REPORT

Title: Stepper Motor

Submission Date	2021.11.05	Major	Mechanical and control engineering
Subject	Embedded Controller	Professor	Young-Keun Kim
Name	Yeong-Won Song	Student Number	21700375
Partner Name	Sang-Hyeon-Kim	Partner Number	21700102

Contents

I. Introdution

1.1 Purpose

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

1.2 Parts List

Parts	Specification	Quantity
Stepper-Motor	28BYJ-48, Motor Driver ULN2003	1
breadboard	-	1
NUCLEO-F411RE	Arm®(a) Cortex®-M4 with FPU	1

Table 1. Part List

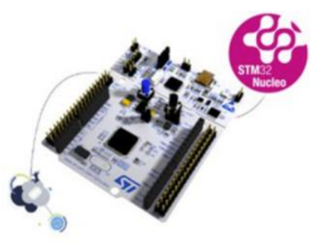
		
NUCLEO-F411RE	Stepper-Motor	Breadboard

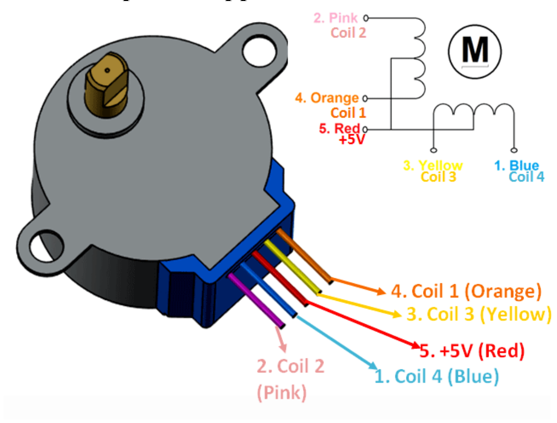
Table 2. experiment equipment

II. Procedure

2.1 Hardware Connection

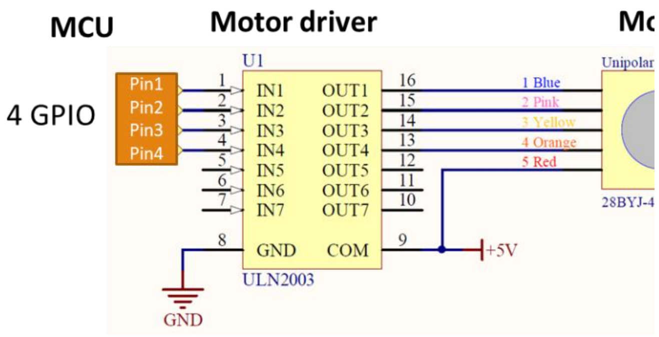
Read specification sheet of the motor and the motor driver for wiring and min/max input voltage/current

Unipolar Stepper Motor 28BYJ-48



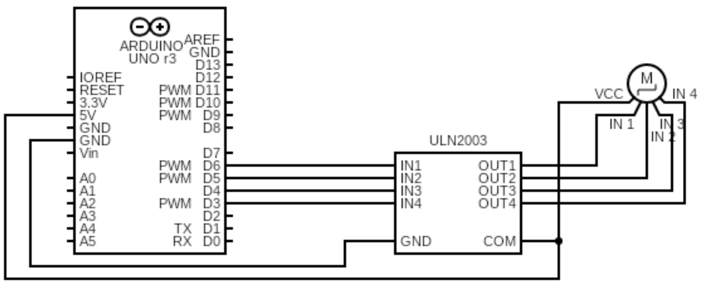
- Rated Voltage: 5V DC
- Number of Phases: 4
- Stride Angle: 5.625°/64
- Gear ratio: 1/32
- Pull in torque: 300 gf.cm
- Coil: Unipolar 5 lead coil

MCU connection wiring



Motor driver (ULN2003)

Figure 1. Stepper motor connection



ARDUINO UNO r3
AREF GND D13 D12 PWM D11 PWM D10 PWM D9 D8 D7 PWM D6 PWM D5 D4 PWM D3 D2 TX D1 RX D0
IOREF RESET 3.3V 5V GND GND Vin A0 A1 A2 A3 A4 A5
ULN2003
IN1 IN2 IN3 IN4 GND
OUT1 OUT2 OUT3 OUT4 COM
M
VCC
IN 1
IN 2
IN 3
IN 4

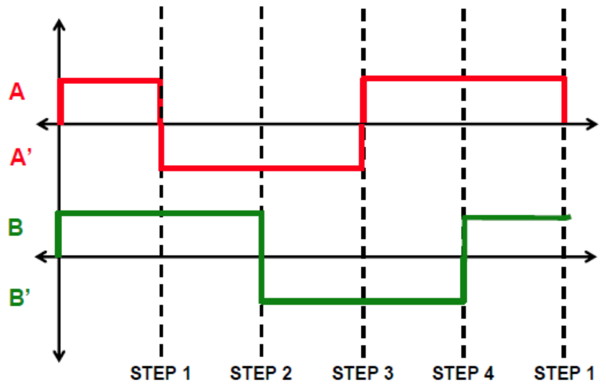
Figure 2. External Circuit

2.2 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.

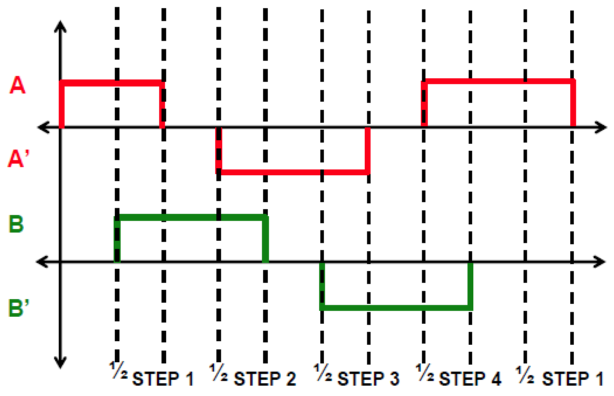
Full-stepping sequence



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

Table 3. Full stepping

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

Table 4. Half stepping

2.3 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 B A' 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 B A' 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

2.4 Configuration

Create a new project named as “**LAB_Steppermotor”.**

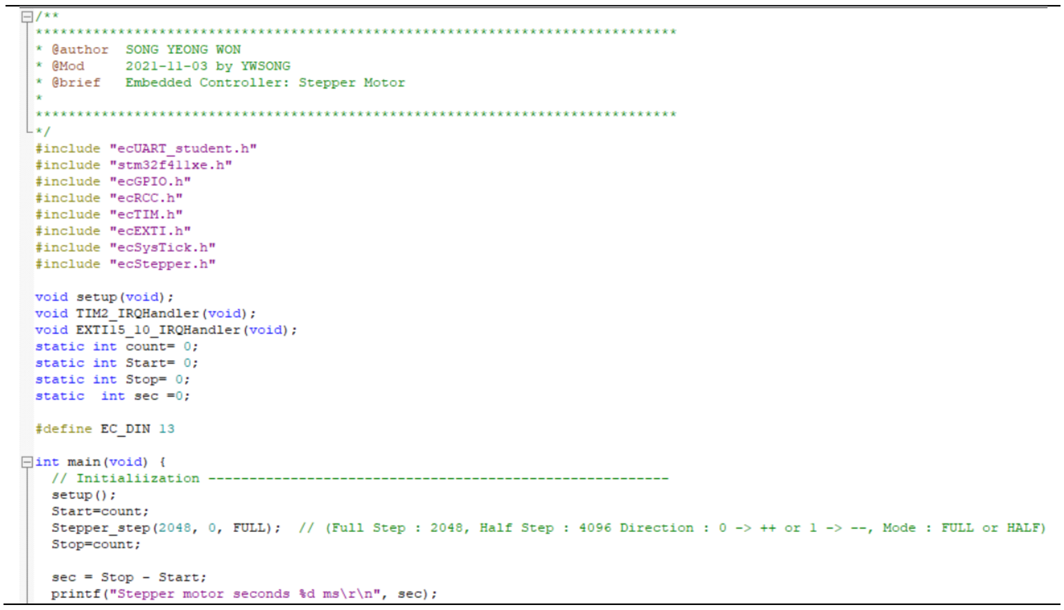
Name the source file as “**LAB_Steppermotor.c**”

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

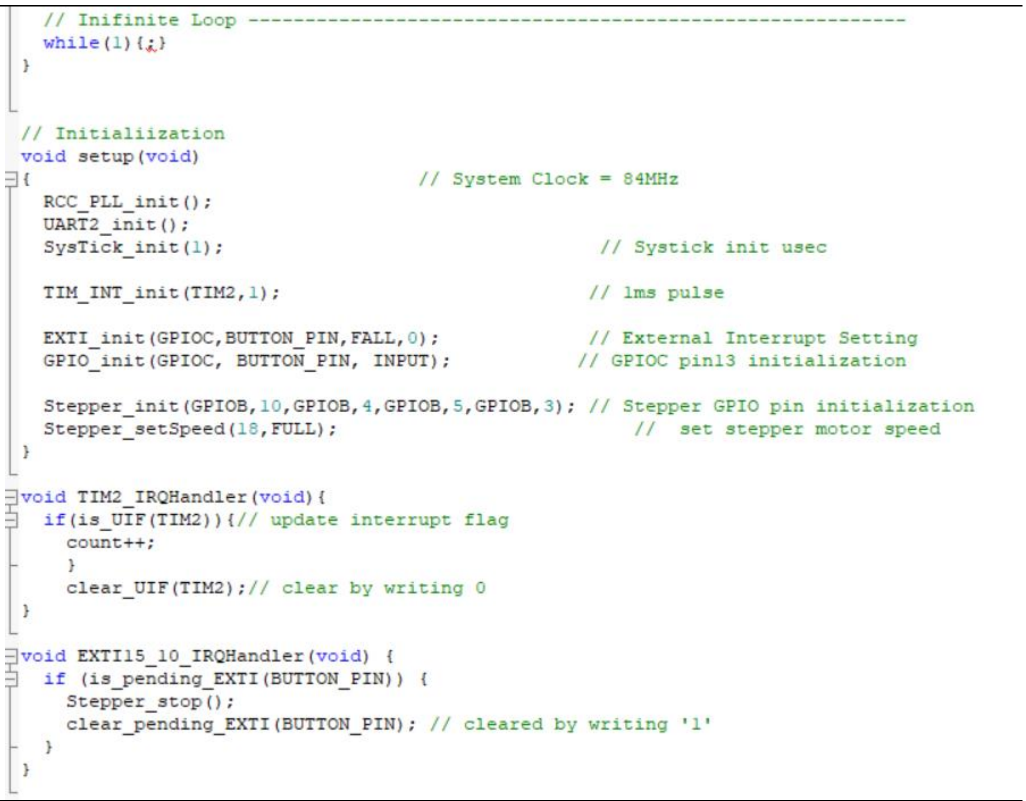
Configure Input and Output pins

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

- **Main code**



```
/**
******************************************************************************
* @author  SONG YEONG WON
* @Mod     2021-11-03 by YWSONG
* @brief   Embedded Controller: Stepper Motor
*
******************************************************************************
*/
#include "ecUART_student.h"
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecEXTI.h"
#include "ecSysTick.h"
#include "ecStepper.h"

void setup(void);
void TIM2_IRQHandler(void);
void EXTI15_10_IRQHandler(void);
static int count= 0;
static int Start= 0;
static int Stop= 0;
static  int sec =0;

#define EC_DIN 13

int main(void) {
  // Initialiization --------------------------------------------------------
  setup();
  Start=count;
  Stepper_step(2048, 0, FULL);  // (Full Step : 2048, Half Step : 4096 Direction : 0 -> ++ or 1 -> --, Mode : FULL or HALF)
  Stop=count;

  sec = Stop - Start;
  printf("Stepper motor seconds %d ms\r\n", sec);
```



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

// Initialiization
void setup(void)
{                                  // System Clock = 84MHz
  RCC_PLL_init();
  UART2_init();
  SysTick_init(1);                                // Systick init usec

  TIM_INT_init(TIM2,1);                           // 1ms pulse

  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(18,FULL);                          //  set stepper motor speed
}

void TIM2_IRQHandler(void){
  if(is_UIF(TIM2)){// update interrupt flag
    count++;
    }
    clear_UIF(TIM2);// clear by writing 0
}

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

Figure 3. main code of stepper motor

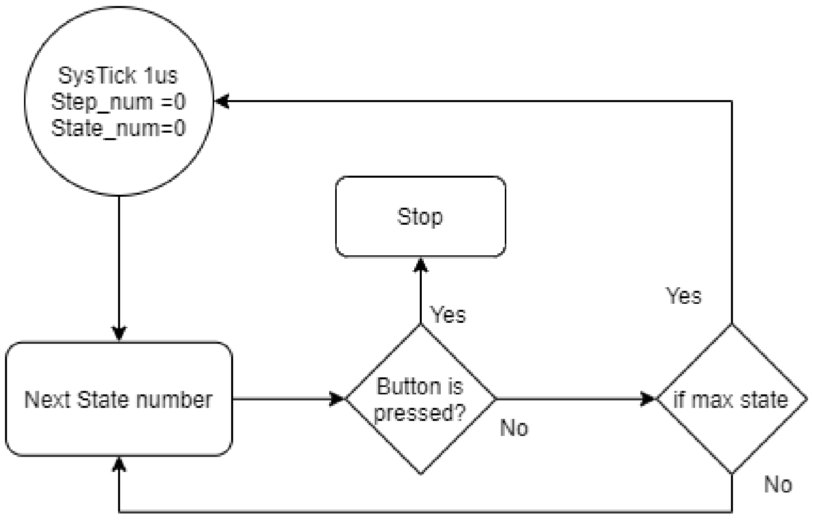


Figure 4. Flow chart

2.5 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

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 stepping

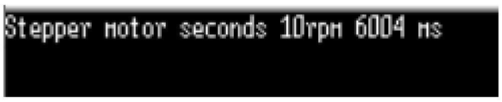
  The stride angle of the Stepper motor is $5.625°/64$. Also, the gear ratio is 1/32.

  Full mode is 64*32=2048steps, and Half mode is 64*32*2 =4096steps.

  - Full mode

    Direction CW (clockwise)

    Direction CCW (counterclockwise)



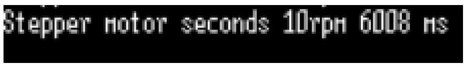
```
Stepper motor seconds 10rpm 6004 ms
```

Figure 5. full mode time check

Half mode

Direction CW (clockwise)

Direction CCW (counterclockwise)

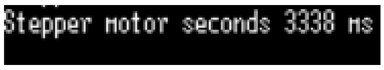


```
Stepper motor seconds 10rpm 6008 ms
```

Figure 6. Half mode time check

- 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.

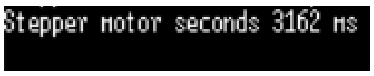
  When stepper motor is full mode and clockwise direction, the maximum rpm is 18 and it takes a total of 3.338 seconds for 1 revolution.



```
Stepper motor seconds 3338 ms
```

Figure 7. full mode CW max rpm time check

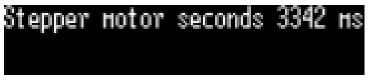
  When stepper motor is full mode and counterclockwise direction, the maximum rpm is 19 and it takes a total of 3.162 seconds for 1 revolution.



```
Stepper motor seconds 3162 ms
```

Figure 8. full mode CCW max rpm time check

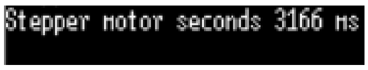
  When stepper motor is half mode and clockwise direction, the maximum rpm is 18 and it takes a total of 3.342 seconds for 1 revolution.



```
Stepper motor seconds 3342 ms
```

Figure 9. half mode CW max rpm time check

  When stepper motor is half mode and counterclockwise direction, the maximum rpm is 18 and it takes a total of 3.166 seconds for 1 revolution.



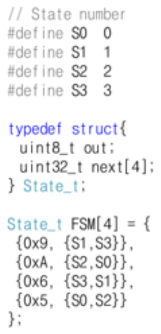
```
Stepper motor seconds 3166 ms
```

Figure 10. half mode CCW max rpm time check

- Apply the half-stepping and repeat the above

Requirement

You must 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}}
};
```

Figure 11. FSM structure

- Discussion

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

Stepper motors cannot change angular velocity simply by changing voltages like DC motors. Usually, the angle of the step motor is controlled by connecting the step motor driver to input a pulse. The angular velocity is controlled by changing the frequency of this pulse. In this case, the step motor may not be turned on or off by raising the target angular speed from the stopped state to the target angular speed at once. In the case of instantaneous acceleration, there is a possibility that the torque becomes infinite regardless of the inertia, resulting in the dephosphorization method. In addition, synchronization loss between the input pulse and the motor rotation may occur due to overload or rapid speed change. Therefore, when using a step motor, an acceleration/deceleration profile is created to set the angular acceleration to the target acceleration. To provide time for acceleration and deceleration, linear changes in pulse rates are related. The Trapezoid-shape velocity profile expands the range of operation that the stepper motor can start on its own. Since acceleration/deceleration is performed in a linear state, the rotational power of the motor for the input pulse may be stably transmitted or reduced.

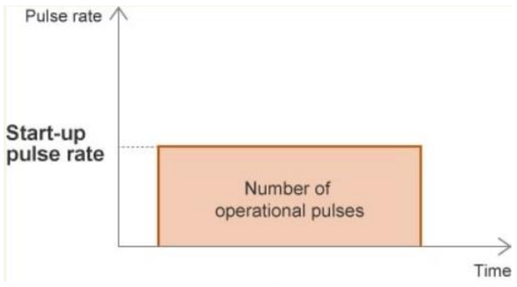


Figure 12. rectangular-shape velocity profile

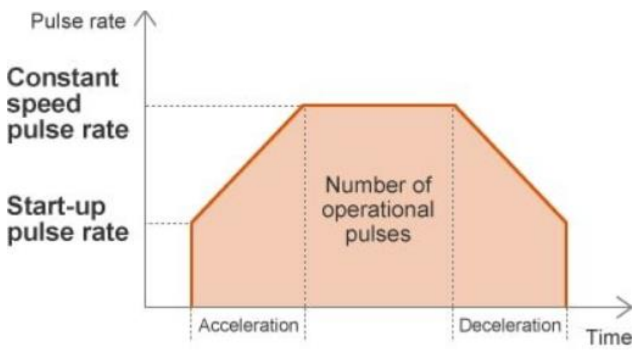


Figure 13. trapezoid-shape velocity profile

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.**

   A microcontroller can use full stepping or half stepping to control a stepper motor. Micro-stepping divides a full step into multiple smaller steps. It moves the shaft in a smaller angle increment, provides a much smoother movement, and reduces the problems of movement noise and vibration. we want to do more to do smooth. Vibration can be reduced. A Simple way is increasing rotor pole pairs or stator phase. However, when micro-stepping is not possible in hardware, we should be able to perform micro-stepping control in software. For example, if voltage 1 is applied to increase from 0 to 1, the voltage is applied through a total of 5 steps to 0.2, 0.4 ,0.6 ,0.8 1 by adjusting the duty ratio. To do so, divide one step into five steps and increase it. This allows us to adjust the voltage in detail and allows us to perform the micro control we want because one step splits into five more times. Then, more detailed step adjustment may be performed than the original step, and vibration and smoothness of the motor may be increased.

III. Conclusion

Through this Lab, we learned about how to control the stepper motor. We understood the operating principle of the stepper motor, and it could be implemented as an actual code based on the state definition according to the full mode or half mode. In addition, the rotation time could be calculated using timer interrupt, and the operation of the stepper motor could be stopped when using external interrupt.

- Troubleshooting

Q. How did you accurately measure the time according to the revolution?

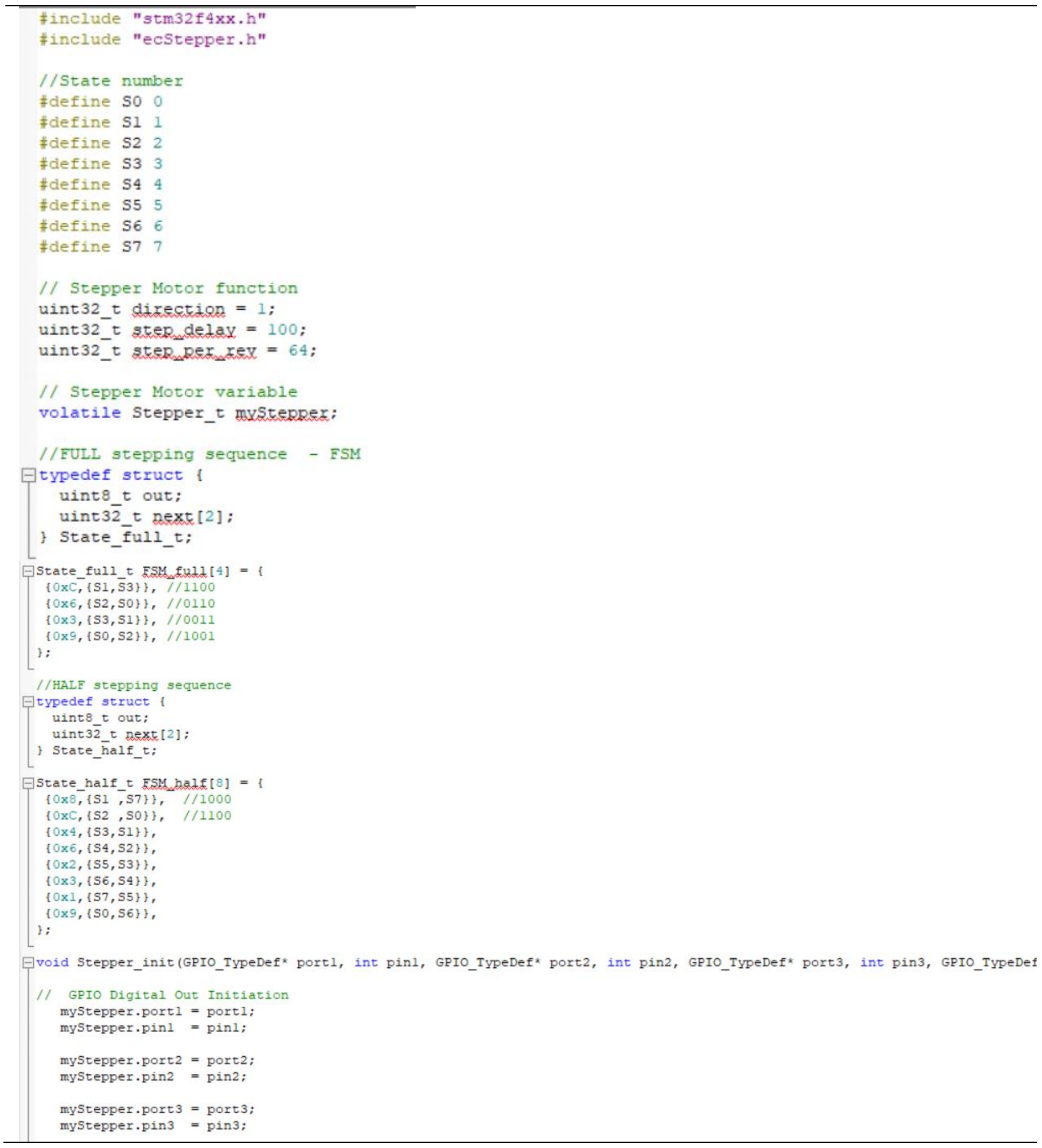
A. Since step_delay is an integer, it is impossible to calculate exactly the time to decimal place. Therefore, SysTick was converted into us(microseconds) units to calculate the time to the exact decimal place. Then, it was confirmed that it spins exactly once every six seconds.

Appendix

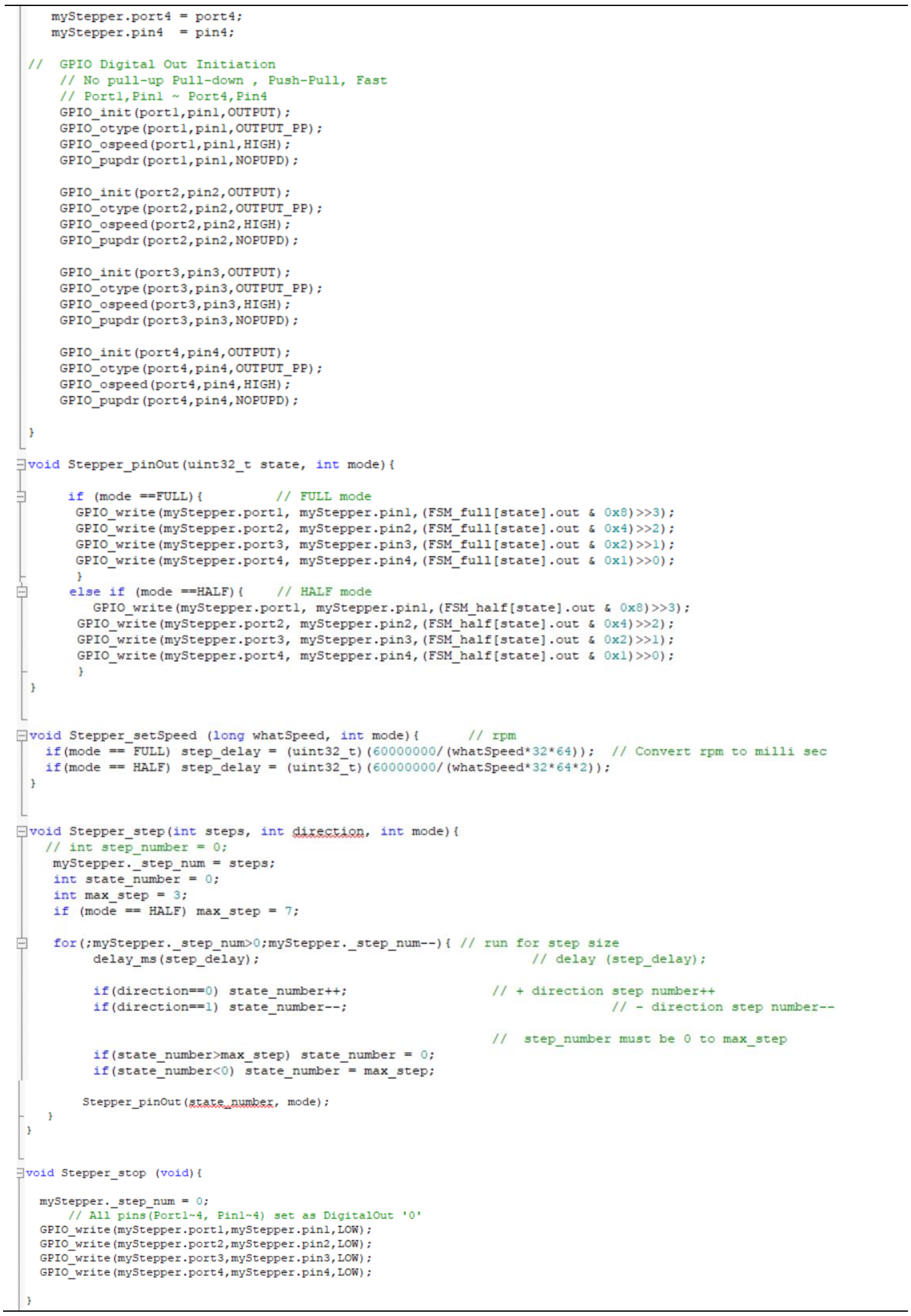
- **Video demo Link**

  Click here watch video.

- **Source code**

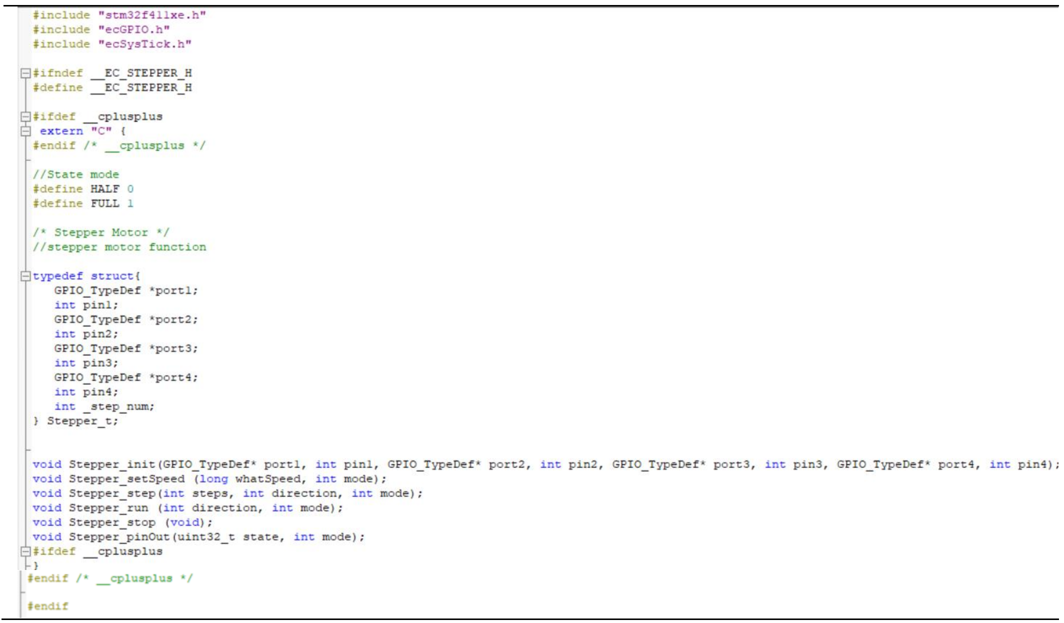


```
#include "stm32f4xx.h"
#include "ecStepper.h"

//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}}, //1100
 {0x6,{S2,S0}}, //0110
 {0x3,{S3,S1}}, //0011
 {0x9,{S0,S2}}, //1001
};

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

State_half_t FSM_half[8] = {
 {0x8,{S1 ,S7}},  //1000
 {0xC,{S2 ,S0}},  //1100
 {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, GPIO_TypeDef

//  GPIO Digital Out Initiation
   myStepper.port1 = port1;
   myStepper.pin1  = pin1;

   myStepper.port2 = port2;
   myStepper.pin2  = pin2;

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



```c
    myStepper.port4 = port4;
    myStepper.pin4  = pin4;

//  GPIO Digital Out Initiation
    // No pull-up Pull-down , Push-Pull, Fast
    // Port1,Pin1 ~ Port4,Pin4
    GPIO_init(port1,pin1,OUTPUT);
    GPIO_otype(port1,pin1,OUTPUT_PP);
    GPIO_ospeed(port1,pin1,HIGH);
    GPIO_pupdr(port1,pin1,NOPUPD);

    GPIO_init(port2,pin2,OUTPUT);
    GPIO_otype(port2,pin2,OUTPUT_PP);
    GPIO_ospeed(port2,pin2,HIGH);
    GPIO_pupdr(port2,pin2,NOPUPD);

    GPIO_init(port3,pin3,OUTPUT);
    GPIO_otype(port3,pin3,OUTPUT_PP);
    GPIO_ospeed(port3,pin3,HIGH);
    GPIO_pupdr(port3,pin3,NOPUPD);

    GPIO_init(port4,pin4,OUTPUT);
    GPIO_otype(port4,pin4,OUTPUT_PP);
    GPIO_ospeed(port4,pin4,HIGH);
    GPIO_pupdr(port4,pin4,NOPUPD);

}

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);
      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);
      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){      // rpm
   if(mode == FULL) step_delay = (uint32_t)(60000000/(whatSpeed*32*64));  // Convert rpm to milli sec
   if(mode == HALF) step_delay = (uint32_t)(60000000/(whatSpeed*32*64*2));
}


void Stepper_step(int steps, int direction, int mode){
   // int step_number = 0;
    myStepper._step_num = steps;
    int 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_ms(step_delay);                               // delay (step_delay);

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

                                                         //  step_number must be 0 to max_step
         if(state_number>max_step) state_number = 0;
         if(state_number<0) state_number = max_step;

        Stepper_pinOut(state_number, mode);
    }
}


void Stepper_stop (void){

   myStepper._step_num = 0;
      // All pins(Port1~4, Pin1~4) set as DigitalOut '0'
   GPIO_write(myStepper.port1,myStepper.pin1,LOW);
   GPIO_write(myStepper.port2,myStepper.pin2,LOW);
   GPIO_write(myStepper.port3,myStepper.pin3,LOW);
   GPIO_write(myStepper.port4,myStepper.pin4,LOW);

}
```

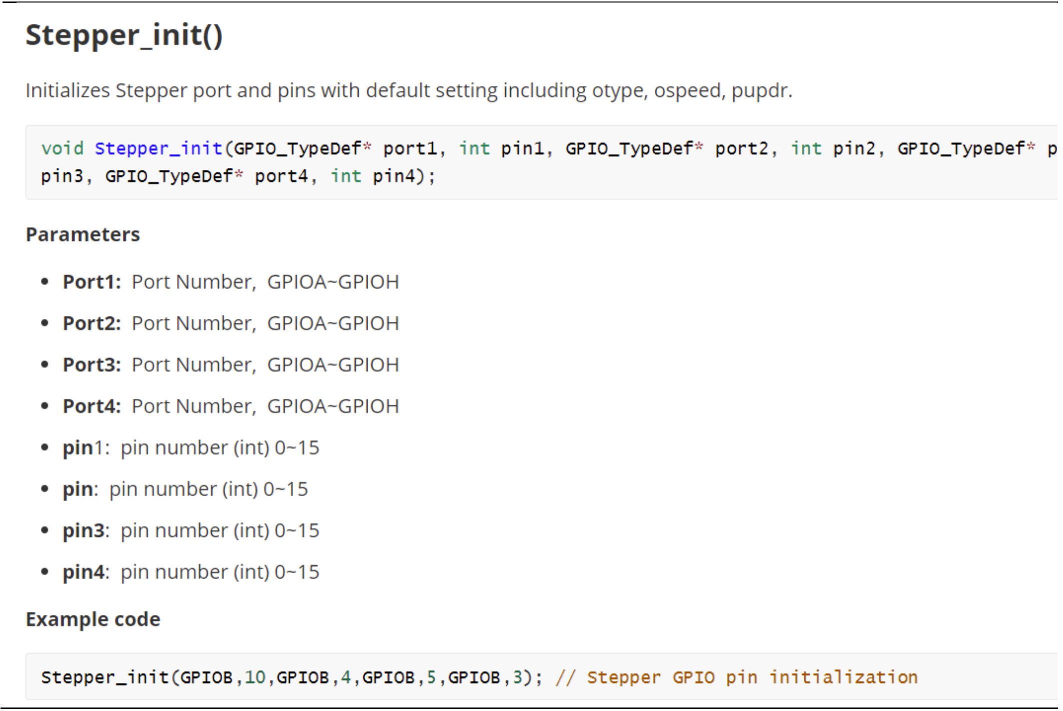
Figure 14. Stepper.c code



```
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecSysTick.h"

#ifndef __EC_STEPPER_H
#define __EC_STEPPER_H

#ifdef __cplusplus
 extern "C" {
#endif /* __cplusplus */

//State mode
#define HALF 0
#define FULL 1

/* Stepper Motor */
//stepper motor function

typedef struct{
   GPIO_TypeDef *port1;
   int pin1;
   GPIO_TypeDef *port2;
   int pin2;
   GPIO_TypeDef *port3;
   int pin3;
   GPIO_TypeDef *port4;
   int pin4;
   int _step_num;
} Stepper_t;


void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2, GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);
void Stepper_setSpeed (long whatSpeed, int mode);
void Stepper_step(int steps, int direction, int mode);
void Stepper_run (int direction, int mode);
void Stepper_stop (void);
void Stepper_pinOut(uint32_t state, int mode);
#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif
```

Figure 15. Stepper.h code

- **Documentation**



Stepper_init()

Initializes Stepper port and pins with default setting including otype, ospeed, pupdr.

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

Parameters

- **Port1:** Port Number, GPIOA~GPIOH
- **Port2:** Port Number, GPIOA~GPIOH
- **Port3:** Port Number, GPIOA~GPIOH
- **Port4:** Port Number, GPIOA~GPIOH
- **pin**1: pin number (int) 0~15
- **pin**: pin number (int) 0~15
- **pin3**: pin number (int) 0~15
- **pin4**: pin number (int) 0~15

Example code

```
Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // Stepper GPIO pin initialization
```

Stepper_setSpeed()

it can control stepper motor's speed.

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

Parameters

- wahtSpeed : RPM
- mode : FULL or Half mode

Example code

```
Stepper_setSpeed(18,FULL);     //  set stepper motor speed
```

Stepper_step()

Control the stepper motor according to the number of steps, direction, and mode.

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

Parameters

- step : step size determined by the user according to the mode.
- direction : CW(clockwise) , CCW(counterclockwise)
- mode : FULL or Half

Example code

```
Stepper_step(2048, 0, FULL);  // (Full Step : 2048, Half Step : 4096 Direction : 0 -> ++ or 1 -
FULL or HALF)
```

Stepper_stop()

When button is pressed, stepper motor stops immediately.

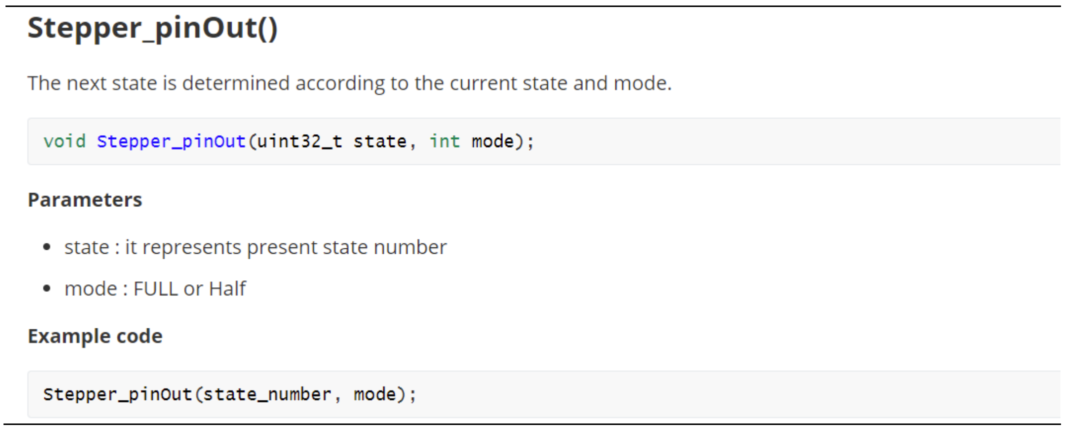
```
void Stepper_stop (void);
```

Parameters

- void

Example code

```
void EXTI15_10_IRQHandler(void) {
    if (is_pending_EXTI(BUTTON_PIN)) {
        Stepper_stop();
        clear_pending_EXTI(BUTTON_PIN);
    }
}
```



Stepper_pinOut()

The next state is determined according to the current state and mode.

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

Parameters

- state : it represents present state number
- mode : FULL or Half

Example code

```
Stepper_pinOut(state_number, mode);
```

Figure 16. documentation of Stepper