LAB: Timer & PWM

LAB: PWM – Servo motor and DC motor

Date: 2023-10-19

Author/Partner: NohYunKi

RC motor Demo Video: https://youtu.be/_n2eSJBnQZc

DC motor Demo Video:https://youtu.be/5q4cigOydmk

Introduction

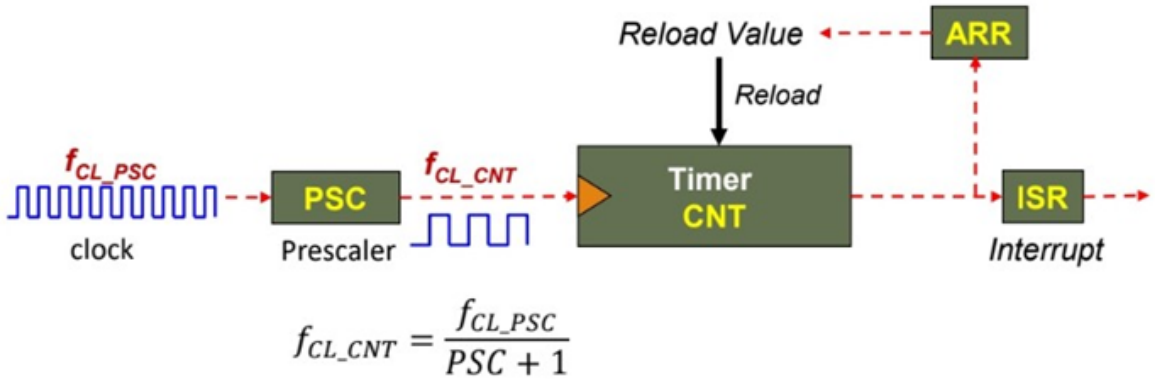

Figure 1. Timer's flow

Timer generates signal which is occurred automatically and repeatedly. So, I will use this signal to work RC & DC motors. As you can see above Figure 1, the original frequency is f_CL_PSC(Clock frequency). Through dividing this original frequency by "PSC + 1", we can modify original frequency value into count timer's frequency.(f_CL_CNT). After this process, by counting each steps several times which are determined by ARR value, we can obtain final period of signal. At the problem 1, RC motor will be worked by using Timer signal and at the problem 2, DC motor will be worked by using Timer & PWM signal.

Requirement

Hardware

- MCE
  - NUCLEO-F411RE
- Actuator/Sensor/Others:
  - 3 LEDs and load resistance
  - RC Servo Motor (SG90)
  - DC motor (5V)
  - DC motor driver(LS9110s)
  - breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem1: RC servo motor

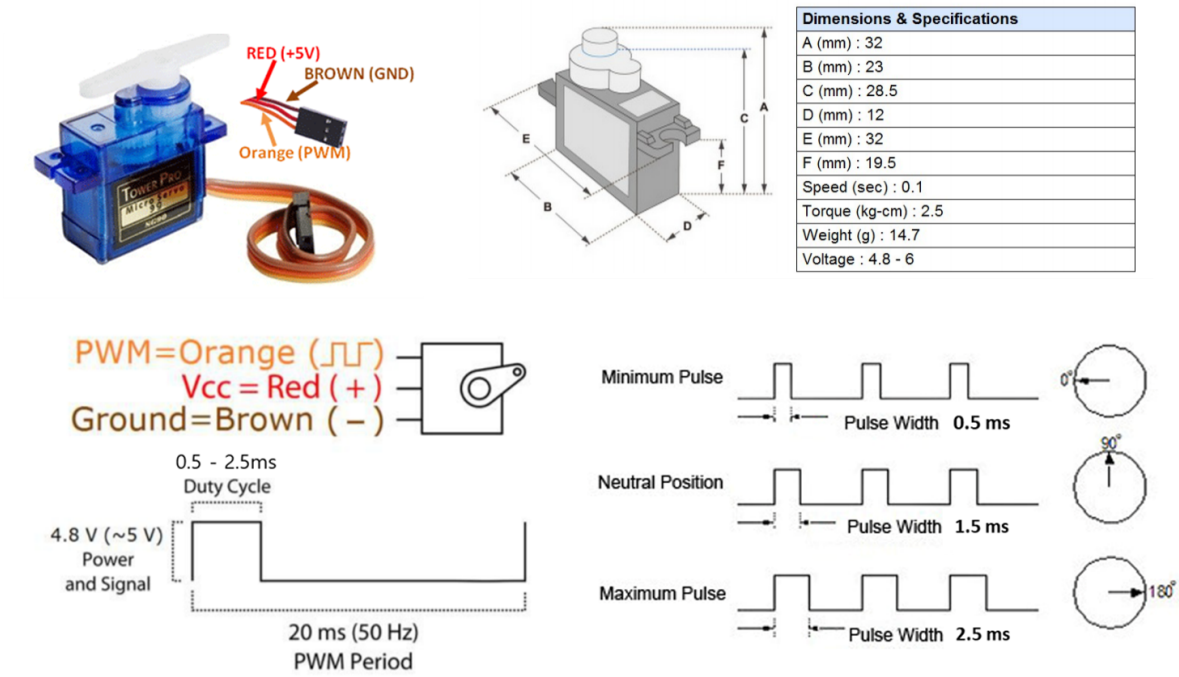

Figure 2. RC_Servo_Motor

Servo Motor's rotation angle is approximately from 0 to 180 degree and this angle value is determined by PWM's period. When PWM's period is 0.5ms, it's angle becomes 0 degree and when PWM's period is 2.5ms, it's angle becomes 180 degree. So, if PWM's period changes between 0.5ms and 2.5ms, it's angle will be changed between 0 degree and 180 degree. In this reason, to change angle from 0° to 180° with a step of 10° every 500ms, PWM's duty ratio must be changed according this formula.**[duty ratio = (0.5ms/20ms) x (i x 4/17.0 + 1), i = 0 ~ 17]**.

Configuration

ype	Port - Pin	Configuration
Button	Digital In (PC13)	Pull-Up
PWM Pin	AF (PA1)	Push-Pull, Pull-Up, Fast
PWM Timer	TIM2_CH2 (PA1)	TIM2 (PWM) period: 20msec, Duty ratio: 0.5~2.5msec
Timer Interrupt	TIM3	TIM3 Period: 1msec, Timer Interrupt of 500 msec

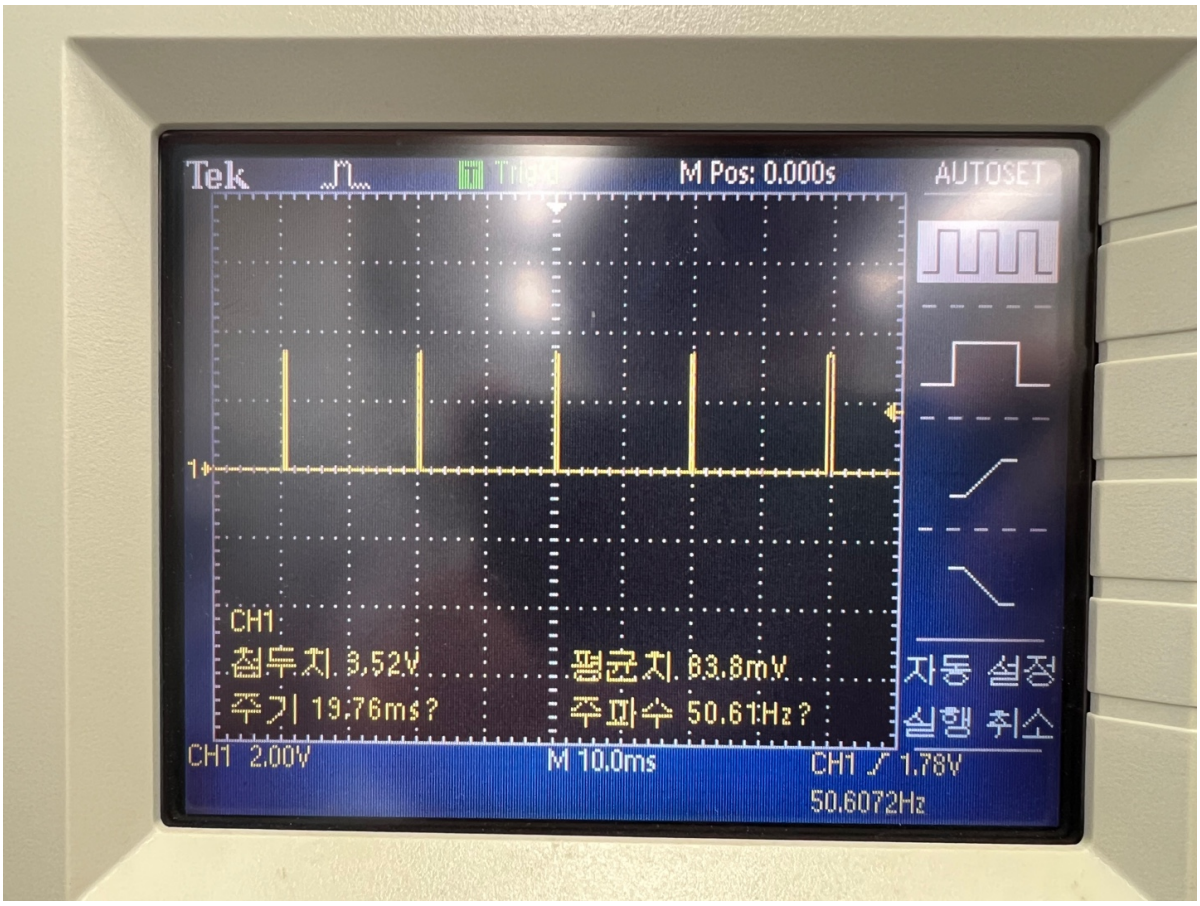

Figure 3. Oscilloscope for PWM's frequency(50Hz)

Circuit Diagram

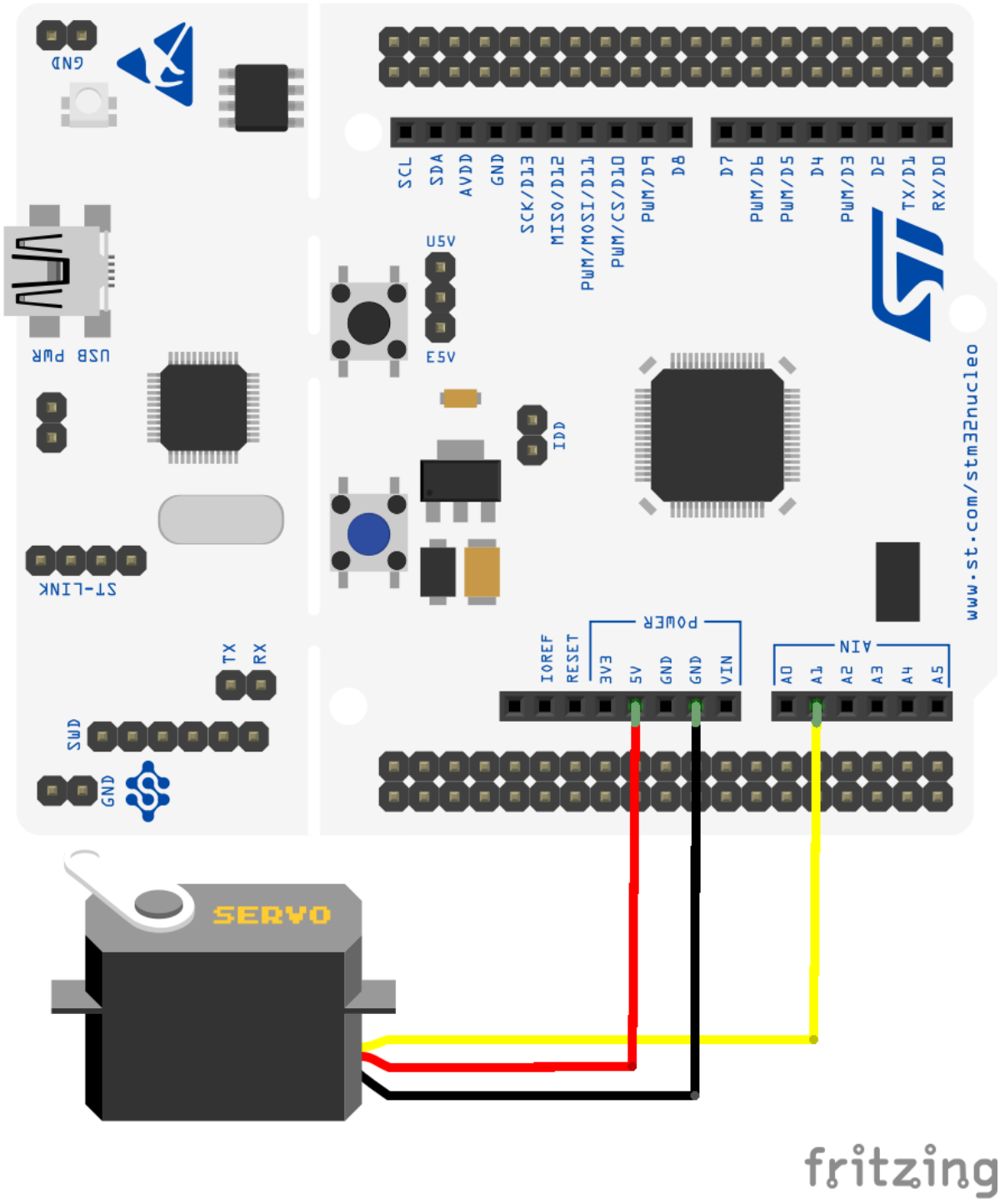

Figure 4. Servo Motor Circuit

Discussion

1. Derive a simple logic to calculate CCR and ARR values to generate x[Hz] and y[%] duty ratio of PWM. How can you read the values of input clock frequency and PSC?

- PSC value[Clock frequency = 84Mhz(PLL) or 16Mhz(HSI)]

$$Timer\,frequency[Hz] = Clock\,frequency[Hz]/(PSC + 1)$$

- ARR value

$$1/UEV\,frequency[Hz] = (1 + ARR)/Timer\,frequency[Hz]$$

- CCR value

$$Dutyratio = CCR/(ARR+1)$$

Frequency of PWM means Update Event frequency of PWM's timer. So, to generate x[Hz] frequency of PWM, we have to know frequency of Clock we are using and we have to choose PSC and ARR values. In formula, this method will be performed like this.

$$ARR = Timerfrequency/x - 1$$

In addition to this, if we want to generate y[%] duty ratio of PWM, formula would be derived like this.

$$CCR = y*(ARR+1) = y*(Timerfrequency/x) = (Clockfrequency*y)/(x*(PSC+1))$$

If we want to know Clock frequency(input clock frequency) value, we have to know first Clock frequency from whether this clock is PLL or HSI. In addition to this, because Clock frequency(input clock frequency) and PSC values are related with Timer frequency value, if we want to know PSC value, we have to use above first formula.

2. What is **the smallest and highest PWM frequency** that can be generated for Q1?

Timer frequency is step value of PWM's Updated Event frequency(50Hz). So, PWM's Timer frequency is from 50Hz to 50Hz/(ARR + 1).

$$50[Hz] >= PWM'sTimerfrequecny >= 50/(ARR+1)[Hz]$$

Code

- main

```
/*----------------------------------------------
Class: 23_2_Embeded_Controller
LAB: LAB_PWM_RCmotor
Name: NohYunKi
----------------------------------------------*/

#include "stm32f411xe.h"
#include "math.h"

// #include "ecSTM32F411.h"
#include "ecPinNames.h"
#include "ecGPIO.h"
#include "ecSysTick.h"
#include "ecEXTI.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"   // ecPWM2.h

// Definition Button Pin & PWM Port, Pin
#define PWM_PIN PA_1

void setup(void);
void EXTI15_10_IRQHandler(void);
void TIM3_IRQHandler(void);
```

```
int cnt = 0;

int main(void) {

  setup();

    while(1){}
}


// Initialiization
void setup(void) {
     // System clock setting
   RCC_PLL_init();

     // Pin setting
   GPIO_init(GPIOA, 1, AF);
     GPIO_init(GPIOA, LED_PIN, OUTPUT);
   GPIO_init(GPIOC, BUTTON_PIN, INPUT);

     // Timer setting
   TIM_init(TIM3, 500); // TIMER period = 500ms
     TIM3->DIER |= 1;   // Update Interrupt Enabled
     NVIC_EnableIRQ(TIM3_IRQn); // TIM3's interrupt request enabled
     NVIC_SetPriority(TIM3_IRQn, 2); // set TIM's interrupt priority

     // External Interrupt setting
     EXTI_init(GPIOC, 13, FALL, 0); // set C_port's 13_pin as signal of EXTI
     NVIC_EnableIRQ(EXTI15_10_IRQn); // enable request interrupt
     NVIC_SetPriority(EXTI15_10_IRQn, 3); // set priority

     // PWM setting
   PWM_init(PWM_PIN); // set Port A's 1_pin as PWM's output pin
   PWM_period(PWM_PIN, 20); // 20 msec PWM period --> set PSC, ARR value to make
PWM_period as 20ms
}

void TIM3_IRQHandler(void){
    if((TIM3->SR & TIM_SR_UIF) == 1){
        // 0.5ms = 0dgree, 1.5ms = 90degree, 2.5ms = 180degree
        PWM_duty(PWM_PIN, (float)0.025*(cnt*4/17.0 + 1));
        cnt ++;
        // count
        if(cnt > 17) cnt = 0;
        // clear by writing 0
        TIM3->SR &= ~ TIM_SR_UIF;
    }
}

void EXTI15_10_IRQHandler(void) {
   if (is_pending_EXTI(BUTTON_PIN)) {
      PWM_duty(PWM_PIN, (float)0.025);
          cnt = 0;
          clear_pending_EXTI(BUTTON_PIN);
   }
```

```
}
```

Results

Link: https://youtu.be/_n2eSJBnQZc

Problem 2: DC motor

Configuration

Function	Port - Pin	Configuration
Button	Digital In (PC13)	Pull-Up
Direction Pin	Digital Out (PC2)	Push-Pull
PWM Pin	AF (PA0)	Push-Pull, Pull-Up, Fast
PWM Timer	TIM2_CH1 (PA0)	TIM2 (PWM) period: **1msec (1kHz)**
Timer Interrupt	TIM3	TIM3 Period: 1msec, Timer Interrupt of 500 msec

Circuit Diagram

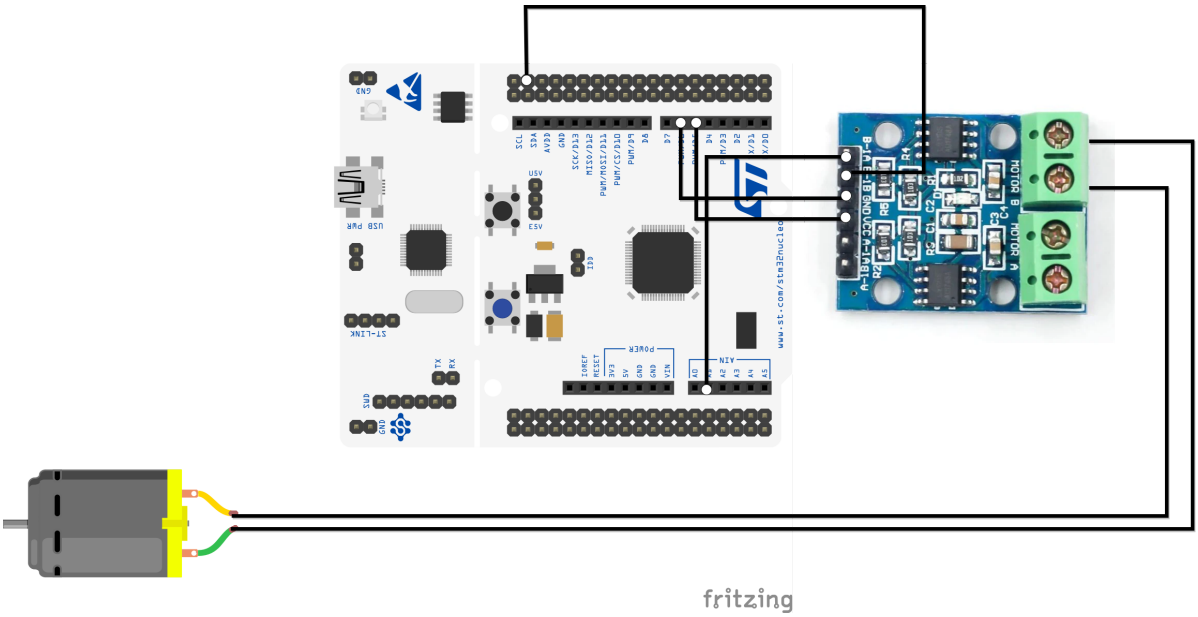

Figure 5. DC motor & Motor Driver Circuit

Code

```
/*----------------------------------------------
Class: 23_2_Embeded_Controller
LAB: LAB_PWM_DCmotor
Name: NohYunKi
----------------------------------------------*/

#include "stm32f411xe.h"
#include "math.h"

// #include "ecSTM32F411.h"
```

```
#include "ecPinNames.h"
#include "ecGPIO.h"
#include "ecSysTick.h"
#include "ecEXTI.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"   // ecPWM2.h

// Definition Button Pin & PWM Port, Pin
#define PWM_PIN PA_1

void setup(void);
void EXTI15_10_IRQHandler(void);
void TIM3_IRQHandler(void);

int cnt = 0;
float ratio = 0.75;
int pause = 0;
int button_signal = 0;

int main(void) {

  setup();

    while(1){}
}


// Initialiization
void setup(void) {
    // System clock setting: frequency = 84MHz
   RCC_PLL_init();

     // Button pin setting
     GPIO_init(GPIOC, BUTTON_PIN, INPUT);
     GPIO_pupd(GPIOC, BUTTON_PIN,EC_PU);
     // Direction pin setting
   GPIO_init(GPIOC, 2, OUTPUT);
     GPIO_otype(GPIOC,2,EC_PUSH_PULL);
     // PWM pin setting
     GPIO_init(GPIOA, PA_0, AF);
     GPIO_otype(GPIOA, PA_0,EC_PUSH_PULL);
     GPIO_pupd(GPIOA, PA_0, EC_PU);
     GPIO_ospeed(GPIOA, PA_0,EC_FAST);

     // Timer setting
   TIM_init(TIM3, 1); // TIMER period = 1ms
     TIM3->DIER |= 1;   // Update Interrupt Enabled
     NVIC_EnableIRQ(TIM3_IRQn); // TIM3's interrupt request enabled
     NVIC_SetPriority(TIM3_IRQn, 2); // set TIM's interrupt priority

     // External Interrupt setting
     EXTI_init(GPIOC, 13, FALL, 0); // set C_port's 13_pin as signal of EXTI
     NVIC_EnableIRQ(EXTI15_10_IRQn); // enable request interrupt
     NVIC_SetPriority(EXTI15_10_IRQn, 3); // set priority
```

```
     // PWM setting
   PWM_init(PWM_PIN); // set Port A's 1_pin as PWM's output pin
   PWM_period(PWM_PIN, 1); // PWM(PA0-->TIM2_CH1) period = 1ms

     // ?? even use GPIO_write to set 1 value into PC_2 pin, it's value is 0
     GPIO_write(GPIOC, 2, 0);
}

void TIM3_IRQHandler(void){
    if((TIM3->SR & TIM_SR_UIF) == 1){
        // speed rate: 100% or 30% (if too low[ex.25%], DC motor didn't rotate)
        PWM_duty(PWM_PIN, ratio*pause);
        //PWM_duty(PWM_PIN, 0);
        cnt ++;
        // count 2s to change speed rate per 2 seconds
        if(cnt > 1999)
        {
            cnt = 0;
            if(ratio == (float)0.75)  ratio = (float)0.25;
            else if(ratio == (float)0.25) ratio = (float)0.75;
        }
        // clear by writing 0
        TIM3->SR &= ~ TIM_SR_UIF;
    }
}

// Button external interrupt
void EXTI15_10_IRQHandler(void) {
   if (is_pending_EXTI(BUTTON_PIN)) {
          if(GPIO_read(GPIOC,2) == 0) pause ^= 1;
          clear_pending_EXTI(BUTTON_PIN);
   }
}
```

Results

Link: https://youtu.be/5q4cigOydmk