LAB: Stepper Motor

Embedded Controller Lab Report

LAB: Stepper Motor

Date: 2023.11.09

Author: Hanmin Kim

Github:https://github.com/ansterdaz/HKim/tree/main

Demo video: https://youtube.com/shorts/SLHrq7HjZ-4

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.

Requirement

Hardware

MCU

- NUCLEO-F411RE

Actuator/Sensor/Others:

- 3Stepper Motor 28BYJ-48
- Motor Driver ULN2003
- breadboard

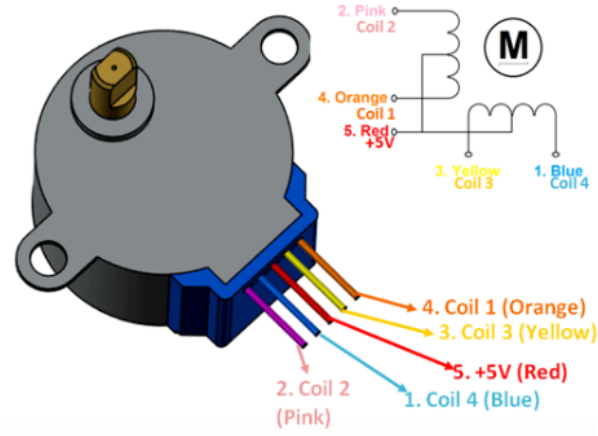
Software

Keil uVision, CMSIS, EC_HAL library

Problem 1: Stepper Motor

Hardware Connection

Unipolar Stepper Motor 28BYJ-48



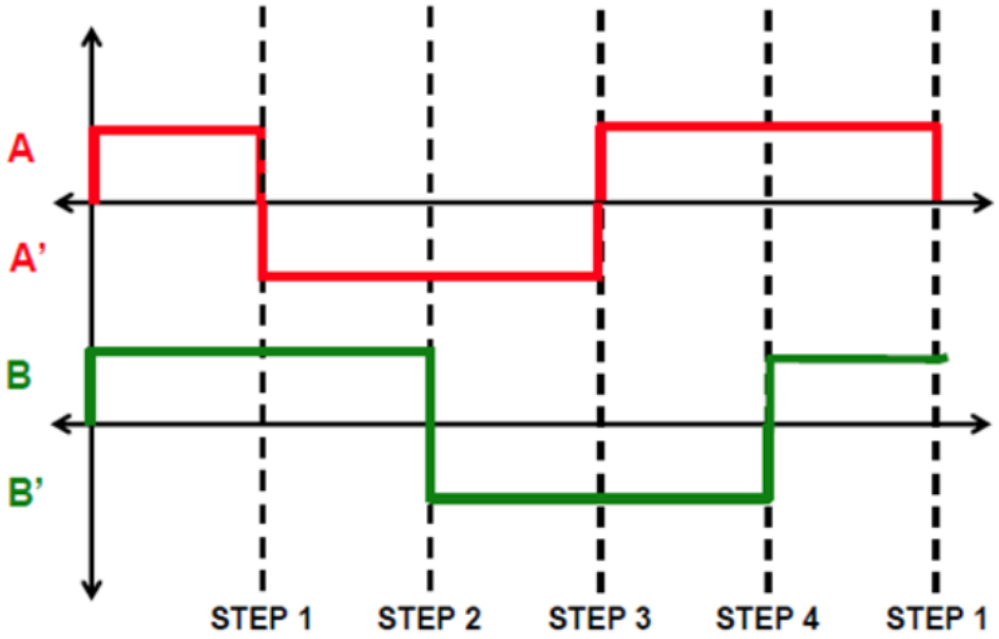
- Rated Voltage: 5V DC
- Number of Phases: 4
- Stride Angle: 5.625°/64
- Gear ratio: 1/32
- Pull in torque: 300 gf.cm
- Coil: Unipolar 5 lead coil

Stepper Motor Sequence

We will use unipolar stepper motor for this lab

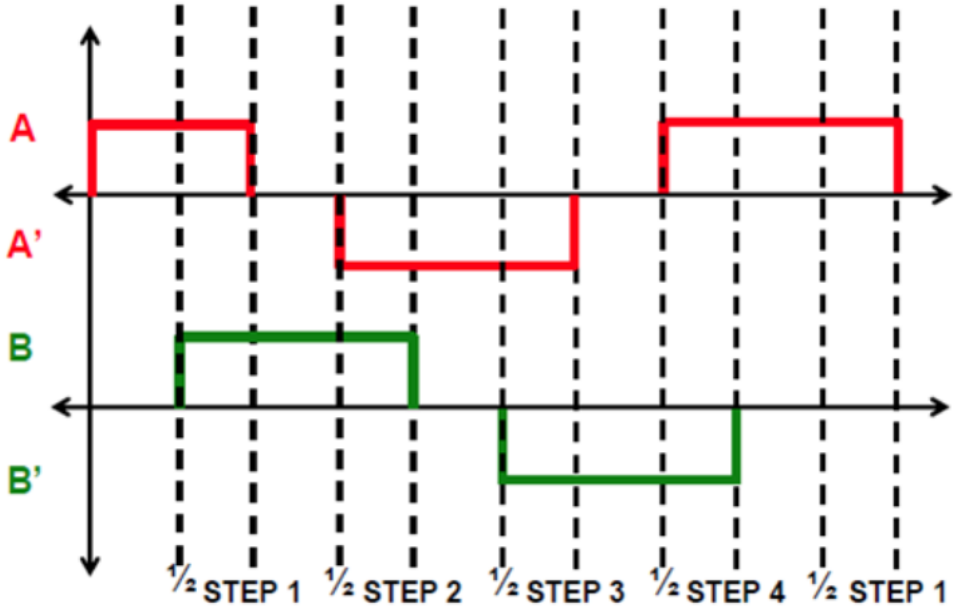
Fill in the blanks of each output data depending on the below sequence.

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

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

Finite State Machine

Draw a State Table for Full-Step Sequence. Use Moore FSM for this case. If you want, you may use Mealy FSM.

- Full-Stepping Sequence

State	Next State		output
	DIR=1	DIR=0	ABA'B'
S0	S1	S3	1100
S1	S2	S0	0110
S2	S3	S1	0011
S3	S0	S2	1001

- Half-Stepping Sequence

State	Next State		output
	DIR=1	DIR=0	ABA'B'
S0	S1	S7	1000
S1	S2	S0	1100
S2	S3	S1	0100
S3	S4	S2	0110
S4	S5	S3	0010
S5	S6	S4	0011
S6	S7	S5	1110
S7	S0	S6	1001

Problem 2: Firmware Programming

Creat HAL library

Download files:

- ecStepper_student.h, ecStepper_student.c

ecStepper.h

```
// Initialize with 4 pins
// ( A, B,  AN,  BN)
void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2,
GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);

//or   using ecPinNames.h
void Stepper_init(PinName_t A, PinName_t B,  PinName_t AN, PinName_t BN);


// whatSpeed [rev/min]
void Stepper_setSpeed(long whatSpeed);

// Run for n Steps
void Stepper_step(uint32_t steps, uint32_t direction, uint32_t 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 in the opposite direction.
7. Increase and decrease the speed of the motor as fast as it can rotate to find the maximum and minimum 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.

Discussion

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

   Acceleration Phase: During this phase, the stepper motor accelerates to reach the desired velocity. The acceleration is typically constant until the motor reaches its maximum velocity.

   Constant Velocity Phase: Once the desired velocity is reached, the motor maintains a constant speed for a certain period.

   Deceleration Phase: In the deceleration phase, the motor gradually slows down until it comes to a stop. The deceleration is usually symmetric to the acceleration phase.

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.

   Acceleration and Deceleration Profiles: Apply trapezoidal velocity profiles not only to the overall motion but also to the micro-steps within each full step. This ensures smoother transitions between micro-steps during acceleration and deceleration.

   Advanced Interpolation Techniques: Use advanced interpolation algorithms to calculate intermediate positions between micro-steps. This can include polynomial interpolation or spline interpolation to achieve smoother trajectories.

Current Control: Implement dynamic current control based on the micro-step position. Adjust the motor current based on the current micro-step, optimizing energy consumption and reducing heat generation.

Sensor Feedback: If possible, integrate sensor feedback (e.g., encoders) to provide real-time information about the motor's actual position. This information can be used to dynamically adjust the micro-stepping control, compensating for any deviations from the desired trajectory.

Code

ecStepper.c

```
//FULL stepping sequence  - FSM
typedef struct {
   uint8_t out;
     uint32_t next[2];
} State_full_t;

State_full_t FSM_full[4] = {
   {0b1100,{S1,S3}},
   {0b0110,{S2,S0}},
   {0b0011,{S3,S1}},
   {0b1001,{S0,S2}},

};

//HALF stepping sequence
typedef struct {
   uint8_t out;
     uint32_t next[2];
} State_half_t;

State_half_t FSM_half[8] = {
  {0b1000,{S1,S7}},
  {0b1100,{S2,S0}},
  {0b0100,{S3,S1}},
  {0b0110,{S4,S2}},
  {0b0010,{S5,S3}},
  {0b0011,{S6,S4}},
  {0b0001,{S7,S5}},
  {0b1001,{S0,S6}},
};
```

Created a state table for full and half and wrote the code as follows.

```c
int main(void) {
  // Initialiization --------------------------------------------------------
  setup();

  Stepper_step(10000, 0  ,HALF);  // (Step : 1024, Direction : 0 or 1, Mode : FULL or HALF)

  // Inifinite Loop ---------------------------------------------------------
  while(1){;}
}

// 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, EC_DIN);            // GPIOC pin13 initialization

  Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // Stepper GPIO pin initialization
  Stepper_setSpeed(5);                            //  set stepper motor speed
}
```

Can set the direction and mode of the motor in the main code and define the speed of the motor using the stepper_setSpeed function.

Results

It has been confirmed that it operates clockwise and counterclockwise in Half and Full.

It was confirmed that the minimum motor speed was 1rpm and the maximum speed was 14rpm.

Video Link: https://youtube.com/shorts/SLHrq7HjZ-4

Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-stepper-motor

https://github.com/ykkimhgu

Appendix

LAB_steppermotor.c

```c
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecEXTI.h"
#include "ecSysTick.h"
#include "ecStepper.h"

void setup(void);

int main(void) {
  // Initialiization --------------------------------------------------------
  setup();

  Stepper_step(10000, 0  ,HALF);  // (Step : 1024, Direction : 0 or 1, Mode : FULL or HALF)

  // Inifinite Loop ----------------------------------------------------------
  while(1){;}
}

// 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, EC_DIN);             // GPIOC pin13 initialization

  Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // Stepper GPIO pin initialization
  Stepper_setSpeed(5);                             //  set stepper motor speed
}

void EXTI15_10_IRQHandler(void) {
  if (is_pending_EXTI(BUTTON_PIN)) {
    Stepper_stop();
    clear_pending_EXTI(BUTTON_PIN); // cleared by writing '1'
  }
}
```