California State University, San Marcos

Embedded Systems Final Report

DC Motor Control Using Pulse Width Modulation

Joshua Wilber
Andre Bruno
Boyang Fan
December 14th, 2017

Abstract

While thinking of an idea for our final project, we didn't want to do what other groups had done previously, such as a piano, and wanted to do something a little different. Then the idea came up: why not incorporate a DC motor into our project? A DC motor can be used in many applications, ranging from toys, such as remote-controlled cars, to household appliances, such as a vacuum cleaner, and even for industrial use, such as conveyor belts and elevators. For our project, we decided to use our DC motor as a fan that will have two modes: a manual mode, which allows the user to manually adjust the speed of the fan, and an automatic mode, which adjusts the speed of the fan based on the ambient temperature that is read from the temperature sensor. Fans with varying speeds can be used in many applications, such as desktop computers and cars.

Introduction

As previously stated in the abstract, our final project is a fan that is driven by a DC motor with variable speed, which can be manually adjusted via buttons, or automatically adjusted based on the ambient temperature that is read by the temperature sensor. While doing research on how to adjust the speed of DC motors, we found out that there are two methods. The first method, and initial method that we considered, is controlling the voltage. However, we learned that if the voltage varies in a DC motor, a lot of power can be dissipated, which could lead to the motor stalling due to a loss of torque. Due to the inefficiency of varying the voltage, we opted for method two: pulse width modulation, or PWM for short. When a DC motor is controlled via PWM, the full rated voltage is supplied, resulting in no torque loss and stalling issues, and the speed of the motor is controlled by changing the duty cycle, while the frequency is set at a constant value. It is recommended that the frequency of the PWM be set to a frequency that is out of the audible range, so we have ours set at 22KHz.

With that being said, we have a total of five buttons. One button is used to switch to manual mode, and another button is used to switch to automatic mode. The other three buttons are used for manual mode, and change the speed of the fan by changing the duty cycle of the PWM. There is a button for 30%, for 50%, and for 75%, ranging from slowest to fastest. When the automatic

mode button is pressed, the system switches to automatic mode. In automatic mode, the buttons used for manual mode are unable to be used, and instead reads the ambient temperature via the temperature sensor. We coded it so that if the temperature is less than or equal to 20℃ (68℉), the fan will automatically be set to 30%. If the temperature is greater than 20℃ (68℉), then the fan will automatically be set to 75%.

Along with the buttons, there are two LEDs wired to the board as well. One of the LEDs is used to indicate which mode the system is in. The LED is off if it is in manual mode, and it will glow red if it is in automatic mode. The other LED indicates what the fan speed is set to. This LED glows green at 30%, blue at 50%, and red at 75%.

While we have the two LEDs to indicate what mode the system is in and what speed the fan is set to, we also have information displayed on a terminal using the UART protocol. When the system is in manual mode, the following is displayed in the terminal: “Manual Mode - Fan Speed = X%”, where X is the current speed of the fan. In automatic mode, the following is displayed: “Automatic mode - TEMPERATURE = X, Fan Speed = Y%”, where X is the ambient temperature in ℃, and Y is the current speed of the fan.

Conceptual Design

The conceptual design of our system is relatively straight-forward. A circuit with several different inputs that power our one output. To begin, the heart of project revolves around the DC motor. The motor is our one output that controls the on/off and speed of the fan. The various inputs into the circuit controls the output or speed of the fan. External or internal vcc can power the circuit. There are a total of 5 inputs, 4 buttons that connect to the input pins on the Nucleo board and also a temperature sensor that reads the current temperature in Celsius. Three buttons to control the fan at a manual fixed speed and also 1 button to put the circuit in auto mode which reads in the temperature and turns on depending on the temperature.

Implementation of Code (Flow chart showing main routines)

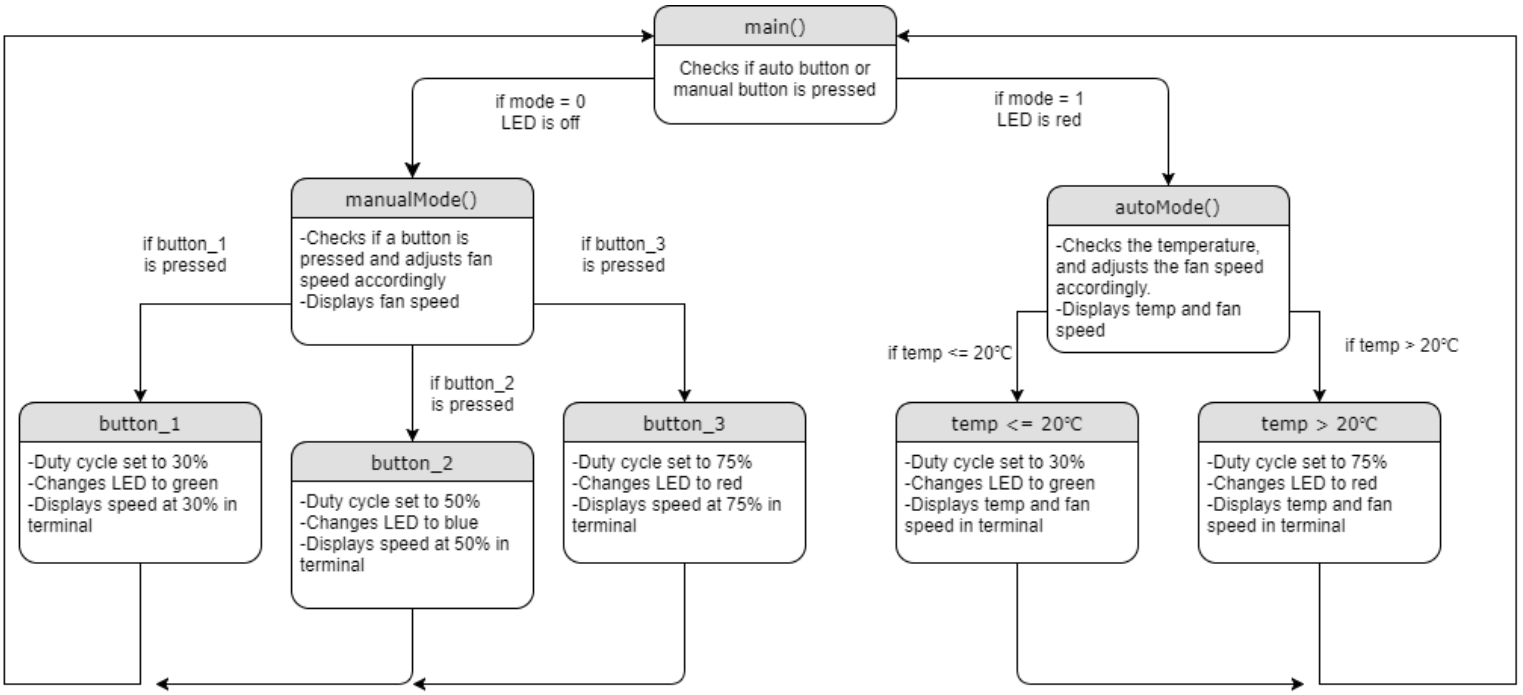

Figure 1: this flow chart illustrates the procedural nature of the routines we coded.

Main Challenges and Testing

The main challenges pertaining to this project were the circuit design and the implementation of the code. Figure 2 below is an image of the breadboard and the circuit we designed to run our code.

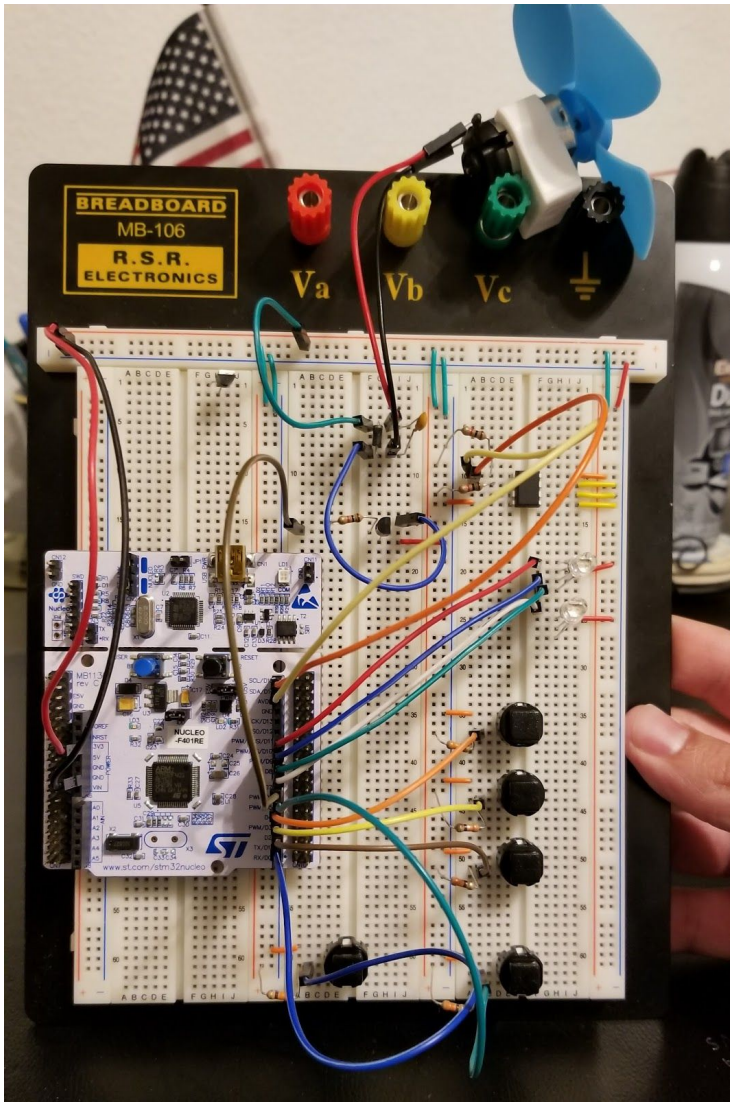

Figure 2: The circuit design is of course controlled by our STM32 Nucleo

The buttons, the temperature sensor and the LEDs we used are the same as the ones we used in our labs throughout the semester and they were wired exactly the same way. The most challenging part was coming up with a way to drive our DC motor using pulse width modulation

without damaging the board or the motor. For this we picked a transistor circuit. The PMW port is wired to the base of the transistor which activates it via a small current which is regulated by the resistor. This allows a current, much larger than the one applied to the base, to flow from the collector to the emitter. The DC motor is hooked up in parallel with a Schottky Diode and a small 100 pF capacitor with the positive ends wired to the board's 3.3V output and the negative ends wired to the collector. The current is then allowed to flow from the emitter to the ground pin. The diode is used to protect the transistor from the current build up in the DC motor. The motor, acting as an inductor, generates a current that keeps flowing when the PWM pulse goes low. The diode can absorb this current without suffering any damage. Some research had to be done to figure out which diode was most appropriate in this circuit. We had originally used a common N4001, but it turns out that it is not capable of handling the frequency of our pulse wave. This led to several transistors burning out and ceasing to work. The final choice was the 1N5819 which has a much faster recovery rate and can safely handle the quick on and off transitions. For a very similar reason, it took some research to pick an appropriate capacitor. At high frequencies, it is best to use a small capacitor, such as the 100 pF component stated above. This capacitor is used to reduce the high speed voltage transitions thus managing the magnitude of current flowing through the DC motor. This concludes the testing and challenges that we encountered in the electronics portion of our assignment.

Summary

All in all, this project has been a great learning experience and fun for us. We used what we had learned from the previous labs in this class and applied it to our final project, which allowed us to get a better understanding of all the concepts that this course outlines. While we did encounter some difficulties during the project especially via debugging the circuit, nothing serious affected our desired outcome. In the end, this project contributed a great deal to our learning experiences and we hope we can apply these concepts that we have learned to our future.

Reference/Bibliography

April 13, 2016, Pulse Width Modulation Vs Variable Voltage Control, https://www.actuatorzone.com/blog/about-actuators/pulse-width-modulation-vs-variable-voltage-control-2/ (December 11, 2017).

Appendix

```
//Final project: DC motor with fan
//Has 2 modes: Manual mode and auto mode. Manual mode allows the
//user to manually adjust the speed of the fan by pressing one of the
//buttons (for 30%, 50%, 75% fan speed).
//While auto mode takes in the temperature, and adjusts the speed of the fan according
//to the temperature.

#include "mbed.h"
#include "pindef.h"


//Define buttons
DigitalIn button_1(PB_4);          //Duty cycle = 30%
DigitalIn button_2(PB_5);          //Duty cycle = 50%
DigitalIn button_3(PB_3);          //Duty cycle = 75%
DigitalIn button_4(PB_10);         //Button to switch to auto mode
DigitalIn button_5(PA_10);         //button to switch to manual mode

//Define LEDs
DigitalOut red1(PA_9);             //auto mode on when red1 is high
DigitalOut green(PC_7);            //green is on when motor is at 30% speed
```

```
DigitalOut blue(PB_6);          //blue is on when motor is at 50% speed
DigitalOut red2(PA_7);          //red2 is on when motor is at 75% speed

// Serial tx, rx connected to the PC via an USB cable
Serial device(UART_TX, UART_RX);

//Define the PWM motor
PwmOut motor(PA_8);

//Define the temperature sensor
I2C temp_sensor(I2C_SDA, I2C_SCL);

//I2C address of temperature sensor DS1631
const int temp_addr = 0x90;

char cmd[] = {0x51, 0xAA};
char read_temp[2];

//Define variables
float motorDutyCycle = 0.0;
int mode = 0;                           //0 for manual mode; 1 for auto mode
int fanSpeed = 1;               //Fan speed: 0 = 30%, 1 = 50%, 2 = 75%
float temp = 0.0;               //Variable for temperature
float previousTemp = 0;
int lastButtonState = 1;
int lastButtonState_1 = 1;
int lastButtonState_2 = 1;
int lastButtonState_3 = 1;
```

```
int lastButtonState_4 = 1;

int reading;

//Function for manual mode
void manualMode()
{
	red1 = 0;
	//Button 1 - Set speed to 30%
	lastButtonState_1 = button_1;
	if(!lastButtonState_1)
	{
		wait(0.4);
		reading = button_1;

		if((reading != lastButtonState_1) && (fanSpeed != 0))
		{
			fanSpeed = 0;
			motorDutyCycle = 0.3;
			motor = motorDutyCycle;
			green = 1;
			blue = 0;
			red2 = 0;

			device.printf("\033[HManual Mode - Fan Speed = %6.2f%%\n", motorDutyCycle * 100);
		}
	}
```

```
        //Button 2 - Set speed to 50%
        lastButtonState_2 = button_2;
        if(!lastButtonState_2)
        {
                wait(0.4);
                reading = button_2;

                if((reading != lastButtonState_2) && (fanSpeed != 1))
                {
                        fanSpeed = 1;
                        motorDutyCycle = 0.50;
                        motor = motorDutyCycle;
                        green = 0;
                        blue = 1;
                        red2 = 0;

                        device.printf("\033[HManual Mode - Fan Speed = %6.2f%%\n",
motorDutyCycle * 100);
                }
        }

        //Button 3 - 75%
        lastButtonState_3 = button_3;
        if(!lastButtonState_3)
        {
                wait(0.4);
                reading = button_3;

                if((reading != lastButtonState_3) && (fanSpeed != 2))
```

```
                {
                        fanSpeed = 2;
                        motorDutyCycle = 0.75;
                        motor = motorDutyCycle;
                        green = 0;
                        blue = 0;
                        red2 = 1;

                        device.printf("\033[HManual Mode - Fan Speed = %6.2f%%\n", motorDutyCycle * 100);
                }
        }



}

//Function for auto mode
void autoMode()
{
        red1 = 1;
        temp_sensor.write(temp_addr, &cmd[0], 1);
        wait(0.8);
        temp_sensor.write(temp_addr, &cmd[1], 1);
        wait(0.1);
        temp_sensor.read(temp_addr, read_temp, 2);

        //Convert temperature to Celsius
        temp = (float((read_temp[0] << 8) | read_temp[1]) / 256.0f);
```

```
            if(temp <= 25.00f)                          //If temp is less than or equal to 20C = 68F
                {
                    motorDutyCycle = 0.3f;
                    motor = motorDutyCycle;     //Set the duty cycle to 30%
                    green = 1;
                    blue = 0;
                    red2 = 0;
                }
                else if(temp > 25.00f)              //If temp is greater than 20C = 68F
                {
                    motorDutyCycle = 0.75f;
                    motor = motorDutyCycle;     //Set the duty cycle to 75%
                    green = 0;
                    blue = 0;
                    red2 = 1;
                }
                if (temp != previousTemp)
                    device.printf("\033[HAutomatic Mode: TEMPERATURE = %.2f, Fan Speed = %.2f%%\n\r", temp, motorDutyCycle * 100);
                previousTemp = temp;


}
/*---------------------------------------------------------------------------
 MAIN function
 *---------------------------------------------------------------------------*/
```

```
int main(){

        motor.period(0.000045);                 //~22KHz
        motor = motorDutyCycle;

        while(1)
        {

                lastButtonState = button_4;

                if(!lastButtonState)
                {
                        wait(0.4);
                        reading = button_4;
                        if(reading != lastButtonState)
                        {
                                mode = 1;
                        }
                }


                lastButtonState_4 = button_5;

                if(!lastButtonState_4)
                {
                        wait(0.4);
                        reading = button_5;
                        if(reading != lastButtonState_4)
```

```
                {
                        mode = 0;
                }
        }


        if(mode == 0)
                manualMode();

        else if(mode == 1)
                autoMode();

    }//End while

}

// ********************************ARM University Program Copyright © ARM Ltd 2016*********************************
```