CPE301 – SPRING 2019
Design Assignment 4A

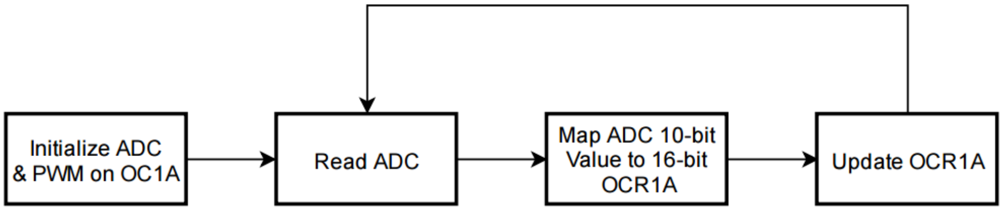
Student Name: Yannick Kengne Tatcha
Student #: 5003294512
Student Email: kengneta@unlv.nevada.edu
Primary Github address: **Vasty1995/submission_da**
Directory: DA4A

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

- Atmega328P
- DC motor
- ULN2003
- PC
- 5V power supply

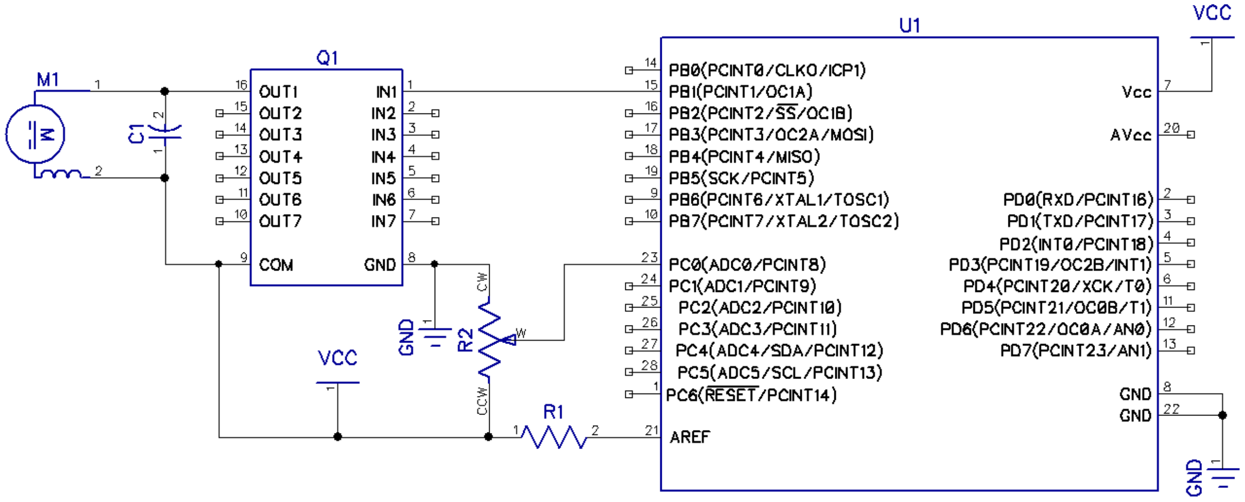
2. DEVELOPED CODE OF TASK 1 in AVR C

Flow chart


Initialize ADC & PWM on OC1A
Read ADC
Map ADC 10-bit Value to 16-bit OCR1A
Update OCR1A

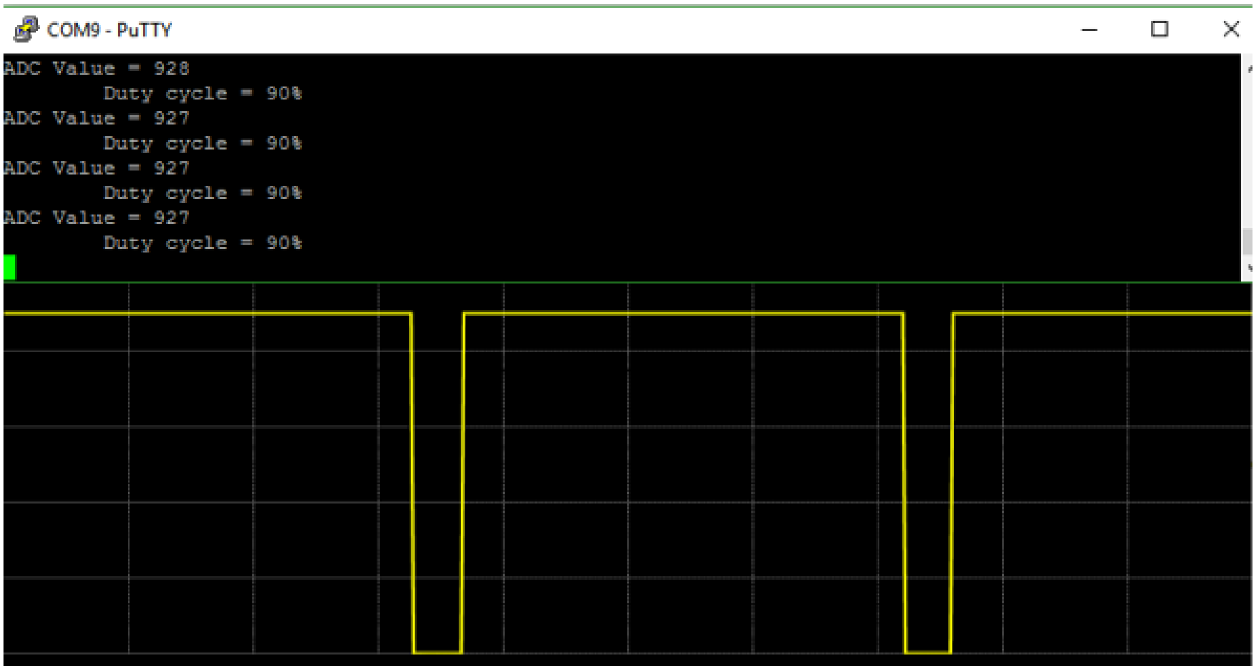
```
/*
 * DA4A.c
 *
 * Created: 4/12/2019 12:13:36 PM
 * Author : YKengne
 */

#include <avr/io.h>


#define F_CPU 8000000UL // XTAL = 8MHZ

#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>

#define BAUDRATE            9600        // Define baudrate
#define ASYNCH_NORM_PRESCALER (F_CPU/16/BAUDRATE - 1) // Calculate prescaler for USART0

void ADC0init();                                     // Initialize ADC0 input
void PWM_OC1A_init();                           // Initialize PWM on OC1A at 50Hz
unsigned short readADC();               // read ADC0 analog input and return it
void updateDC_OC1A(unsigned char); // Change duty cycle on OC1A
int USART0_sendChar(char, FILE*);   // Send character on USART0
void usart0_init (void);                // Initialize USART0

// reset stream pointer
// http://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group__avr__stdio.html
FILE USART0_stream = FDEV_SETUP_STREAM(USART0_sendChar, NULL, _FDEV_SETUP_WRITE);

int main()
{
        unsigned short adcVal;          // Variable to store input ADC Value
        unsigned char  dc;              // Store calculated DC value based on adcVal

        stdout = &USART0_stream;  // change standard output to point to a USART stream

        PWM_OC1A_init();                // initialize pwm  on OC1A
        ADC0init();                             // Initialize ADC0 input
        usart0_init();                  // Initialize USART0 for debugging and monitoring

        while (1)
        {
                adcVal = readADC();             // read ADC0;
                dc = (unsigned short)(100.0*adcVal / 1023); // get percentage of input voltage from Vcc.
                updateDC_OC1A(dc);              // Update OCR1A to update duty cycle of OC1A
                printf("ADC Value = %u\n", adcVal);                     // Monitoring output
                printf("\tDuty cycle = %u%%\n", dc);   // Monitoring output
                _delay_ms(100);                 // Have an imperceivable delay
        }
}

void usart0_init (void)
/*
 * Procedure to initialize USART0 asynchronous with enabled RX/TX, 8 bit data,
 * no parity, and 1 stop bit.
*/
{
        UCSR0B = (1<<TXEN0)  | (1<<RXEN0);      // enable transmit/receive
        UCSR0C = (1<<UCSZ01) | (1<<UCSZ00);     // asynchronous, 8N1
        UBRR0L = ASYNCH_NORM_PRESCALER;                 // Set prescaler based on desired baudrate
}

int USART0_sendChar(char data, FILE *stream)
/*
 * Procedure to send a single character over USART0. If character is linefeed, reset
 * line.
 * Assumes ASCII code.
 */
{
        if(data == '\n')        // If character is linefeed,
        {                                       // First send return.
                while(! (UCSR0A & (1<<UDRE0)) );
                UDR0 = '\r';
        }
        while(! (UCSR0A & (1<<UDRE0)) ); // Wait for last data to be transmitted.
        UDR0 = data;            // send data.
        return 0;
}

void updateDC_OC1A(unsigned char DC)
// Procedure to update PWM duty cycle on OC1A. Given an unsigned character DC, this
// procedure will calculate the appropriate OCR1A value based on the top value of
// Timer1.
{
        OCR1A = (unsigned short)(DC * 2499.0 / 100);
}


unsigned short readADC()
// readADC will read the adcValue after it has been calculated.
{
        ADCSRA |= (1<<ADSC);                            // Begin conversion
        while((ADCSRA & (1<<ADIF)) == 0 );          // Wait for conversion to finish.
        return ADC;
}

void PWM_OC1A_init()
{
        //Set PORTB1 pin as output
        DDRB |= (1<<DDB1);        // make OC1A as output.
        // Output compare mode on OC1A. Fast PWM with top = ICR1.
        // Clear OC1A on Compare match and set at bottom.
        TCCR1A |=
(1<<COM1A1)|(0<<COM1A0)|(0<<COM1B1)|(0<<COM1B0)|(0<<FOC1A)|(0<<FOC1B)|(1<<WGM11)|(0<<WGM10);
        // Start timer with prescaler 64
        TCCR1B |= (0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(1<<WGM12)|(0<<CS12)|(1<<CS11)|(1<<CS10);
        ICR1 = 2499;  // F_CPU / (N * F_pwm) - 1, where N is the prescaler = 64, and F_pwm is the desired 50Hz frequency.
}

void ADC0init()
// ADC0init will initialize analog input on ADC0, set voltage reference to Vcc, with
// data right justified on data register.
{
        DDRC    &= ~(0<<DDC0);
        ADCSRA          = 0x87;                         // Make ADC enable and select ck/128
        ADMUX = (1<<REFS0);       // VCC reference, ADC0 single ended input
                                                        // data will be right-justified
}
```

3. SCHEMATICS



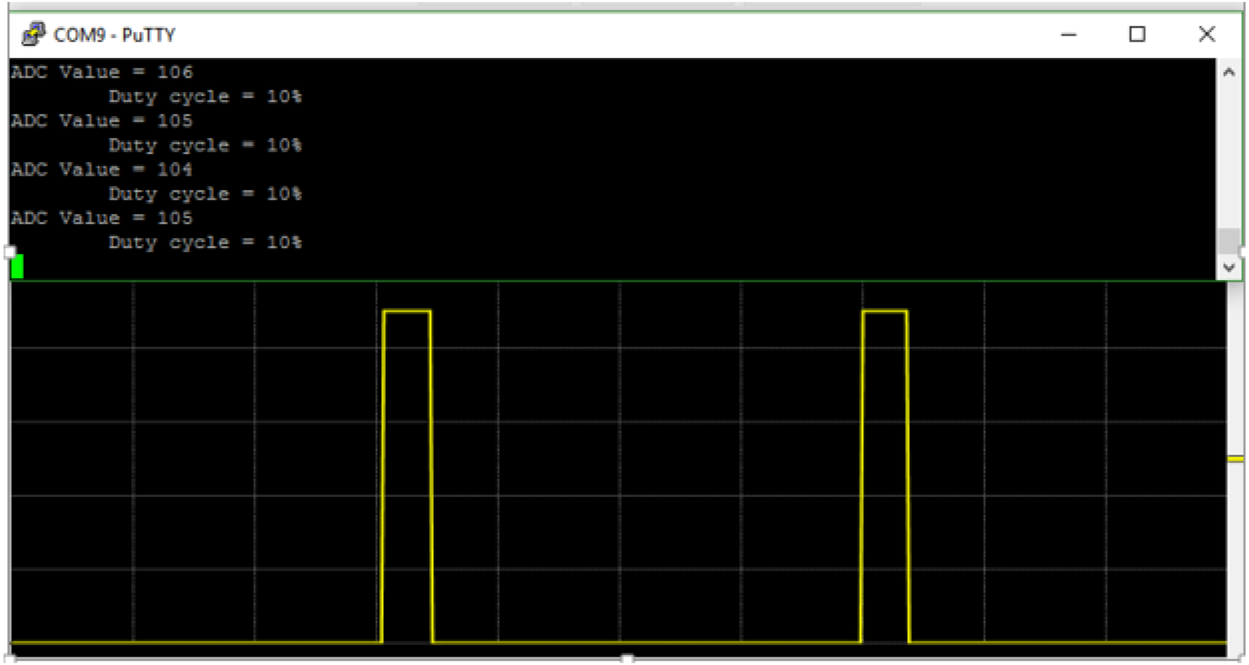
4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

The figure below illustrates where the potentiometer was at a 90% position. Output duty cycle is displayed with the Pololu monitoring software for Pololu USB AVR Programmer



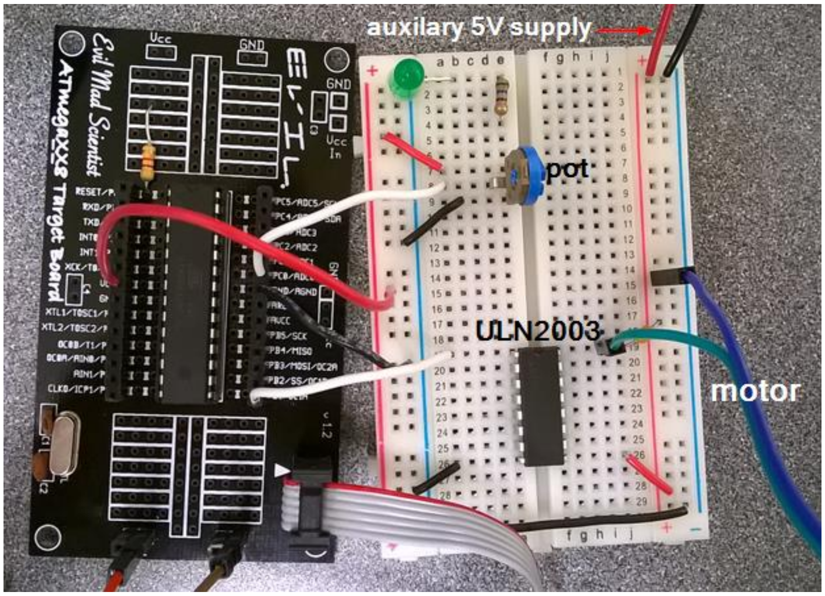
```
ADC Value = 928
        Duty cycle = 90%
ADC Value = 927
        Duty cycle = 90%
ADC Value = 927
        Duty cycle = 90%
ADC Value = 927
        Duty cycle = 90%
```

And then with 10% duty cycle



```
ADC Value = 106
        Duty cycle = 10%
ADC Value = 105
        Duty cycle = 10%
ADC Value = 104
        Duty cycle = 10%
ADC Value = 105
        Duty cycle = 10%
```

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. VIDEO LINKS OF EACH DEMO

7. GITHUB LINK OF THIS DA

https://github.com/Vasty1995/submission_da/tree/master/DA4A

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

“*This assignment submission is my own, original work*”. Yannick Kengne Tatcha