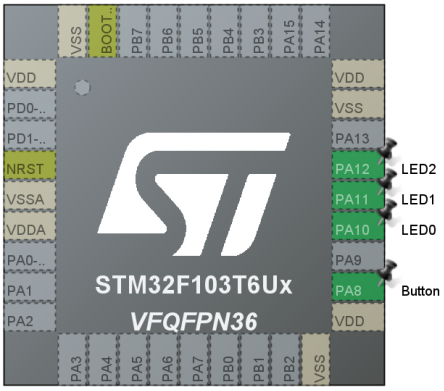

Fig. 1. Settings, button is defined as a pull-up button.

2) - Code:

```c
int main(void)
{
  /* USER CODE BEGIN 1 */
int i = -1;
uint16_t LED_Pin[3]={LED0_Pin,LED1_Pin,LED2_Pin};
...

  while (1)
  {
      if (HAL_GPIO_ReadPin(Button_GPIO_Port, Button_Pin) == 0){

          if (i != -1)
              HAL_GPIO_WritePin(GPIOA, LED_Pin[i], GPIO_PIN_RESET);
          i++;
          i = i%3;
          HAL_GPIO_WritePin(GPIOA, LED_Pin[i], GPIO_PIN_SET);


      }
      HAL_Delay(300);
    /* USER CODE END WHILE */

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

B. Simulation in Proteus

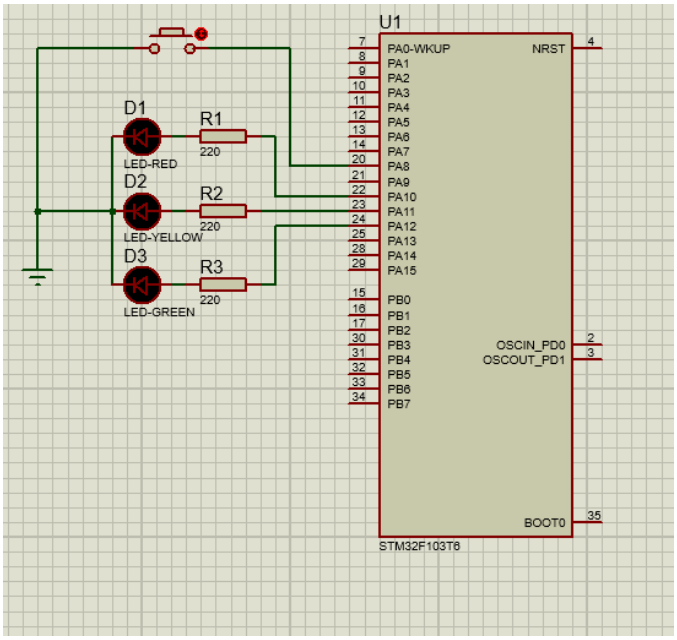

Fig. 2. Connections

1) - Simulation:

2) - Results:

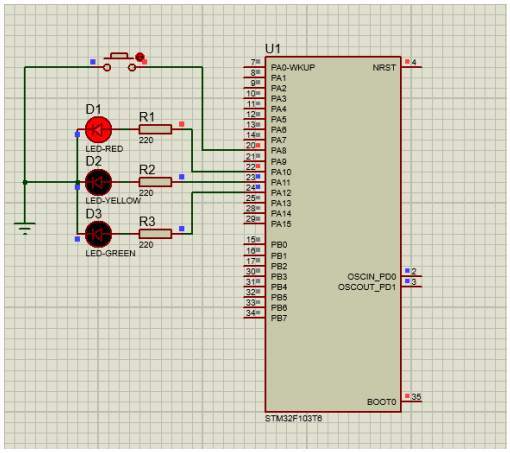

Fig. 3. Traffic Light—By pressing the button, the previous LED turns off and the next turns on.

III. Problem 2— Timer

A. Configuration

1) - Description: : The buttons are defined as GPIO pull-down external interrupts, and the other 6 pins are defined for the LCD. Note that lcd.h is manually added to the Inc sub folder under the core under the Core folder.

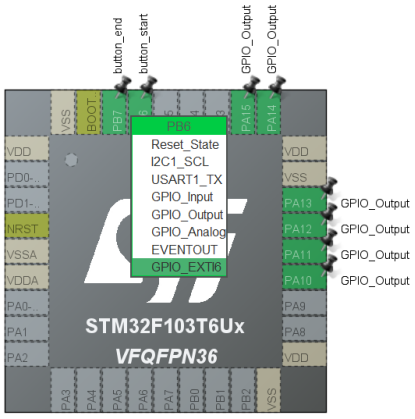

Fig. 4. Pins and Configurations

It is worth mentioning that the timer parameters are defined as bellow (in order to count every second): Notice how by substituting the above numbers below we get 1 HZ for the output frequency:

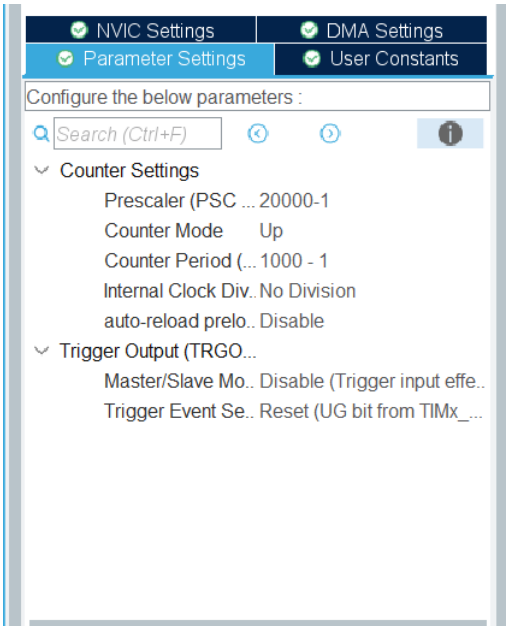

Fig. 5. Tim2 Parameter Settings

$$F_{Output} = \frac{F_{timer}}{(PSC+1)(ARR+1)}$$

```c
int x = 0; //second
int y = 0; //minute
char s[4];
char m[4];

int main(void){
    while (1){
  if (x == 60){
          y++;
          x=1;
          lcd_clear();
   }
        itoa(x,s,10);
        itoa(y,m,10);
        lcd_puts(0,0,m);
        lcd_puts(0,2,(int8_t*)":");
        lcd_puts(0,3, s);
        lcd_puts(1,0,(int8_t*)"810800024");
        HAL_Delay(10);
  }
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
{
    UNUSED(htim);
    if (htim -> Instance == TIM2){
        x=x+1;
    }
}
```

2) - Code:

B. Simulation in Proteus

1) - Simulation: :

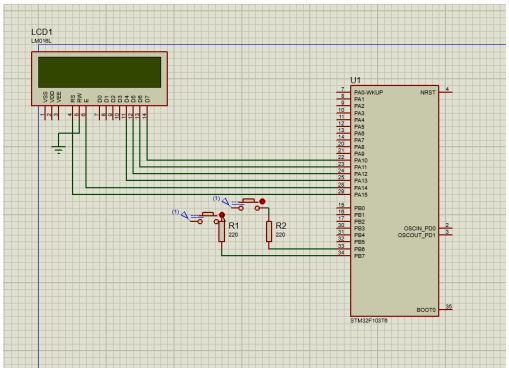

Fig. 6. Timer, Connections

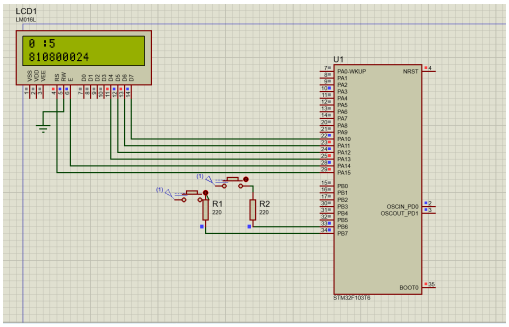

Fig. 7. Timer Results

2) - Results::

IV. Problem 3—ADC

A. Configurations

1) - Description: : ADC is used to convert analog signal to digital signal. PA10 to PA15 are defined as GPIO output pins are aimed for the LCD.

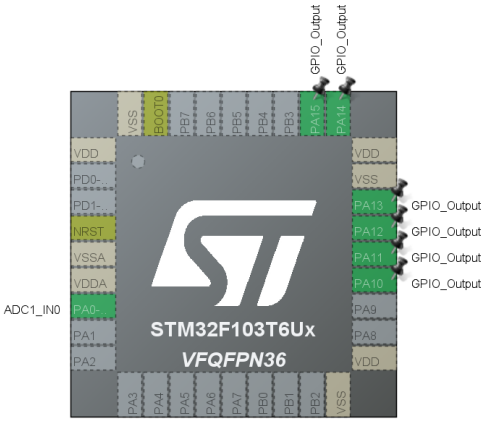

Fig. 8. ADC - Connections and Pins

2) - Code::

B. Simulation in Proteus

1) - Simulation: :

```c
main(void){
MX_ADC1_Init();
  /* USER CODE BEGIN 2 */
  lcd_init();
  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
     if (abs(val - 1000) < 100|| abs(val - 100) < 20 || abs(val -10 )<3){
            lcd_clear();
        }

      HAL_ADC_Start(&hadc1);
      HAL_ADC_PollForConversion(&hadc1, 20);
      val = HAL_ADC_GetValue(&hadc1);
      itoa(val,s,10);
      lcd_puts(0,0,s);
      lcd_puts(1,0, (uint8_t*)"810800024");
      HAL_Delay(1);
  }
  /* USER CODE END 3 */
}
```

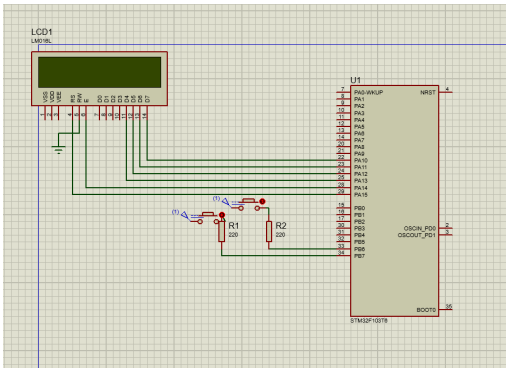

Fig. 9. ADC - Connections

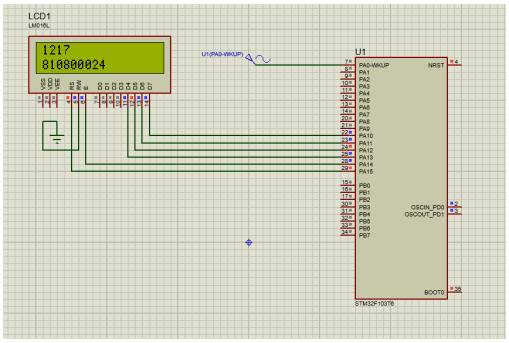

Fig. 10. ADC - Results

2) - Results:: Notice how since the input voltage is set at 1volts, the sine amplitude is between -4096/5 and 4096/5 and since we've chosen the offset to be 1, the value is between 0 and 1638 in digital.

C. Problem 4—PWM Generation

D. Configurations

1) - Description: : Notice how the input voltage is 2 volts and thus digital value obtained by ADC is maximum 0.4*4096 = 1638. Timer's parameters are defined as below to get pwm with 100Hz frequency:

2) - Code::

E. Simulation in Proteus

1) - Simulation: :

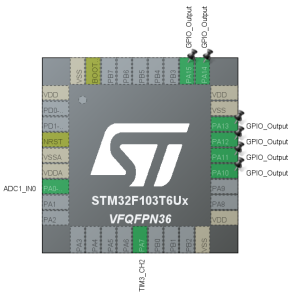

Fig. 11. PWM-Pins and configuration

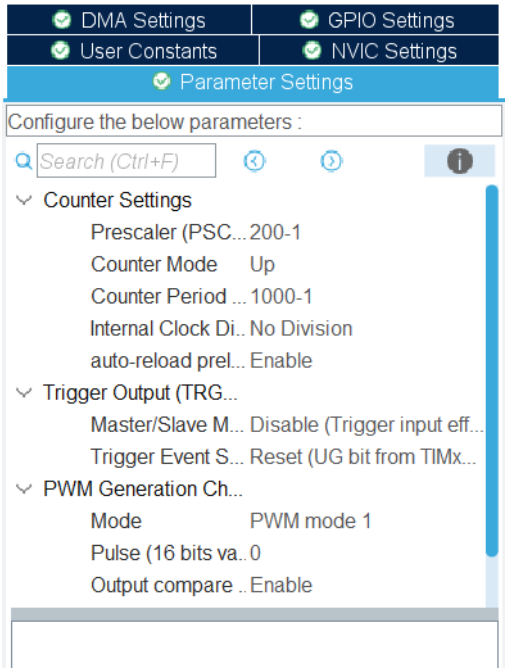

Fig. 12. Tim3, param settings

2) - Results: Notice how the LCD is only there to help debug the process.

1) Duty Cycle : 40%

2) duty cycle 10 % or ... based on the potentiometer

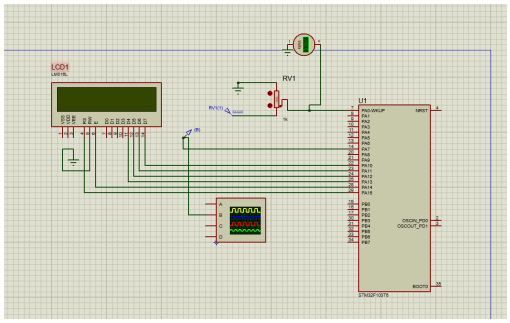

Fig. 13. Connections

```
main(void){
M/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ADC1_Init();
MX_TIM3_Init();
/* USER CODE BEGIN 2 */
lcd_init();
TIM3-> CCR2 = 400;
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
/* USER CODE END 2 */

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

  /* USER CODE BEGIN 3 */
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 20);
    pot_val = HAL_ADC_GetValue(&hadc1);

    dc = 10000*pot_val/1638;
    lcd_clear();
    itoa(dc, s,10);
    lcd_puts(0,0,s);
    TIM3-> CCR2 =dc ;
    HAL_Delay(100);
    //1638 max for 2 volts
    //0 min
}
}
```

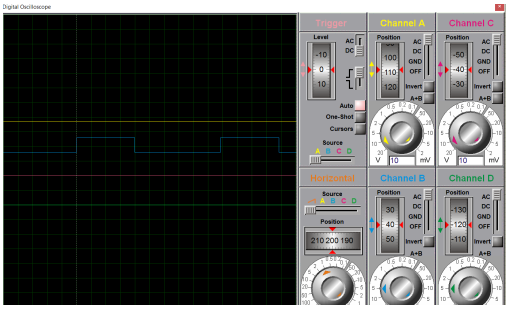

Fig. 14. Duty cycle 40 %

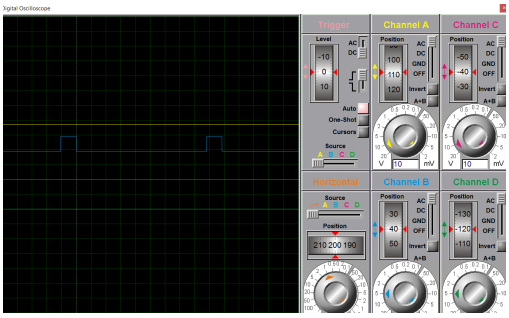

Fig. 15. Duty cycle 10 % based on potentiometer