REPORT

Title : Automatic mini-fan

Submission Date	2021.09.27	Major	Mechanical and control engineering
Subject	Embedded Controller	Student Number	21700375
Professor	Young-Keun Kim	Name	Yeong-Won Song

Contents

I. Introdution

1.1 Purpose

In this experiment, we make a simple embedded digital application program using mbed API. The mbed is 32bit-ARM Cortex-M micro controller operating system. We intend to control the hardware sensor and output of NUCLEO-F411RE Board through the mbed OS programming conding.

1.2 Parts List

Parts	Specification	Quantity
NUCLEO -F411RE	Arm®(a) Cortex®-M4 with FPU	1
Ultrasonic distance sensor	HC-SR04),	1
DC motor	(RK-280RA)	1

Table 1. Part List

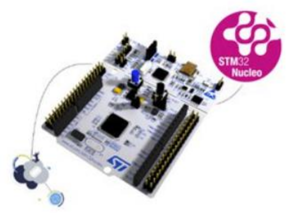
		
NUCLEO-F411RE	Ultrasonic distance sensor	DC motor

Table 2. experiment equipment

II. Experimental Method

2.1 Experimental condition

Condition 1.

As the button B1 is pressed, change the DC motor velocity

The mode is OFF(0%), MID(50%), HIGH(70%), V.HIGH(100%)

As the B1 is pressed, it should toggle from OFF mode to V.HIGH mode and so on

Condition 2.

Automatically ReStart and Stop the DC motor when the mode is

RESTART: The distance is within about 50mm

PAUSE : The distance is beyond about 50mm

Condition 3.

Print the distance and PWM duty ratio in Tera-Term console (every 1 sec).

Condition 4.

When the DC is turned OFF temporarily then turned on again depending on the distance as in Condition (2), it should be turning at the previous speed.

Condition 5.

When the mode is OFF, turn off the LED(LED1). Otherwise Blink the LED by 1 sec.

2.2 Mbed code

```
#include "mbed.h"

Ticker      tick; //Print ticker every 2sec
Ticker      tick1; //LED Blink ticker every 1sec
DigitalOut led(LED1);

int prints = 0 ;
void Printstate(){
        prints =1;
}


Serial          pc(USBTX, USBRX, 9600);
PwmOut          trig(D10); // Trigger 핀
InterruptIn echo(D7);    // Echo 핀
Timer           tim;

int begin = 0;
int end = 0;


PwmOut pwm1(D11);
InterruptIn    button(USER_BUTTON);
PwmOut pwm2(PC_8);

int pwm = 0;
int flag = 0;
int speedstate = 0;
int duty = 0;


void pressed()
{
        speedstate = speedstate + 1; //if button is pressed, update present state
                if(speedstate == 1)
            {
                pc.printf("MODE = MID\n");
                }
                if(speedstate == 2)
            {
                pc.printf("MODE = HIGH\n");
                }
                if(speedstate == 3)
            {
                pc.printf("MODE = VERY HIGH\n");
                }
                if(speedstate == 4)
            {
                speedstate = 0;
                pc.printf("MODE = OFF\n");
                }
}
```

```
void INT(){
	if(speedstate == 0){ //if DC-motor MODE is off, turn off LED
		led = false;
	} else {
	led = !led;
	}
}

void rising(){
	begin = tim.read_us();
}

void falling(){
	end = tim.read_us();
}

int main(void){
	float distance = 0;

	trig.period_ms(60);		// period		= 60ms
	trig.pulsewidth_us(10); // pulse-width = 10us

	echo.rise(&rising);
	echo.fall(&falling);

	tim.start();

	pwm1.period_ms(10);
	pwm2.period_ms(10);
	tick1.attach(&INT, 1);
	tick.attach(&Printstate, 2);
	button.fall(&pressed);

	while(1){

		distance =	(float)(end - begin) / 58; // [cm]
		wait(0.5);
		if(prints == 1){
			pc.printf("Distance = %.2f[cm]\r\n", distance);
			pc.printf("PWM duty ratio = %d[percent]\r\n", duty);
			prints = 0;
			}
		if(distance <50){ //if distance is less than 50cm, active DC-motor
			flag = 1;
			}
		else flag = 0;
		if(flag == 1){
				if(speedstate == 0){ // STATE 0% OFF
								duty = 0;
								pwm1.pulsewidth_ms(duty);

				}
				if(speedstate == 1){ // STATE 50% MID
								duty = 5;
								pwm1.pulsewidth_ms(duty);
```

```
                }
                 if(speedstate == 2){ // STATE 70% HIGH
                                    duty = 7;
                                    pwm1.pulsewidth_ms(duty);


                }
                 if(speedstate == 3){ // STATE 100% V.HIGH
                                    duty = 10;
                                    pwm1.pulsewidth_ms(duty);


                }
            }
            else    pwm1.pulsewidth_ms(0);
        }
}
```

- **DC-Motor**

After the DC motor is activated, the speed of the DC motor must be controlled by button. DC motor's velocity can consist of four states.

SpeedState	Velocity ratio	LED
0(OFF)	0%	Off
1(MID)	50%	On
2(HIGH)	70%	On
3(V.HIGH)	100%	On

Table 3. Speed State

Whenever User-button or button B1 is pressed, the speed state must move the next state. In the ongoing DC motor velocity state, the speed state should be changed every time a button is pressed, so the Interruptin command was used, not digitalin. Because the Interruptin command was used, When the DC motor is turned off and turned on again, the DC motor is operated again in the same speed state before turning off. Whenever the button was pressed, the void pressed () function was executed, and the variable speed state value was added 1 in the function. Then, each time a button is pressed, the speed state value increases by 1, and the speed of the DC motor can be controlled according to the speed state value through the conditional statement. If button is pressed again in the last speed state = 3, the speed state should be backed to 0, that is, the speed of the DC motor must be 0. Therefore, when the speed state value is 4 in the void pressed () function, a condition statement was added so that the speed state returned to the initial state where speed state is zero when the button was pressed in the speed state=3.

- **Ultrasonic distance sensor**

First, when ultrasonic distance sensor was used to allow the DC motor to operate when an object was identified within 50cm. The distance can be determined by measuring the original wave and the time of the reflected wave from the object. The continuously measured distance value was set as a distance variable, and when an object was recognized within 50cm, the flag variable value was given 1 to allow the DC motor to operate within the if condition statement. If the distance value was greater than 50cm, flag variable value was given zero to prevent the DC motor from operating.

- **Print PWM and Duty-raito**

The current distance value and duty ratio should be output every 2 seconds. Ticker was used to meet the condition of once every 2 seconds. By defining a void function called Printstate, the value 1 was fixed to the prints variable in the function. The Printstate function was executed with tick.attach (&Printstate, 2), and in the if condition statement, print=1, the distance value w from the main statement.

- **Blink LED**

When the DC motor was operated, the LED was blinking once a second using a ticker. When DC motor is not operated, the LED should be turned off. The LED value is given a false value in the void INT() function so that the LED is turned off. If speed state is not zero, the LED is blinking.

2.3 Flow chart

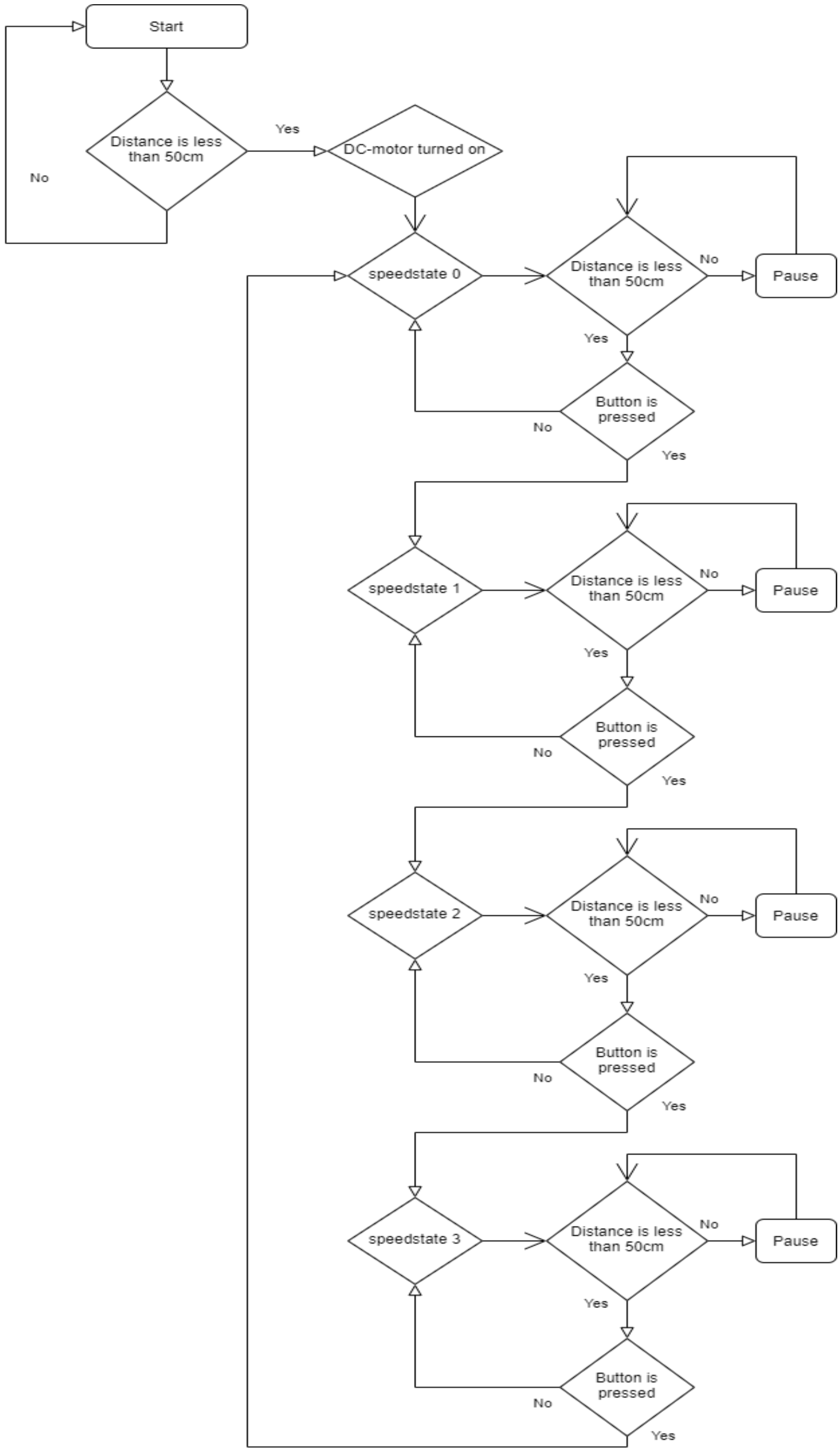


Figure 1. Algorithm Flow Chart

III. Experimental Result

3.1 Result Validity

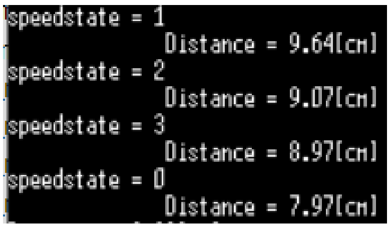
SpeedState	Velocity ratio	LED	Print distance & duty ratio	Validity
0(OFF)	0%	Off	2sec	O
1(MID)	50%	On(1sec)	2sec	O
2(HIGH)	70%	On(1sec)	2sec	O
3(V.HIGH)	100%	On(1sec)	2sec	O

Table 4. Result check

The operation of the DC-Motor was determined according to the recognized distance value of the object. It was confirmed that the LED blinks once a second when the DC motor is operating or in the Pause state. It was checked whether the distance value and duty ratio were properly output every 2 seconds.

```
MODE = OFF
         Distance = -1028.34[cm]
PWM duty ratio = 0[percent]
Distance = -756.41[cm]
PWM duty ratio = 0[percent]
MODE = MID
         Distance = 4.10[cm]
PWM duty ratio = 5[percent]
Distance = 5.40[cm]
PWM duty ratio = 5[percent]
MODE = HIGH
          Distance = 4.14[cm]
PWM duty ratio = 7[percent]
Distance = 4.10[cm]
PWM duty ratio = 7[percent]
MODE = VERY HIGH
               Distance = -760.02[cm]
PWM duty ratio = 10[percent]
Distance = 4.41[cm]
PWM duty ratio = 10[percent]
```

Figure 2. Tera-term Print



```
speedstate = 1
               Distance = 9.64[cm]
speedstate = 2
               Distance = 9.07[cm]
speedstate = 3
               Distance = 8.97[cm]
speedstate = 0
               Distance = 7.97[cm]
```

Figure 3. Check speed state changing

It was confirmed whether the DC motor was operated in the previous speed state when it was temporarily turned off and then restarted. Figure 4 confirmed whether the Speed state was well changed to the next state.

IV. Conclusion

Through this experiment, we were able to determine the correlation between software and hardware. I was able to understand the operating principles of each sensor and learn how to operate the sensor by program coding. In addition, it was possible to control the speed of DC-Motor and LDE Blink in hardware through software instructions. Based on the understanding of the flow of digital logic, it was possible to define the state and implement it in coding on the mbed OS program.