Embedded Controller

LAB: Stepper Motor

Date: 2021-11-04

Name(ID):Park JeongWoo

Partner Name:Lee JunGi

I. 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.

Hardware

NUCLEO -F411RE, Stepper Motor 28BYJ-48, Motor Driver ULN2003, 5V power supply

Software

Keil uVision IDE, CMSIS, EC_HAL

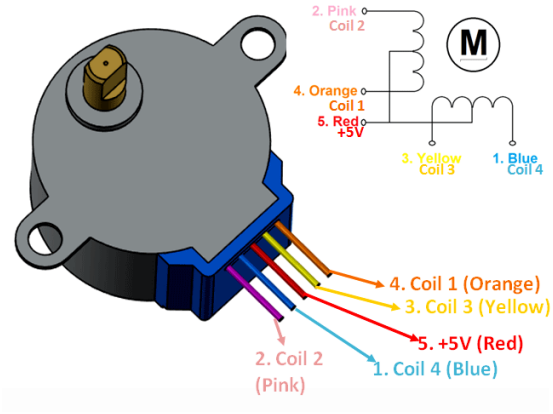
II. Procedure

A. Hardware Connection

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

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



MCU connection wiring

Motor driver (ULN2003)

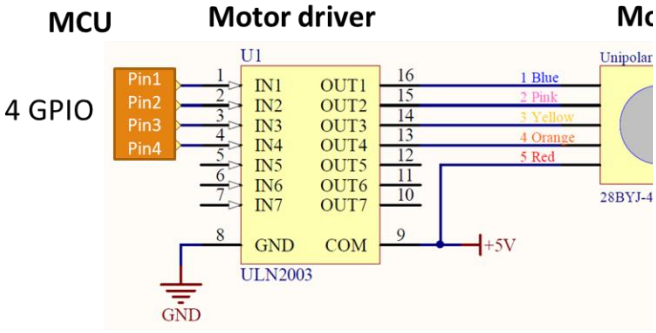


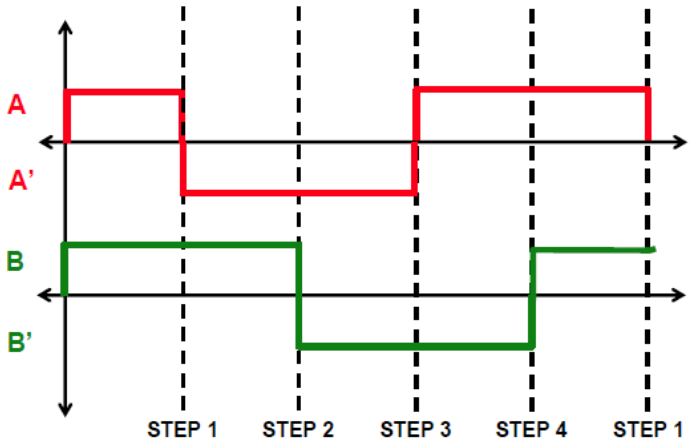
Figure 1. Stepper motor connection

B. Stepper Motor Sequence

We will use bipolar stepper motor for this lab

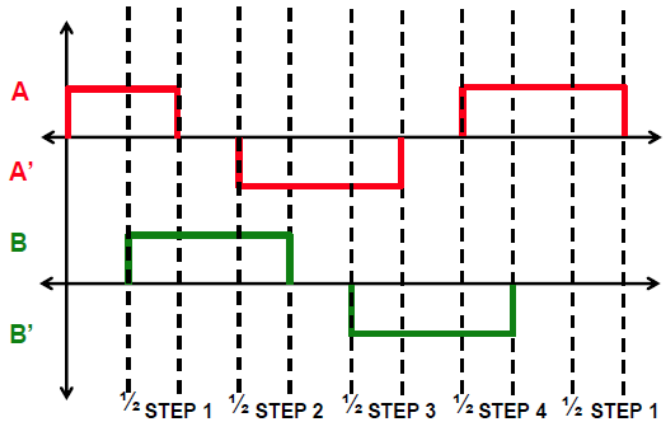
Fill in the blanks of each output data depending on the below sequence.

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

C. Finite State Machine

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

1) Full-Stepping Sequence

State	Next State		Output
	DIR=0	DIR=1	(A A' B B')
S0	S1	S3	1100
S1	S2	S0	0110
S2	S3	S1	0011
S3	S0	S2	1001

2) Half- Stepping Sequence

State	Next State		Output
	DIR=0	DIR=1	(A A' B 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	0001
S7	S0	S6	1001

C. Configuration

Create a new project named as "**LAB_Steppermotor".**

Name the source file as "**LAB_Steppermotor.c"**

You MUST write your name in the top of the source file, inside the comment section.

Configure Input and Output pins

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

D. Firmware Programming

ecStepper.h,c

Function	Description
void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2, GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);	Initialize with 4 pins
void Stepper_setSpeed (long whatSpeed);	whatSpeed [rev/min], Time delay between each step
void Stepper_step(int steps, int direction, int mode);	Run for nSteps
void Stepper_run (int direction, int mode);	Continuous Run
void Stepper_stop (void);	Immediate Stop. It should works with interrupt such as button or serialRead

Documenation of Library

Stepper_init()

Select stepper motor output pin and initialization

```
void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2, GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, i
```

Parameters

- *GPIO_TypeDef port1~4**: Select GPIO port GPIOA~GPIOH
- **pin1~4**: Select the pin number 0~15

Example code

```
Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // PB10, PB4, PB5, PB3 initialization
```

Stepper_setSpeed()

Select Rotation speed of the motor

```
void Stepper_setSpeed (long whatSpeed, int mode);
```

Parameters

- **whatSpeed**: Motor Speed (Unit: RPM)
- **mode**: FULL(0) , HALF(1)

Example code

```
Stepper_setSpeed (3,FULL); //3RPM, FULL mode
```

Stepper_step()

The number of next step, 5steps => S0-> S1-> S2-> S3-> S0

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

Parameters

- **steps:** total steps numbers
- **direction:** Right cycle(0), Left cycle(1)
- **mode:** FULL(0) , HALF(1)

Example code

```
Stepper_step(1000, Right, FULL); // FULL MODE, Right cycle, 1000steps
```

Stepper_stop()

Stop the motor.

```
void Stepper_stop(void);
```

Example code

```
Stepper_stop(); // stop the motor
```

Stepper_pinOut()

It receives voltage depending on the condition.

```
void Stepper_pinOut(uint32_t state, int mode);
```

Parameters

- **state:** S0, S1, S2 ...
- **mode:** FULL(0) , HALF(1)

Example code

```
Stepper_pinOut(1, FULL)// FULL MODE, Output the state S1
```

Create a simple program to drive a stepper motor.

- Connect the MCU to the motor driver and the stepper motor.
- Connect the motor driver to external power supply (5VDC)
- Find out the number of steps required to rotate 1 revolution using Full-steppping
- Then, rotate the stepper motor 10 revolutions with 2 rpm. Measure if the motor rotates one revolution per second.
- Repeat the above process with the opposite direction

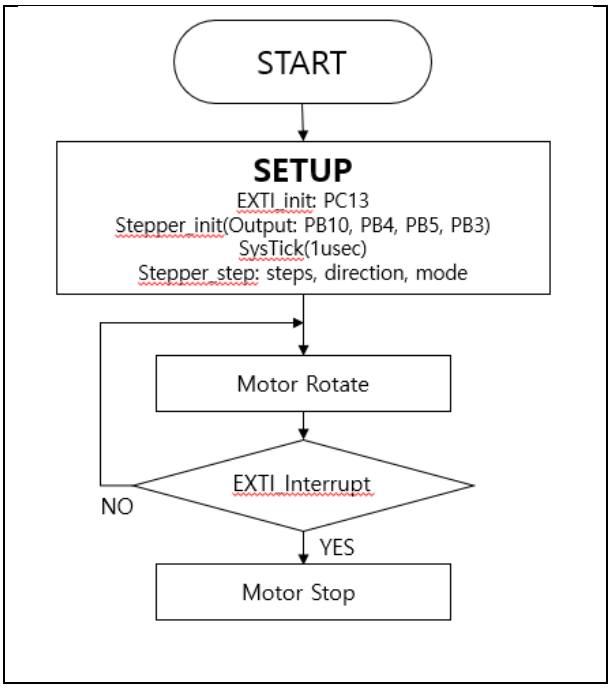
- Increase and decrease the speed of the motor as fast as it can rotate to find the max speed of the motor
- Apply the half-stepping and repeat the above

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.

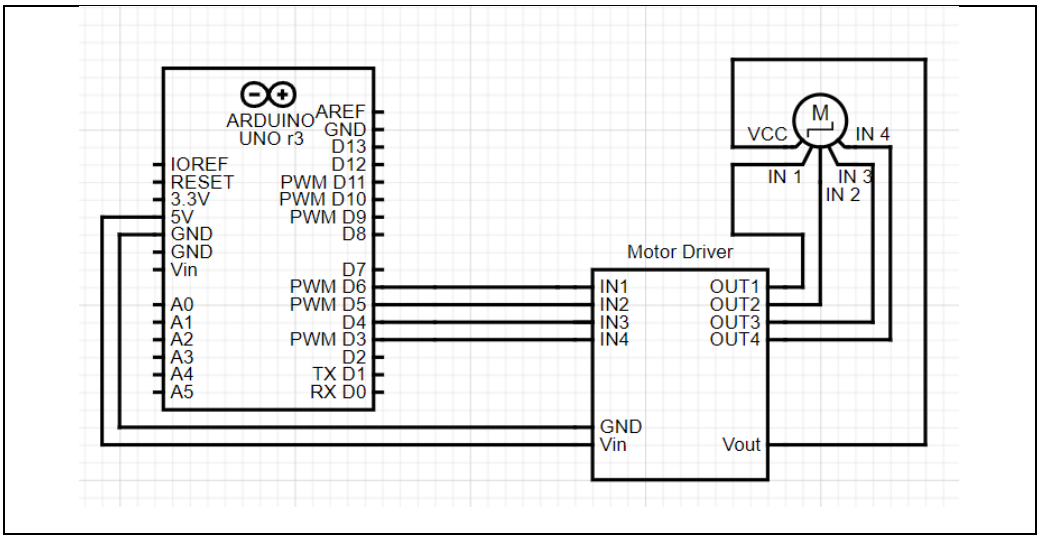
```
// State number
#define S0  0
#define S1  1
#define S2  2
#define S3  3

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}}
};
```

- Flow Chart



-

- Circuit diagram



Discussion

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

When using a rectangular-shape velopcity, inertia may occur due to a sudden change in voltage, resulting in a large vibration of the motor. However, the trapzoid-shape velocity profile has a linear section between acceleration and deceleration to prevent sudden voltage changes and reduce motor vibration.

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. The simplest implementation is to increase the number of rotors or stator. However, if this is limited, the intensity of the current must be adjusted. That is, there is a method of changing the value of PWM. If there are four states in FULL mode, the duty ratio is sequentially increased as shown in 0.3, 0.5, and 0.7, when s0. This can overcome the limited hardware situation.

Conclusion & TrobleShooting

- ## TrobleShooting

  Q: The Delay_ms function receives only integer numbers, but after passing through the Stepper_setSpeed function, decimal points appear and descend. For example, if 2.9 is returned, 2 is delay_ms (2). Then the RPM is not right.

  A: Create a Delay_us function using Systick in units of 1 usec.

- ## Conclusion

  In this experiment, the RPM of the motor was adjusted in consideration of the deceleration ratio and step per rev of the motor gear, and the stepper motor was driven using two modes, HALF and FULL.

Appendix

- Demo Link

  https://youtu.be/X6iJAJ6D8vY

- LAB_StepperMotor.c

```c
/**
*********************************************************************************
* @author  SSSLAB
* @Mod     2021-11-03 by JeongWoo Park
* @brief   Embedded Controller:  LAB6_StepperMotor
*                - Select the mode, direction of motor
*
*********************************************************************************
*/
#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
    SysRick_Initialize(1);                                // Systick init

    EXTI_init(GPIOC,BUTTON_PIN,FALL,0);             // External Interrupt Setting
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);           // GPIOC pin13 initialization
    Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // Stepper GPIO pin initialization
    Stepper_setSpeed(5,HALF);                          //  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'
    }
}
```

- ecStepper.c

```c
#include "stm32f4xx.h"
#include "ecStepper.h"

//State number
#define S0 0
#define S1 1
#define S2 2
#define S3 3
#define S4 4
#define S5 5
#define S6 6
#define S7 7


// Stepper Motor function
uint32_t direction = 1;
uint32_t step_delay = 100;
uint32_t step_per_rev = 64;


// Stepper Motor variable
volatile Stepper_t myStepper;


 //FULL stepping sequence  - FSM
typedef struct {
   uint8_t out;
   uint32_t next[2];
} State_full_t;
State_full_t FSM_full[4] = {
  {0xC,{S1,S3}},
  {0x6,{S2,S0}},
  {0x3,{S3,S1}},
  {0x9,{S0,S2}}
};
 //HALF stepping sequence
typedef struct {
   uint8_t out;
   uint32_t next[2];
} State_half_t;
State_half_t FSM_half[8] = {
  {0x8, {S1, S7}},
  {0xC, {S2, S0}},
  {0x4, {S3, S1}},
  {0x6, {S4, S2}},
  {0x2, {S5, S3}},
  {0x3, {S6, S4}},
  {0x1, {S7, S5}},
  {0x9, {S0, S6}}

};


void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2, GPIO_TypeDef* port3, int pin3, GPI(

//  GPIO Digital Out Initiation
   myStepper.port1 = port1;
   myStepper.pin1  = pin1;
   // Repeat for port2,pin2~port4,pin4
   myStepper.port2 = port2;
   myStepper.pin2  = pin2;

   myStepper.port3 = port3;
   myStepper.pin3  = pin3;

   myStepper.port4 = port4;
   myStepper.pin4  = pin4;



//  GPIO Digital Out Initiation
    GPIO_init(myStepper.port1,myStepper.pin1,OUTPUT);
    GPIO_init(myStepper.port2,myStepper.pin2,OUTPUT);
```

```
    GPIO_init(myStepper.port2,myStepper.pin2,OUTPUT);
    GPIO_init(myStepper.port3,myStepper.pin3,OUTPUT);
    GPIO_init(myStepper.port4,myStepper.pin4,OUTPUT);
    // No pull-up Pull-down , Push-Pull, Fast
    // Port1,Pin1 ~ Port4,Pin4
    GPIO_setting(myStepper.port1, myStepper.pin1, PUSH_PULL, NO_PUPD, FAST_SPEED);
    GPIO_setting(myStepper.port2, myStepper.pin2, PUSH_PULL, NO_PUPD, FAST_SPEED);
    GPIO_setting(myStepper.port3, myStepper.pin3, PUSH_PULL, NO_PUPD, FAST_SPEED);
    GPIO_setting(myStepper.port4, myStepper.pin4, PUSH_PULL, NO_PUPD, FAST_SPEED);

}
void Stepper_pinOut (uint32_t state, int mode){

    if (mode ==FULL){          // FULL mode



     GPIO_write(myStepper.port1, myStepper.pin1, (FSM_full[state].out & 0x8) >> 3);    // YOUR CODE_____);
     GPIO_write(myStepper.port2, myStepper.pin2, (FSM_full[state].out & 0x4)  >> 2);
     GPIO_write(myStepper.port3, myStepper.pin3, (FSM_full[state].out & 0x2)  >> 1);
     GPIO_write(myStepper.port4, myStepper.pin4, (FSM_full[state].out & 0x1)  >> 0);
    }
   else if (mode ==HALF){    // HALF mode
     GPIO_write(myStepper.port1, myStepper.pin1, (FSM_half[state].out & 0x8) >> 3);    // YOUR CODE_____);
     GPIO_write(myStepper.port2, myStepper.pin2, (FSM_half[state].out & 0x4)  >> 2);
     GPIO_write(myStepper.port3, myStepper.pin3, (FSM_half[state].out & 0x2)  >> 1);

      GPIO_write(myStepper.port4, myStepper.pin4, (FSM_half[state].out & 0x1)  >> 0);
    }


}

void Stepper_setSpeed (long whatSpeed, int mode){      // rppm
  if(mode == FULL)
    step_delay = (uint32_t) (60000*1000)/(64*32*whatSpeed);// Convert rpm to milli sec
  else if(mode == HALF)
    step_delay = (uint32_t) (60000*1000)/(64*32*whatSpeed*2);
}

void Stepper_step(int steps, int direction,int mode){
   int step_number = 0;
   myStepper._step_num = steps;
   uint32_t state_number = 0;
   int max_step = 3;
   if (mode == HALF) max_step = 7;


   for(;myStepper._step_num>0;myStepper._step_num--){ // run for step size
        delay_us(step_delay);                                  // delay (step_delay);


        if(direction == 0) step_number++;                 // + direction step number++
        else if(direction ==1)   step_number--;                                 // - direction step number--

        // YOUR CODE                                    //  step_number must be 0 to max_step
       // YOUR CODE
        if (step_number>max_step)
            step_number = 0;
        else if (step_number<0)
            step_number = max_step;



        //if (mode == FULL)
        //{   state_number=FSM_full[state_number].next[direction];
        //    //state_number = step_number;           // YOUR CODE       // state_number = 0 to 3 for FULL step mode
        //    state_number++;
        //}
        //
        //    if(state_number == 4)
        //      state_number = 0;
      //  else if (mode == HALF)
        //      state_number= step_number;// YOUR CODE       // state_number = 0 to 7 for HALF step mode
          state_number = step_number;
          Stepper_pinOut(state_number, mode);
   }
}
```

```
159 void Stepper_stop (void){
160
161       myStepper._step_num = 0;
162       // All pins(Port1~4, Pin1~4) set as DigitalOut '0'
163       GPIO_write(myStepper.port1, myStepper.pin1, 0x0);
164       GPIO_write(myStepper.port2, myStepper.pin2, 0x0);
165       GPIO_write(myStepper.port3, myStepper.pin3, 0x0);
166       GPIO_write(myStepper.port4, myStepper.pin4, 0x0);
167
168
169 }
170
171
```

- ecStepper.h

```
 1  #include "stm32f411xe.h"
 2  #include "ecGPIO.h"
 3  #include "ecSystick.h"
 4
 5  #ifndef __EC_STEPPER_H
 6  #define __EC_STEPPER_H
 7
 8  #ifdef __cplusplus
 9   extern "C" {
10  #endif /* __cplusplus */
11
12  //State mode
13  #define HALF 0
14  #define FULL 1
15
16  /* Stepper Motor */
17  //stepper motor function
18
19 typedef struct{
20    GPIO_TypeDef *port1;
21    int pin1;
22    GPIO_TypeDef *port2;
23    int pin2;
24    GPIO_TypeDef *port3;
25    int pin3;
26    GPIO_TypeDef *port4;
27    int pin4;
28    int _step_num;
29 } Stepper_t;
30
31
32 void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2, GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);
33 void Stepper_setSpeed (long whatSpeed, int mode);
34 void Stepper_step(int steps, int direction, int mode);
35 void Stepper_run (int direction, int mode);
36 void Stepper_stop (void);
37
38 void Stepper_pinOut (uint32_t state, int mode);
39 #ifdef __cplusplus
40 }
41 #endif /* __cplusplus */
42
43 #endif
44
```