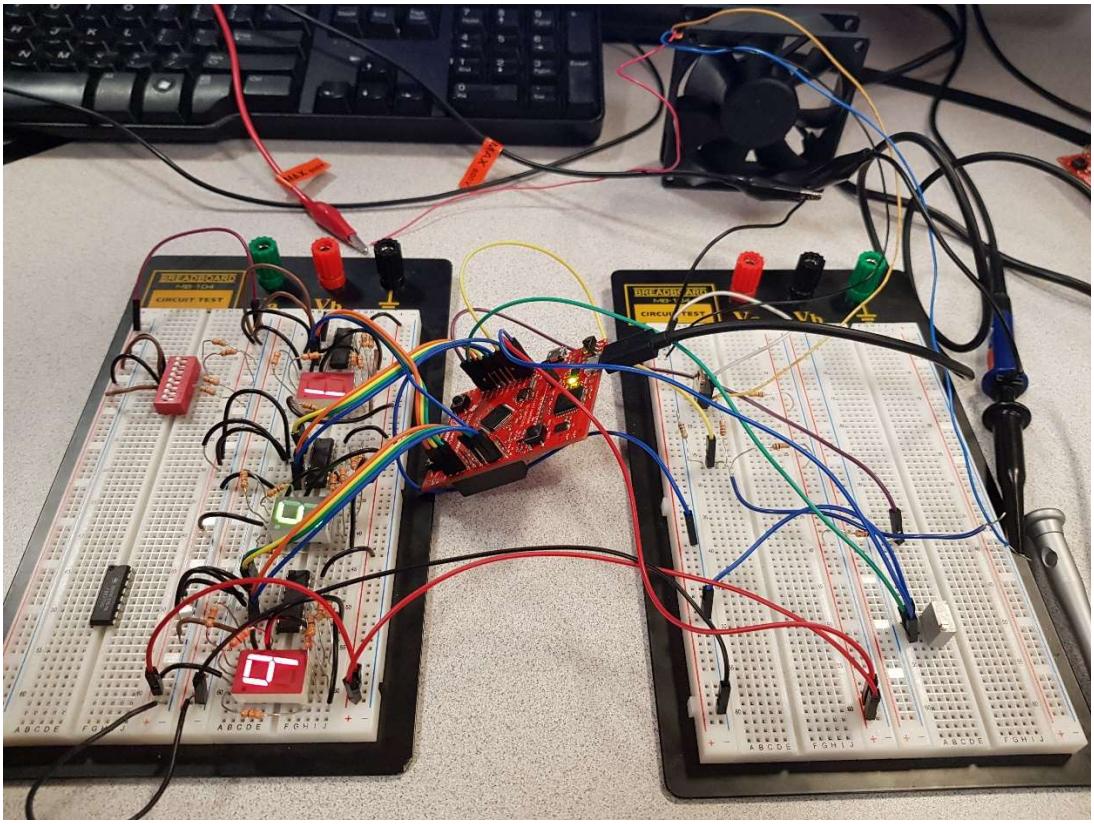
MSE352: Project Report

DESIGN OF A SERVO DC MOTOR SPEED CONTROLLER USING ARM MCU



December 4th, 2018

Written By:

Nam Anh Mai 301326816

Kai Chun (Andy) Yang 301263031

Christine Teodoro 301101505

Abstract

The main objectives here is to program a proportional controller to generate PWM signal to control the speed of a Servo DC motor and display the Revolutions Per Minute (RPM) on 3-digit 7-segment. A Tiva C Series TM4C123G LaunchPad is a microcontroller unit (MCU), which is used in this project. Then, Code Composer Studio (CCS) is a software to program the MCU. Additionally, the model of Servo DC motor is AUB0812L-9X41 from Delta Electronics. The MCU can output PWM signal to the Servo DC motor and get closed loop feedback signal from the Servo DC motor by programming Analog-to-Digital Converter (ADC) in the MCU. Servo DC motor can be controlled by changing resistance in potentiometer. Then, the result of the speed is displayed on to three BCD to 7-segment Display Decoders. Overall, the control and feedback system operate well as expected.

Introduction

Direct Current (DC) motor speed can be varied over a wide range by encoding analog signal through Pulse Width Modulation (PWM) or input voltage by potentiometer, and the implementation of a proportional controller will make the desired speed more obtainable. In this project, Code Composer Studio (CCS) is used for writing codes, a Tiva C Series TM4C123G LaunchPad is used as MCU, a DC brushless axial flow fan AUB0812L-9X41 from Delta Electronics is used as servo DC motor, three BCD connected to three 7-segments for speed display, as well as other components like resistors, potentiometer, N-MOSFET, and breadboards.

Project Description

The MCU needs to generate PWM signal to the Servo DC motor and be able to proportionally control the speed of the motor. Then, the Servo DC motor encoder should give a closed loop feedback signal from the motor shaft speed, which is also known as tachometer. From there, MCU must be set up to run a timer to start counting the edges of digital signal that is converted by ADC from Servo DC motor to MCU. In order to calculate the speed of the motor, the time is recorded for every three edge changes because every three-edge change is one complete period of the feedback signal. Afterwards, the speed of the motor is determined from the average of three continuous samples. Next, another ADC in MCU is set up to read the reference speed from the potentiometer. The reference speed is then compared to the actual speed to find the error source. With the error source, the speed reading is adjusted to displayed on 3-digit 7-segment.

Design Reflection

CCS Code Design

1) Description

The general structure of the code comprises of library headers followed by declaration of global variables, constants, supporting functions, and a main function with an infinite loop running to control signal to the motor. The code is developed by following the project description step by step as written in the previous section. In this case, program can be tested for specific task to achieve an objective. In the beginning of the main function, configuration and initialization have been coded for clock, ports and pins, ADC, and AIN. Then, PWM signal is generated with desired duty cycle, and ADC is utilized for recording the input signal from analog to digital. Afterwards, the program runs into an infinite loop. Within the infinite loop, PWM is adjusted for upper limit and lower limit. Moreover, MCU is programmed for ADC conversion for signals from Servo DC motor and potentiometer. Next, a timer is set up to count the period of the motor for every three-edges change from the signal. The speed value is then manipulated with error calculated from reference speed to make a correct reading for the speed. Finally, the value for the speed is displayed to 3-digit 7-segement. The fan speed can then be controlled by manually turning the potentiometer knob with the numbers displayed on the 7 segments.

2) PWM

The clocks, peripherals and PWM signal is configured on General Purpose Input/ Output (GPIO) pins. Next, the pins PF0 and PF4 will be unlocked from original functions and reconfigured to be used as pad buttons for duty control manual modification. The PWM input signal is controlled by changing different values for ui32Adjust, which is the DC line that changes time on and off in a period. The Figure below shows the conversion from Timer signal to PWM signal [2].

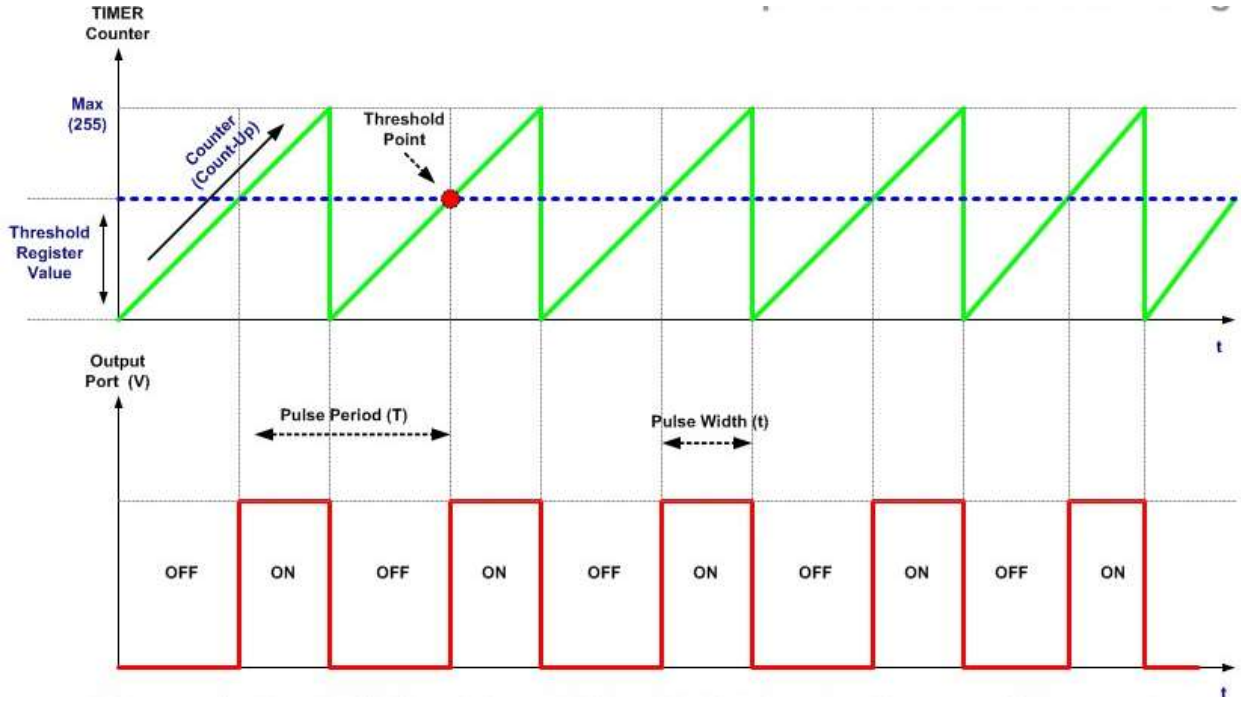


Figure 1: PWM Peripheral using Comparator (DC line)

In case of over or under adjustment, constraints are set so that the time on/off is within range of 0-100%. The code snippet below shows the corresponding function.

```
uint32_t constraints (uint32_t ui32Adjust)
{
    if (ui32Adjust < 10)
        ui32Adjust = 10;
    if (ui32Adjust > 1000)
        ui32Adjust = 1000;
    return ui32Adjust;
}
```

3) ADC for Edge Detection and Reference Speed

Two ADCs are used in the MCU, in which ADC0 will read PWM analog signal and output digital square wave signal and ADC1 outputs digital signal when input voltage to potentiometer is changed from 0-3.3V. The conversion results from first ADC vary from 200 – 4000, and the average will be taken to be around 2000. The results between two consecutive readings will determine if there is a rising/falling edge, and the counter will start timing for 1 period. Speed of the fan in RPM is calculated as:

```
speed = 60/(period/ui32Period);

ui32Period = SysCtlClockGet(); //period for fan = period for crystal clock
```

The second ADC will take reading from potentiometer, which is powered by 3.3V pin from MCU. The result in 12-bit will reflect the current voltage being used. Equation for such relationship is shown in the figure below [3].

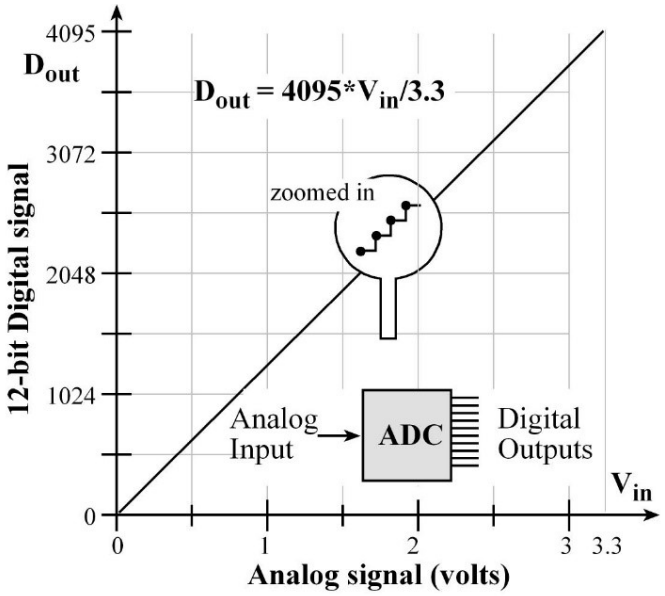


Figure 2: 12-bit Digital Signal vs MCU Analog Signal

4) BCD Output

To create a physical user interface which outputs the RPM of the fan on the circuit the problem is separated into two parts. Two functions were used to implement this. A function is created to convert a decimal number to binary and another is used to output each digit in the binary representation.

4.1 Decimal to Binary

Essentially this function takes a single decimal digit as an input and finds the digit of the equivalent representation in an inputted base at a specific position. For example if the base was input as 2 and the desired position is 2 and the input digit is 9 this function will output 0. If the desired position is 3 then this function will output 1.

The function achieves this through this equation

$$\frac{binary}{denom^{pos}} = digit$$

Digit is the value of the digit of the number input at position “pos” in its base. Taking the modulus of this digit with a specified base will return the digit in that base at that position.

4.2 Binary to output BCD

Using the previous function, the binary digit at any position of an input number could be determined. Using a simple selection algorithm, this function determines whether to write 0 or 1 to the BCD pin.

Pseudocode:

```
if (binary digit at A is 1)
    Output 1 at pin A;
else
    Output 0;
```

Circuit Design

Initially, N-MOSFET circuit is designed to increase the current from PWM signal output in MCU through gate terminal with a 1-kΩ resistor in between N-MOSFET, and then PWM signal is passed into Servo DC motor from drain terminal. The N-MOSFET is power by 5 VDC from MCU. The closed loop feedback signals are then connected back to MCU by 5 VDC again through a resister. Then, three BCD to 7-segment Display Decoders are connected for 3-digit display, where twenty-one 330Ω resisters are used between BCD and 7-segment Display to prevent burning the 7-segment. MCU also supplies the power to all the ICs and 7-segment Displays and grounded to GND pin on MCU. A DIP switch is also used to physically test all the BCD to 7-segment Display Decoders. In this case, it is easier to find errors rather than implement codes and debug from programming. For each BCD, there are four inputs that represent a binary number. In total, there are twelve inputs required from MCU, and thus, twelve pins are used according to written code in CCS. Finally, a potentiometer is utilized for referencing the speed of the Servo DC motor. It is connected to 3.3V from the MCU and grounded at GND pin with an ADC connection between potentiometer and MCU.

Conclusion

In conclusion, the ARM MCU is enabled to proportionally control the speed of the Servo DC motor by PWM signal and display the speed value with 3-digit 7-segments. This project is gradually developed by focusing on a specific objective at a time. For each of the objectives, a lot of tests, debugging, and analysis have been performed to make the system work as required. The project itself provides a worthy experience on how an embedded system work in real life. Throughout the project, C programming skills are acquired in Code Composer Studio. In addition, analysis on control and feedback system is one of the most valuable skill for future studies.

Reference

[1] Narimani, M. (2018). *Design of a Servo DC motor speed controller using ARM MCU* [PDF file].

[2] *The LM324 Quad Op-Amp Line Follower Robot with Pulse Width Modulation*. (2011). [Image]. Retrieved from http://www.ermicro.com/blog/?p=1908

[3] Valvano, J., & Yerraballi, R. (2014). *Chapter 14: Analog to Digital Conversion, Data Acquisition and Control* [Image]. Retrieved from http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C14_ADCdataAcquisition.htm