SCHOOL OF MECHATRONIC SYSTEMS ENGINEERING
SIMON FRASER UNIVERSITY

MSE 352:
Digital Logic & Microcontrollers

Final Project Report

December 7, 2022

Group 10

Chris Yee	301411599
Jackson Hamanishi	301362183
Eileen Ha	301369053

Introduction

For our MSE 352 project, we will control a DC brushless axial fan using a PWM signal created by a mikronBoard 8051 microcontroller. The servo motor will have a built-in encoder which will measure the speed of the fan and be shown onto three 7-segment displays. A 3-bit input from a DIP switch will control the motor speed as a percentage of the maximum speed by a pulse width modulation (PWM) signal. The maximum speed of the motor is 2400 RPM, and the clock frequency of the microcontroller is 11.0592 MHz.

Design Objective

Our first objective of this project is to read the reference speed from a DIP switch using a GPIO input in a microcontroller. A combination of DIP switches will set the percentage of the maximum speed, which will be set in the output. The sequence of DIP switch combinations are shown in Table 1.

Table 1: Combination of DIP Switch for Setting Speed

MSB	Middle Bit	LSB	RPM	Speed Percentage
0	0	0	0	0%
0	0	1	240	10%
0	1	0	720	30%
0	1	1	960	40%
1	0	0	1200	50%
1	0	1	1680	70%
1	1	0	1920	80%
1	1	1	2400	100%

The second objective is to read the encoder output to measure the motor speed in RPM by using external interrupts and internal timers. This speed should be shown on 3 7-segment displays and will only display the 3 most significant bits of the speed output.

Our last objective is to use a proportional controller to create a Pulse Width Modulation, PWM, signal that controls the speed of the motor with respect to the DIP Switch combinations shown in Table 1.

Materials

Materials:

- AT89S8253 40-pin microcontroller
- AUB0812L-9X41 DC motor
- One IRF540-st N-channel MOSFET transistor
- Three 74LS47 BCD to 7-segment decoders
- Three 7-segment displays (common-anode)
- One DIP switch
- Resistors: 1 kΩ, 10 kΩ, and two 1 MΩ

Hardware:

- A 5V and a 12V DC power supply
- Oscilloscope

Software:

- Keil µVison5 (C-programmer)
- Proteus 8 Professional (circuit simulation software)
- Mikroprog 8051 flash programmer (8051 uploader)

Circuit Diagram & Pin Layout

A general idea of the design is shown in Figure 1. To connect the motor to the PWM, we were provided with a circuit diagram shown in Figure 2. The chosen value of each of the resistors are shown in Figure 2.

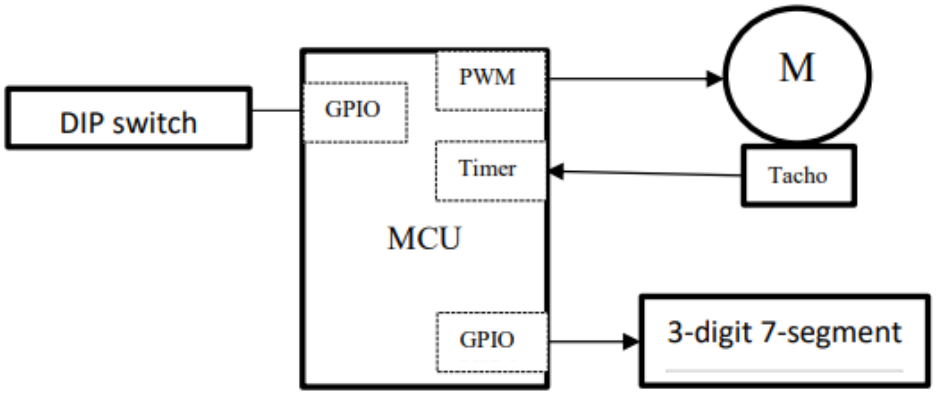

Figure 1: Concept of Design

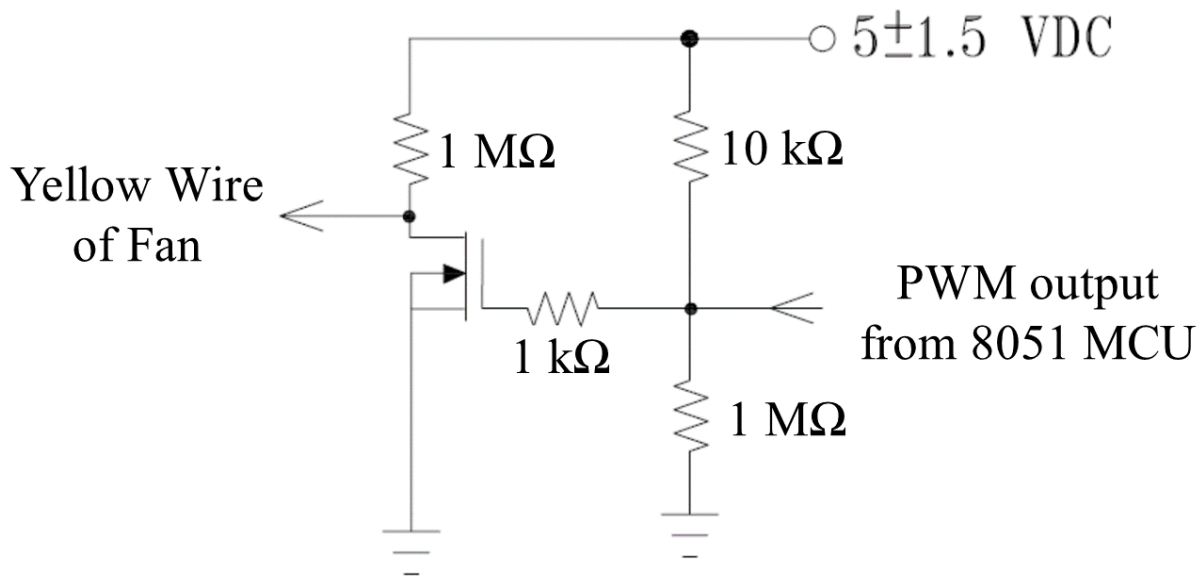

Figure 2: MOSFET Connection

Using our basic knowledge of 8051 from laboratories and lectures, we simulated the circuit design in Proteus 8 as shown in Figure 3. From our simulations, we can see that the circuit functions properly with 3 BCD decoders, external interrupt, and the provided MOSFET diagram. Thus, our pin layout for the microcontroller is shown in Table 2.

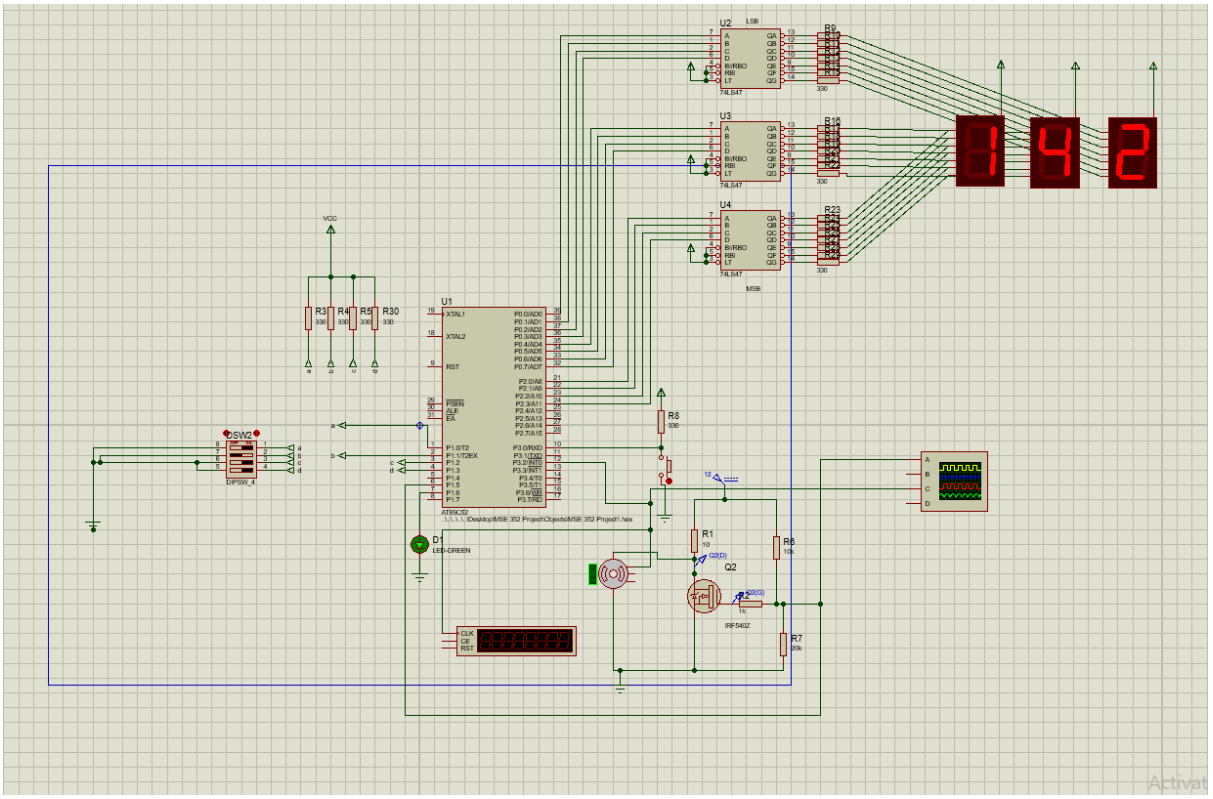

Figure 3: Simulated Circuit Diagram

Table 2: Pin Layout of Microcontroller

Function	Ports and Pins
DIP switch input	Port 1 – pins 0, 1, and 2
PWM output	Port 1 – pin 7
Encoder counter	Port 3 – pin 2 (INT0)
LSB BCD to 7-segment display	Port 0 – pins 0 to 3
Middle BCD to 7-segment display	Port 0 – pins 4 to 7
MSB BCD to 7-segment display	Port 2 – pins 0 to 3

Design Process

We first started by constructing our circuit, as shown in Figure 3.

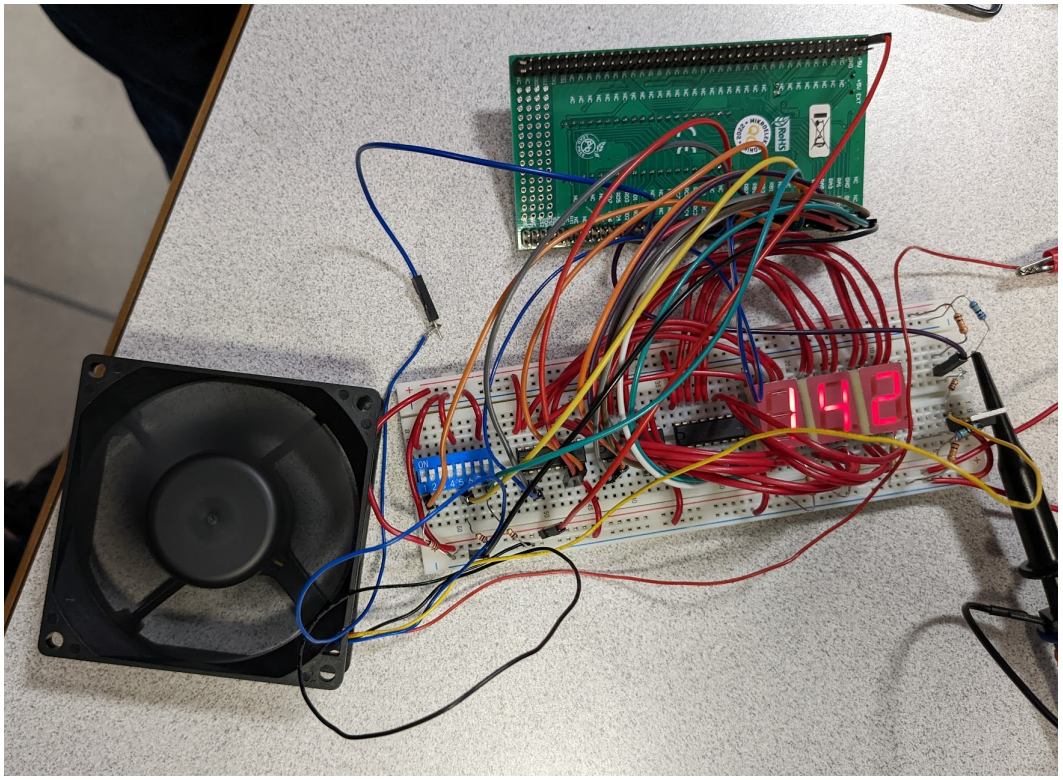

Figure 4: Physical Circuit

Next, we obtained the RPM of the fan by using an external interrupt to count the number of falling edges from the fan's encoder signal. Referring to the data sheet we found that one rotation of the motor equates to two complete periods from the encoder. Using the external interrupt, we are able to count the number of half rotations. Using timer 1, we can determine the number of half rotation after a given amount of time. Timer 1 was configured to mode 1 (16-bit no auto reload), therefore the maximum amount of time for one timer cycle is approximately 71.1 ms given that the clock frequency is 11.0592 MHz.

$$\begin{aligned}
&\frac{12 \text{ Clock Cycle}}{1 \text{ Machine Cycle}} \times \left(\frac{1}{11.0592 \text{ MHz}}\right) \frac{1}{\text{Clock Cycle}} \times \frac{2^{16} \text{ Machine Cycles}}{1 \text{ Timer Cycle}} \\
&= \frac{1.805 \text{ µs}}{1 \text{ Machine Cycle}} \times \frac{2^{16} \text{ Machine Cycles}}{1 \text{ Timer Cycle}} \\
&= 71.1 \text{ ms per Timer Cycle}
\end{aligned}$$

Using the full length of the timer, we can loop the timer 7 times to obtain approximately 500 ms. We can then calculate the RPM of the motor by taking in the number of falling edges in each 500 ms interval.

$$\frac{n \text{ falling edges}}{500 \text{ ms}} \times \frac{1 \text{ rotation}}{2 \text{ falling edges}} \times \frac{1000 \text{ ms}}{1 \text{ s}} \times \frac{60 \text{ s}}{1 \text{ min}} = 60n \text{ RPM}$$

Since we only have three 7-segment displays, we will only be showing the three most significant digits of the RPM. The value to be displayed can be obtained simply from dividing the RPM value by 10, thus resulting in the output to be equal to $6n$, where n is the number of falling edges from the fan. The modulo operator can then by applied to the output to obtain each digit. Each digit is then sent to their respective pins connected to displays, as stated in Table 2.

We then created a PWM where the duty cycle was controlled by the dip switches. To create the duty cycle we obtained the input from 3 pins which are connected to the microcontroller. The duty cycle is determined through toggling the dipswitch pins on and off. The duty cycle for each DIP switch configuration is shown below in Table 3:

Table 3: Combination of DIP Switch for Setting Duty Cycle

MSB	Middle Bit	LSB	Duty Cycle
0	0	0	0%
0	0	1	10%
0	1	0	30%
0	1	1	40%
1	0	0	50%
1	0	1	70%
1	1	0	80%
1	1	1	100%

As stated in Table 2, the DIP switch is connected to pin 0 to 2 on port 1. We simply read only the first three bits of port 1 to obtain the input value of the switch. Given the duty cycle from the DIP switch input, we can create the PWM by setting the output pin high at the duty cycle percentage and low for the rest of the time. For example, if the duty cycle is 40%, the output pin is set to high and the timer will start at 40% progress; once the timer is completed, the output is set to low and the timer will start again but at 60%.

Finally, we replace the duty cycle with the proportional controller to control the speed of the fan. In the code, we changed the DIP switch input to now control the desired speed of the fan rather than the duty cycle (see Table 1). The proportional control is mathematically expressed as:

$$x = K_p \cdot e + x_0 = K_p \cdot (y_d - y) + x_0$$

where *x* is the output of the proportional controller, x_0 is the controller output with zero error, K_p is the proportional gain, e is the error, y_d is the desired RPM, and y is the obtained RPM. The proportional controller changes the value of the duty cycle as the RPM of the fan approaches the desired speed.

Conclusion

Our circuit was able to control the speed of the fan using the DIP switches and display the RPM on 7-segment displays, thus successfully accomplishing all three objectives.

The DC power supplies used in the lab struggle with the fast-changing current drawn from the DC motor as the PWM signal changes. As such, the proportional controller makes the motor speed fluctuate around the desired speed. This fluctuation is roughly ±20 RPM, but if taken the average across the values that are displayed, it will result in the desired motor speed. To smoothen out the output, a more sophisticated PID controller could be developed to help reach the desired motor speed at steady state. Using the provided power supplies from the 110 lab we were able to determine that in order for our circuit to operate below 50% duty cycle a faster switching power supply is needed to hold a stable rpm value. This is because when the fan speed is over our desired RPM the current is dropped down until it reaches the desired target RPM. While dropping the power supply is not able to switch fast enough while being precise to hold the proper current to obtain the correct fan speed.

Contribution

Jackson Hamanishi
- Code development and debugging
- Project report

Eileen Ha
- Physical circuit assembly
- Code debugging
- Project report

Chris Yee
- Circuit design and simulation on Proteus
- Physical circuit assembly
- Code debugging
- Project report