LAB: Stepper Motor

Date: 2022-11-02

Author/Partner: 21800447 Jeahyun Oh / 21800805 SeungEung Hwang

Github: https://github.com/Ohjeahyun1/EC-jeahyun-447.git

Demo Video: https://youtu.be/qqZLf99kT1A

Introduction

In this lab, we will learn how to drive a stepper motor with digital output of GPIOs of MCU. You will use a FSM to design the algorithm for stepper motor control.

You must submit

- LAB Report (*.md & *.pdf)
- Zip source files(main*.c, ecRCC.h, ecGPIO.h, ecSysTick.c etc...).
  - Only the source files. Do not submit project files

Requirement

Hardware

- MCU
  - NUCLEO-F401RE
- Actuator:
  - 3Stepper Motor 28BYJ-48
- Sensor
  - Button(B1)
- Others
  - breadboard
  - Motor Driver ULN2003

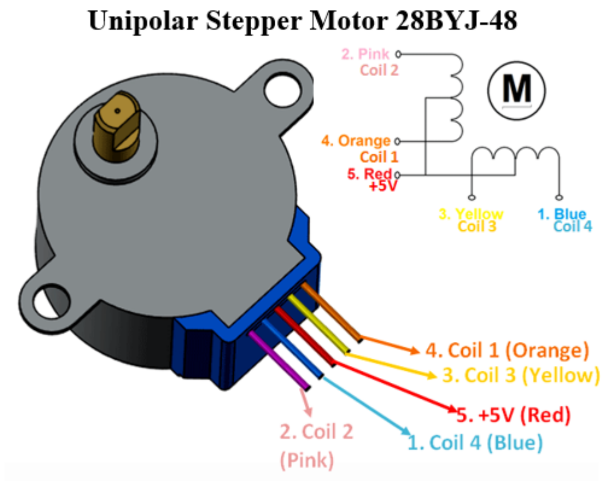
Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Stepper Motor

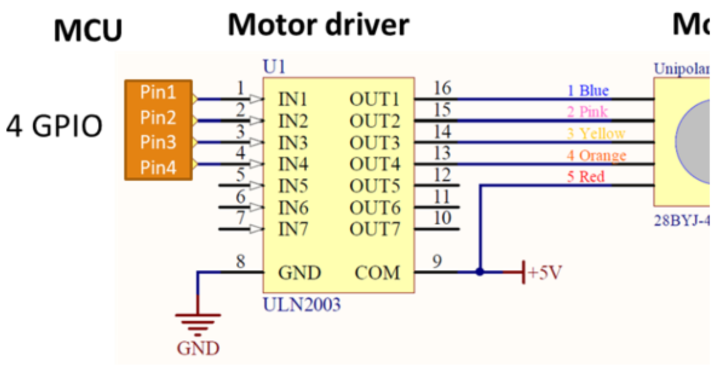
Hardware Connection

Read specification sheet of the motor and the motor driver for wiring and min/max input voltage/current.



- Rated Voltage: 5V DC
- Number of Phases: 4
- Stride Angle: 5.625°/64
- Gear ratio: 1/32
- Pull in torque: 300 gf.cm
- Coil: Unipolar 5 lead coil

MCU connection wiring



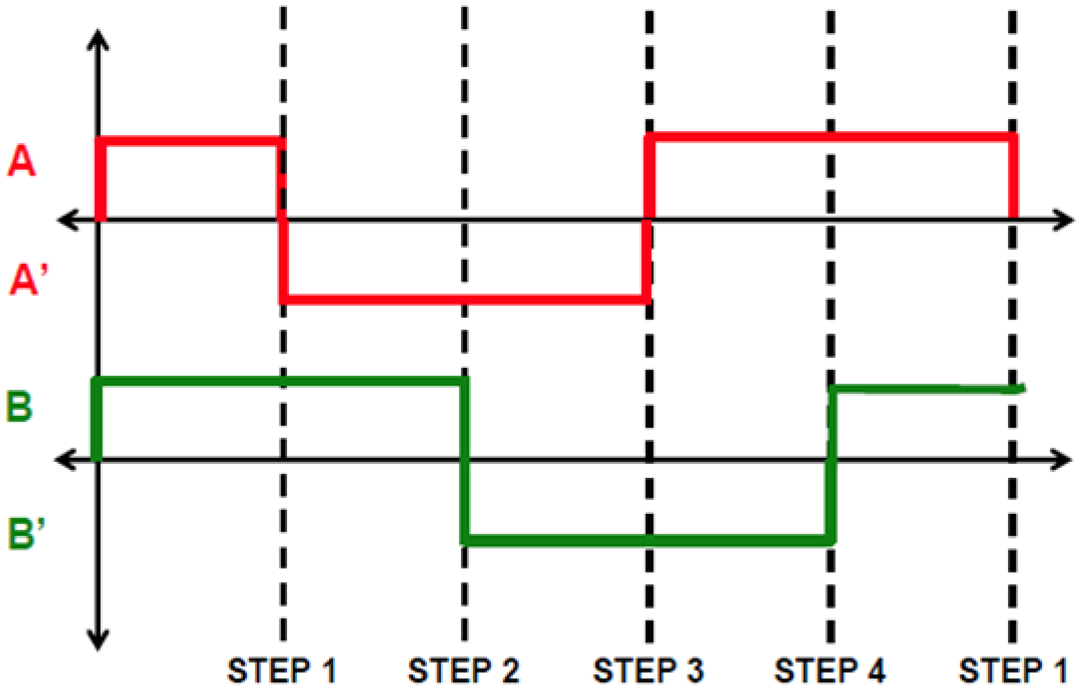
Motor driver (ULN2003)

Stepper Motor Sequence

We will use unipolar stepper motor for this lab

Fill in the blanks of each output data depending on the below sequence.

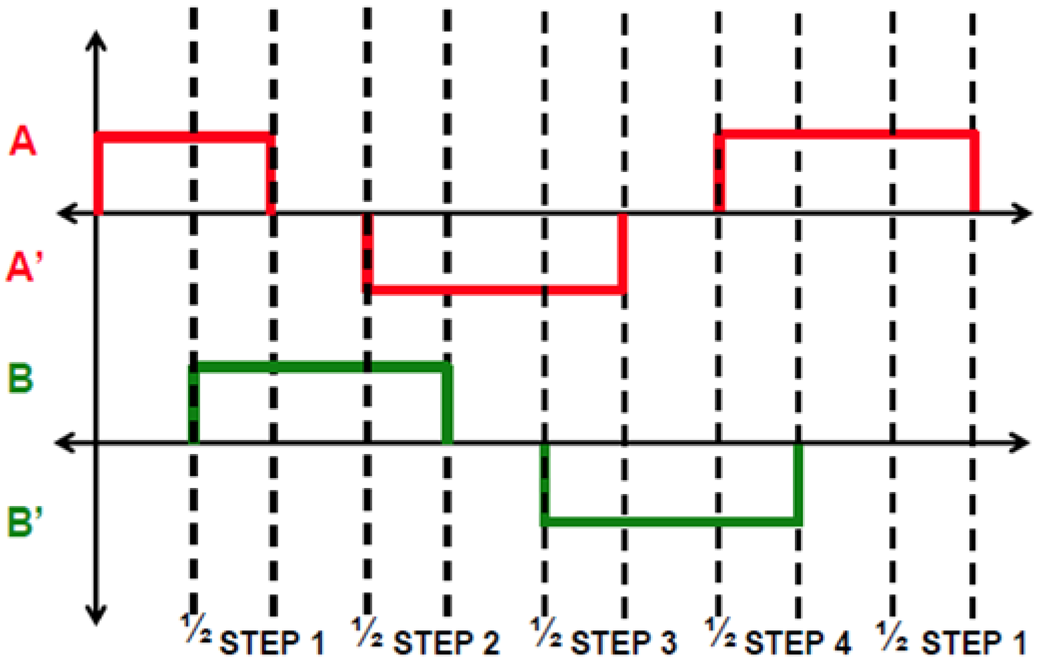
Full-stepping sequence



Phase	Port_Pin	Sequence			
		1	2	3	4
A	PB_10	H	L	L	H
B	PB_4	H	H	L	L
A`	PB_5	L	H	H	L
B`	PB_3	L	L	H	H

Full-stepping Sequence

Half-stepping sequence



Phase	Port_Pin	Sequence							
		1	2	3	4	5	6	7	8
A	PB_10	H	H	L	L	L	L	L	H
B	PB_4	L	H	H	H	L	L	L	L
A`	PB_5	L	L	L	H	H	H	L	L
B`	PB_3	L	L	L	L	L	H	H	H

Half-stepping Sequence

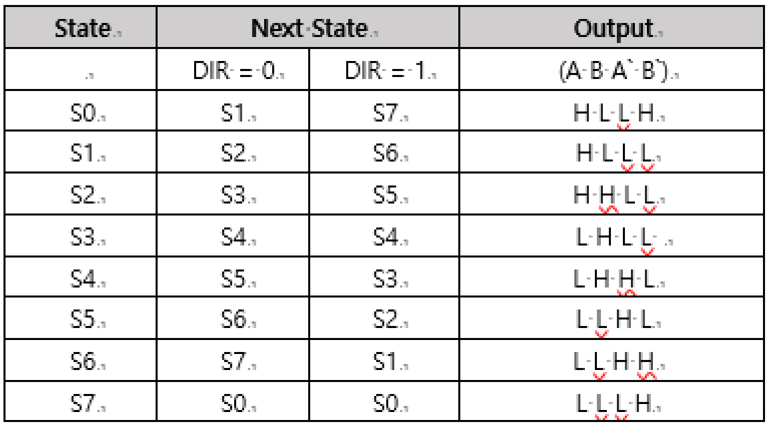
Finite State Machine

Draw a State Table for Full-Step Sequence. Use Moore FSM for this case. See *'Programming FSM'* for hints.

- Full-Stepping Sequence

State	Next State		Output
	DIR = 0	DIR = 1	(A B A` B`)
S0	S1	S3	H L L H
S1	S2	S0	H H L L
S2	S3	S1	L H H L
S3	S0	S2	L L H H

- Half-Stepping Sequence



State	Next State		Output
	DIR = 0	DIR = 1	(A B A` B`)
S0	S1	S7	H L L H
S1	S2	S6	H L L L
S2	S3	S5	H H L L
S3	S4	S4	L H L L
S4	S5	S3	L H H L
S5	S6	S2	L L H L
S6	S7	S1	L L H H
S7	S0	S0	L L L H

Problem 2: Firmware Programming

Create HAL library

Declare and Define the following functions in your library. You must

update your header files located in the directory `EC \lib\`.

ecStepper.h

```
// initlization stepper motor pin (output,NO PUPD,pushpull,Fast)

void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2,
GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);



// whatSpeed [rev/min]

void Stepper_setSpeed(long whatSpeed);



// Run for n Steps

void Stepper_step(int steps, int direction, int mode);



// Immediate Stop.

void Stepper_stop(void);
```

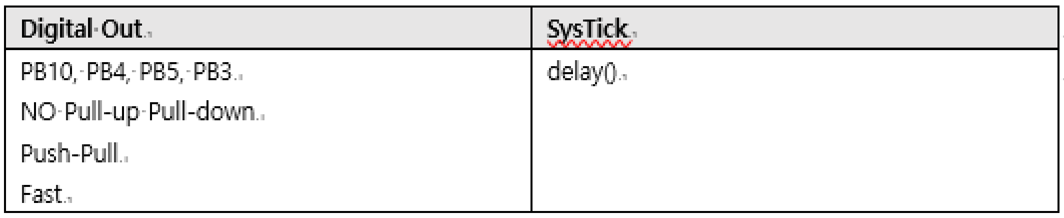
Procedure

1. Create a new project under the directory `\repos\EC\LAB\LAB_Stepper_Motor`
   - The project name is "**LAB_Stepper_Motor".**
   - Create a new source file named as "**LAB_Stepper_Motor.c"**

> You MUST write your name on the source file inside the comment section.

2. Include your updated library in `\repos\EC\lib\` to your project.
   - **ecGPIO.h, ecGPIO.c**
   - **ecRCC.h, ecRCC.c**
   - **ecEXTI.h, ecEXTI.c**
   - **ecSysTick.h**, **ecSysTick.c**
   - **ecStepper.h ecStepper.h**
3. Connect the MCU to the motor driver and the stepper motor.
4. Find out the number of steps required to rotate 1 revolution using Full-steppping.
5. Then, rotate the stepper motor 10 revolutions with 2 rpm. Measure if the motor rotates one revolution per second.
6. Repeat the above process with the opposite direction.
7. Increase and decrease the speed of the motor as fast as it can rotate to find the max speed of the motor.
8. Apply the half-stepping and repeat the above.

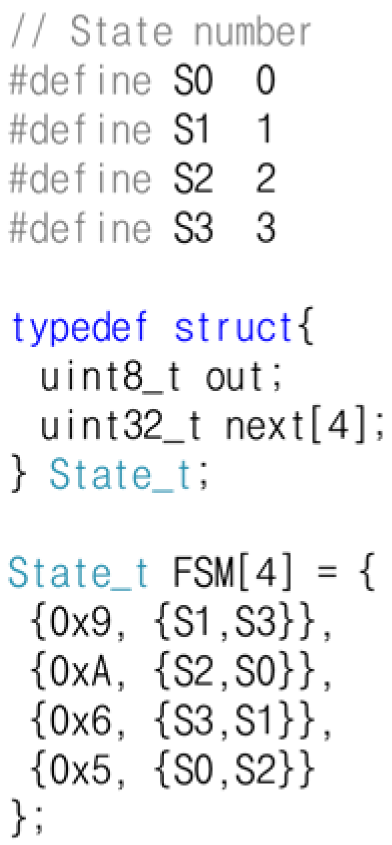
Configuration



Digital Out	SysTick
PB10, PB4, PB5, PB3 NO Pull-up Pull-down Push-Pull Fast	delay()

Requirement

You have to program the stepping sequence using the state table. You can define the states using structures. Refer to *'Programming FSM'* for hints.



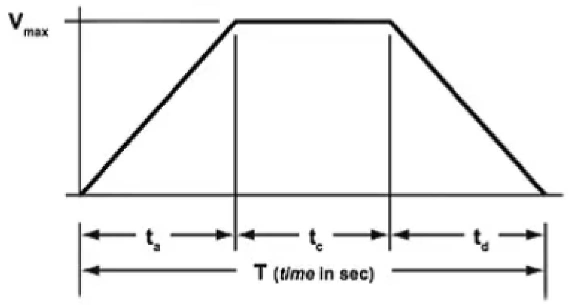
```c
// State number
#define S0  0
#define S1  1
#define S2  2
#define S3  3

typedef struct{
  uint8_t out;
  uint32_t next[4];
} State_t;

State_t FSM[4] = {
 {0x9, {S1,S3}},
 {0xA, {S2,S0}},
 {0x6, {S3,S1}},
 {0x5, {S0,S2}}
};
```

Discussion

1. Find out the trapezoid-shape velocity profile for stepper motor. When is this profile necessary?



For a standard trapezoidal move profile, where 1/3 of the total time is used for acceleration, 1/3 for constant velocity, and 1/3 for deceleration:

$$v_{avg} = \frac{d_t}{t_t}$$

$$v_{max} = 1.5 \cdot v_{avg}$$

$$v_{max} = 1.5 \cdot \frac{d_t}{t_t}$$

$$a = \frac{v_{max}}{t_a}$$

$$a = \frac{4.5 \cdot d_t}{t_t^2}$$

Motion profiles provide physical motion information and describe how the motor should behave during movement (often in terms of position, speed and acceleration) and are used by the servo controller to determine the command (voltage) to be sent to the motor.

Trapezoidal profiles are used for steady, constant-velocity movements.

The trapezoidal motion profile is arguably the most common in motion control applications, since it forms the basis of processes that require a period of constant velocity, such as dispensing, measuring, and machining.

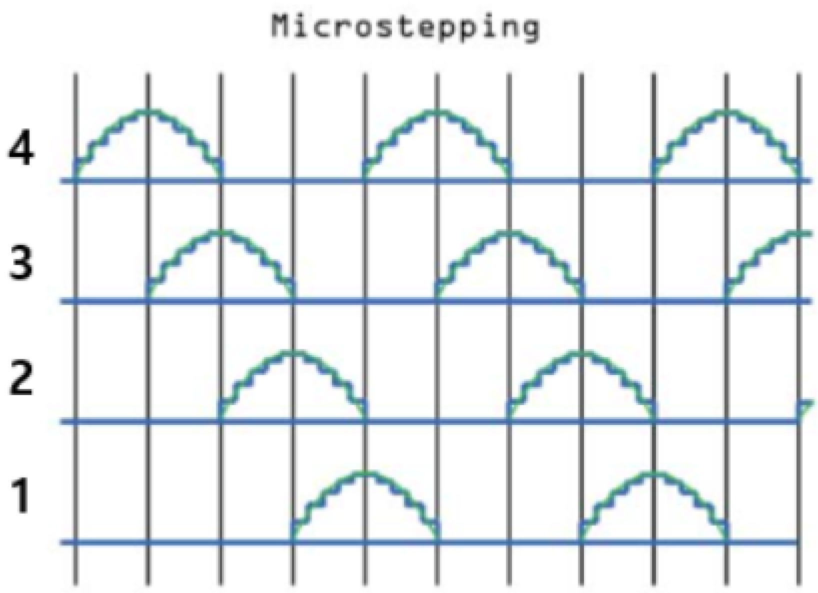
2. How would you change the code more efficiently for micro-stepping control? You don't have to code this but need to explain your strategy.

   For example, a stepper motor with a 1.8 degree step angle will make 200 steps for every full revolution of the motor (360 ÷ 1.8). This discrete motion means the motor's rotation isn't perfectly smooth, and the slower the rotation, the less smooth it is due to the relatively large step size. One way to alleviate this lack of smoothness at slow speeds is to reduce the size of the motor's steps. This is where microstepping comes in.

   Microstepping control divides each full step into smaller steps to help smooth out the motor's rotation, especially at slow speeds. For example, a 1.8 degree step can be divided up to 256 times, providing a step angle of 0.007 degrees (1.8 ÷ 256), or 51,200 microsteps per revolution.

   Microstepping is achieved by using pulse-width modulated (PWM) voltage to control current to the motor windings. The driver sends two voltage sine waves, 90 degrees out of phase, to the motor windings. While current increases in one winding, it decreases in the other winding. This gradual transfer of current results in smoother motion and more consistent torque production than full or half step control.

In conclusion, in order to perform microstepping, I need to give the current we connected to the motor in pwm, not just 0 and 1. That way, I can run the motor more smoothly.



Code

Your code goes here:

LAB_C:https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/c6fd8fb3ade635032660452f5c8675aa7ae753e4/lab/LAB_Stepper_Motor.c

include_C: https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/c6fd8fb3ade635032660452f5c8675aa7ae753e4/include/ecStepper.c

include_H:https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/c6fd8fb3ade635032660452f5c8675aa7ae753e4/include/ecStepper.h

Explain your source code with necessary comments.

Description with Code

- Main code

stepper motor operation with 2048 steps, DIR(1 = CW, 0= CCW), MODE(FULL,HALF)

```
//stepper motor operation with 2048 steps, DIR(1 = CW, 0= CCW), MODE(FULL,HALF)
int main(void) {
    // Initialiization ---------------------------------------------------------
    setup();


    // Inifinite Loop ----------------------------------------------------------
    while(1){
    if(i==0)Stepper_step(2048, 0, FULL);  // (Step : 2048, Direction : 0 or 1,
Mode : FULL or HALF) // 1=CW, 0 = CCW
    else Stepper_stop();                 // stepper motor stop
    }
}
```

- EXTI

```
// when button pin pressed update the flag and stepper motor stop
void EXTI15_10_IRQHandler(void) {
    if (is_pending_EXTI(BUTTON_PIN)) {
        Stepper_stop();                 // stepper motor stop
        i ^= 1;                         // flag update
        clear_pending_EXTI(BUTTON_PIN); // cleared by writing '1'
    }
}

```

- setup

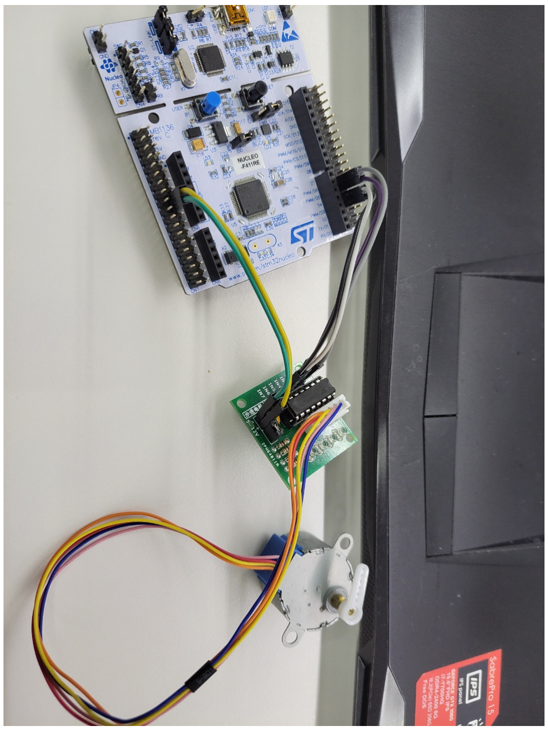
setting(PLL,Button pin(EXTI,GPIO),Stepper(PIN,SPEED))

```
// Initialiization
void setup(void)
{

    RCC_PLL_init();                                     // System Clock = 84MHz
    SysTick_init();                                     // Systick init
    EXTI_init(GPIOC,BUTTON_PIN,FALL,0);                 // External Interrupt
Setting
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);                // GPIOC pin13
initialization
    GPIO_pupdr(GPIOC, BUTTON_PIN, EC_PU);
    Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // Stepper GPIO pin
initialization
    Stepper_setSpeed(2);                                //  set stepper motor speed
max=14
}
```

Results

Experiment images and results



It is possible to operate the stepper motor with the desired RPM by determining the half mode and the full mode, determining the directions of CW and CCW. At this time, the maximum rpm is 14.

Add demo video link: https://youtu.be/qqZLf99kT1A

Reference

Complete list of all references used (github, blog, paper, etc)

https://www.motioncontroltips.com/what-is-a-motion-profile/

https://rasino.tistory.com/149

Troubleshooting

(Option) You can write Troubleshooting section

Even if it operates at the desired rpm, it initially works as desired, but it becomes faster as time goes by.

This may have a garbage value that is not completely lost on the Systick, or there may be a problem with the stepper motor itself.