Design Assignment 6

Student Name: Samuel McCormick
Student #: 1014303276
Student Email: samuel.mccormick@gmail.com
Primary Github address: https://github.com/brokenboredom/tech-muffin.git
Directory: DesignAssignments/DA6

Write, implement, and demonstrate using Atmel Studio 7 a C code for the AVR ATMEGA328p microcontroller that performs the following functions:

You'll use the ADC, and PWM/CCP Module of the ATmega328/p to set and determine the speed of the DC Motor.

1. *Using the Potentiometer connected to ADC0, translate the ADC value (0~1023) to PWM value/speed of the motor (0~255 if using Timer0/2). Verify the operation.*

Using Timer2 in Fast PWM mode and ADC setup from previous assignments we get the average value over 4 samples, average it, and then divide by 4 again to convert to 0-255 for Timer2 use.

```
//Configure TIMER2
        TCCR2A |=       (1 << COM2A1) |           // Fast PWM mode non-inverting (OC0B )
                        (1 << WGM21) | (1 << WGM20);            // Fast PWM, TOP: 0xFF
        TCCR2B |=       (1 << CS22) | (1 << CS21) | (1 << CS20);   // 1024 prescaler

ISR(ADC_vect)
{
        adc_temp += ADC; //sum
        j--;
        if (j==1)
        {
                adc_temp /= 16; // 0-1023 / 16 = 0-255 / 4 average
                //snprintf(outs,sizeof(outs),"%3d\r\n", adc_temp);
                // This adc value from POT on protoshield should set our DC PWM duty cycle
                // So this reading needs to be translated to OCR1A
                OCR0A = adc_temp;       // Overflow will just start from 0%
                adc_temp = 0;
                j = 4;
                _delay_ms(125);
        }
}
```

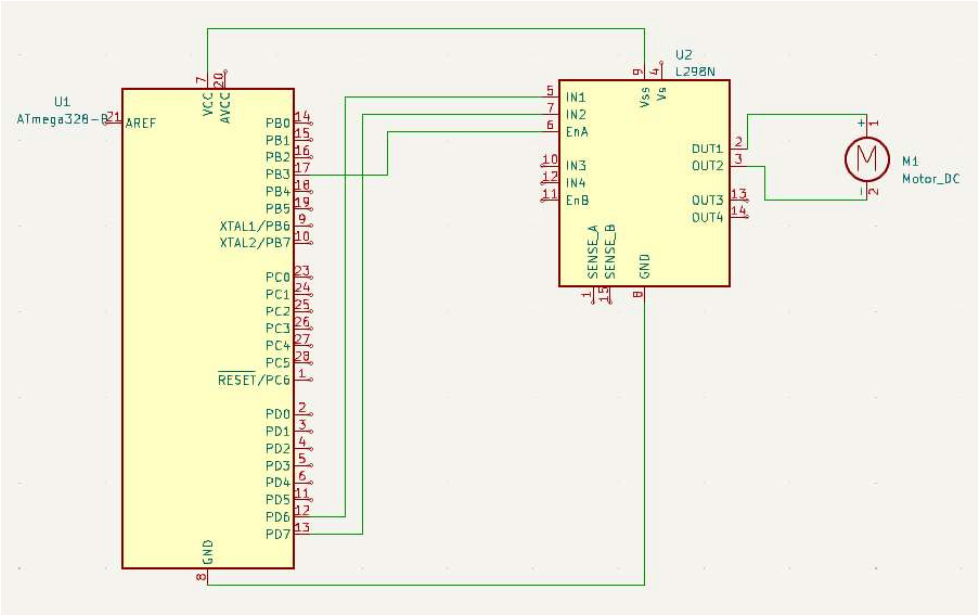
2. *Using the CCP capture pin of PWM1, in mode 1x and/or 2x determine the speed of the DC Motor for a set ADC Pot value/position.*

3. *Display the speed of the motor on the 7-SEG display on the using auto/hardware SPI mode.*

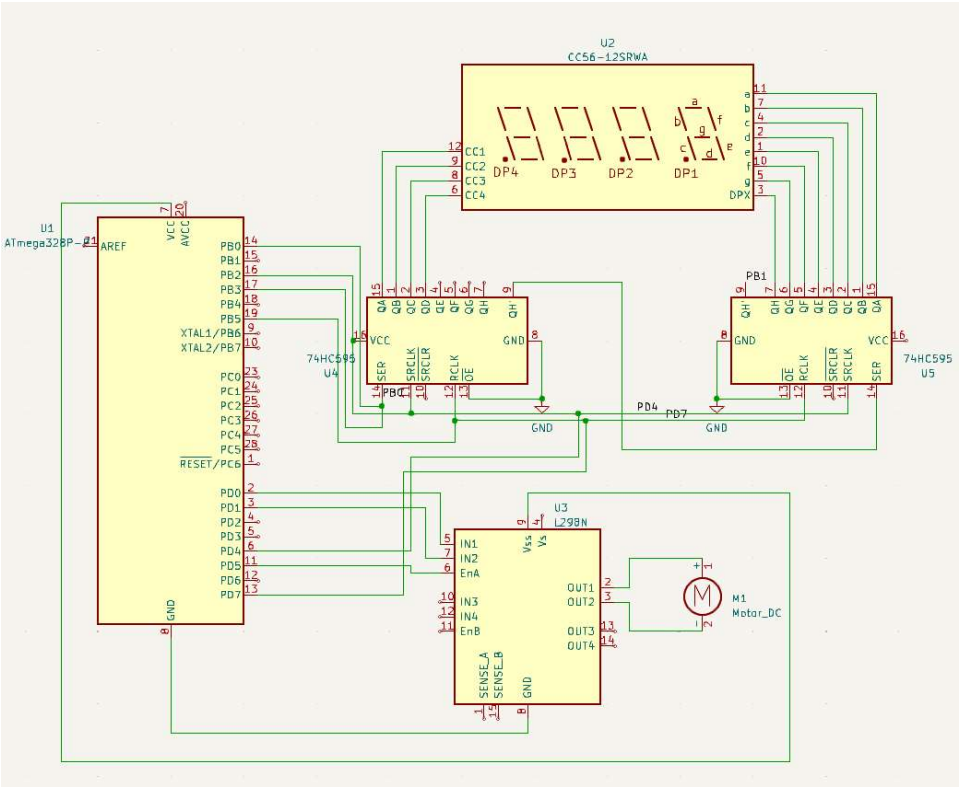
Using the provided setup for SPI latch operation we take our calculated ADC value and extract the value of each digit sending them to the 7-segment display. The display uses ports we had used previously for the motor so we switch to using Timer0 and PD0/PD1 and PD5(OC0B).

```
//Configure TIMER0
        TCCR0A |=       (1 << COM0B1) |   // Fast PWM mode non-inverting (OC0B )
                        (1 << WGM01) | (1 << WGM00);            // Fast PWM, TOP: 0xFF
        TCCR0B |=       (1 << CS02) | (1 << CS01) | (1 << CS00);    // 1024 prescaler
//Configure motor
       DDRD |= 0x03;                // PORTD 0 and 1 as Output
        DDRD |= (1<<PD5);           // PD5(OC0B) motor enable
        PORTD = 0x01;               // set motor to run


ISR(ADC_vect)
{
        adc_temp += ADC; //sum
        j--;
        if (j==1)
        {
                adc_temp /= 16; // 0-1023 / 16 = 0-255 / 4 average
                // This adc value from POT on protoshield should set our DC PWM duty cycle
                // So this reading needs to be translated to OCR1A
                OCR0B = adc_temp;       // Overflow will just start from 0%
                adc_temp = 0;
                j = 4;

                //Pull LATCH low (Important: this is necessary to start the SPI transfer!)
                SHIFT_PORT &= ~LATCH;

               // Extract our ADC digits
                adc_hundreds = adc_temp % 100;
                adc_temp = adc_temp - (adc_hundreds * 100)
                adc_tens = adc_temp % 10;
                adc_temp = adc_temp - (adc_tens * 10)
                adc_ones = adc_temp
                spi_send((unsigned char)SEGMENT_MAP[adc_hundreds]);
                spi_send((unsigned char)0xF2);
                spi_send((unsigned char)SEGMENT_MAP[adc_tens]);
                spi_send((unsigned char)0xF4);
                spi_send((unsigned char)SEGMENT_MAP[adc_ones]);
                spi_send((unsigned char)0xF8);
                //Toggle latch to copy data to the storage register
                SHIFT_PORT |= LATCH;
                SHIFT_PORT &= ~LATCH;
                //wait for a little bit before repeating everything
                _delay_ms(125);
        }
}
```

Schematics:

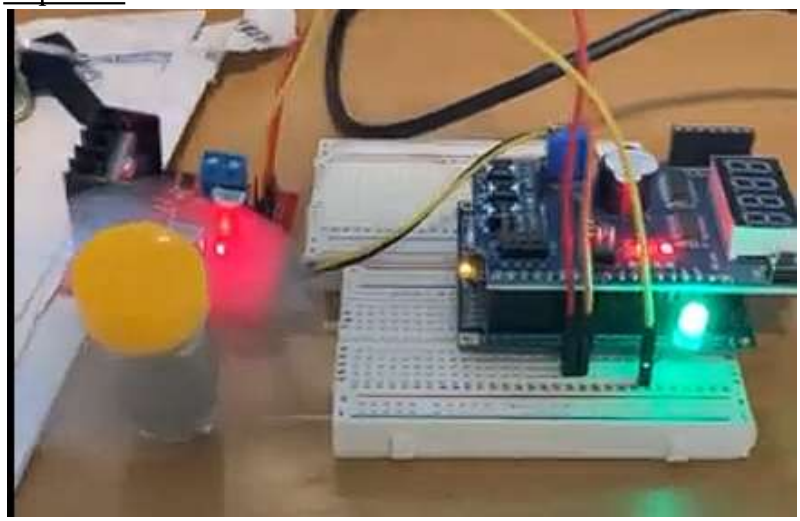


Schematic for part 1. Motor driver and motor connected to atmega328p.

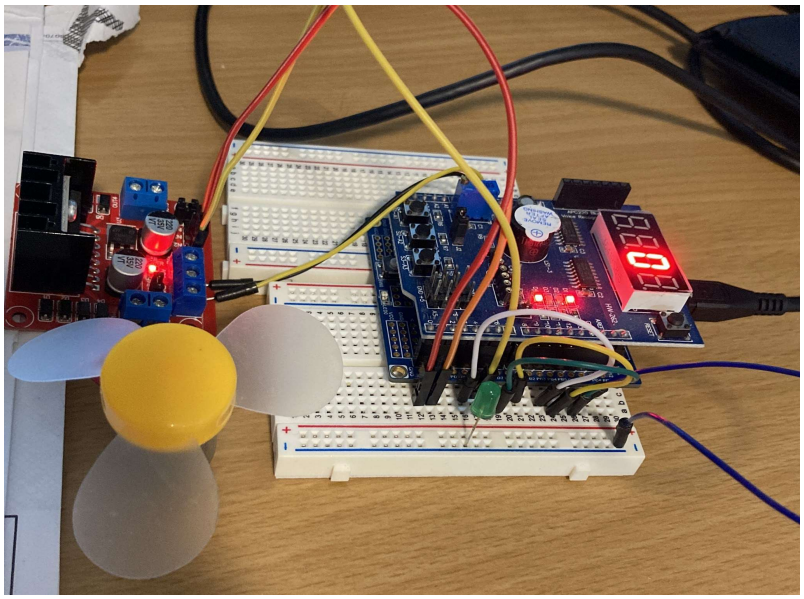


Schematic for part 3. 7-seg display connected to two shift registers through shield to atmega328p. SPI overlap connections (PB2 - PD4. PB3 - PB0. PB5 - PD7.) Adjusted connections for motor driver.

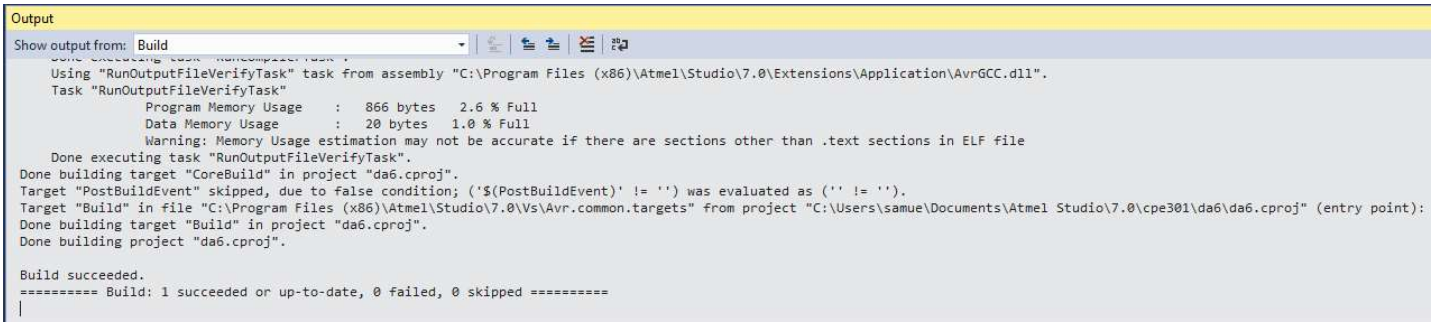
Captures:



Part 1 demo circuit.



Part 3 demo circuit.



Successful compile

Video Links:
Part 1:
https://youtu.be/4DG_QWMf9vo