CPE301 – SPRING 2025

Design Assignment 5

Student Name: Ryan Sewell
Student #: 8000473785
Student Email: sewelr2@unlv.nevada.edu
Primary Github address: https://github.com/sewelr2
Directory: DA-Submissions/DA5
Video Playlist:
https://youtube.com/playlist?list=PLt45mEFhRV6ffOYRcGHhoI5aDeP3Zgqt5&feature=shared

The core objective of this lab is to build a single AVR-based motor-control demo in Atmel Studio 7 that:

1. Reads a potentiometer on ADC0 (0–1023) and maps it to an 8-bit PWM duty (0–255) on OC0A to drive your DC motor.
2. Measures the actual motor speed using the Timer/CCP input-capture (ICP1) hardware.
3. Overrides the pot-based setpoint via a simple UART "GUI," streaming out CSV pairs of (set-value, measured-speed) so you can live-plot both traces in a PC tool.

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Microchip Studio	Atmega328PB-Xmini PC	Multi-Function Shield	Tauno Serial Plotter
- Assembler	- Polulu md08a	- Potentiometer	
- Simulator	- DC Motor		
- Debugger			

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A

```
#define F_CPU 16000000UL /* Define CPU Frequency e.g. here its 8MHz */
#include "uart.h"
#include <avr/interrupt.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <stdio.h>
#include <util/delay.h>

// capture Flag
volatile uint8_t Flag;
volatile uint8_t Direction = 0;
volatile uint32_t revTickAvg;

void ADC_Init() /* ADC Initialization function */
{
  DDRC = 0x00;   /* Make ADC port as input */
  ADCSRA = 0x87; /* Enable ADC, with freq/128  */
  ADMUX = 0x40;  /* Vref: Avcc, ADC channel: 0 */
```

```
}

int ADC_Read(char channel) /* ADC Read function */
{
  ADMUX = 0x40 | (channel & 0x07); /* set input channel to read */
  ADCSRA |= (1 << ADSC);           /* Start ADC conversion */
  while (!(ADCSRA & (1 << ADIF)))
    ; /* Wait until end of conversion by polling ADC interrupt flag */
  ADCSRA |= (1 << ADIF); /* Clear interrupt flag */
  _delay_us(1);          /* Wait a little bit */
  return ADCW;           /* Return ADC word */
}

// INT0 interrupt
ISR(INT0_vect) {
  // Use for Motor direction one trigger for forward, another for reverse
}

// INT1 interrupt
ISR(INT1_vect) {
  // Use for Motor direction one trigger for stop and go
}

volatile uint32_t revTick; // Ticks per revolution
volatile uint32_t revCtr;  // Total elapsed revolutions
volatile uint16_t T1Ovs2;  // Overflows for small rotations

// Initialize timer
void InitTimer1(void) {
  // Set PB0 as input
  DDRB &= ~(1 << DDB0);
  PORTB |= (1 << DDB0);

  // Set Initial Timer value
  TCNT1 = 0;
  ////First capture on rising edge
  TCCR1A = 0;
  TCCR1B = (0 << ICNC1) | (1 << ICES1);
  TCCR1C = 0;
  // Interrupt setup
  // ICIE1: Input capture
  // TOIE1: Timer1 overflow
  TIFR1 = (1 << ICF1) | (1 << TOV1);    // clear pending
  TIMSK1 = (1 << ICIE1) | (1 << TOIE1); // and enable
}

void StartTimer1(void) {
  // Start timer without pre-scaler
  TCCR1B |= (1 << CS10);
  // Enable global interrupts
  sei();
}
```

```
volatile uint32_t tickv, ticks;
// capture ISR
ISR(TIMER1_CAPT_vect) {
    tickv = ICR1; // save duration of last revolution
    revTickAvg = (uint32_t)(tickv) + ((uint32_t)T1Ovs2 * 0x10000L);
  revCtr++;  // add to revolution count
  TCNT1 = 0; // restart timer for next revolution
  T1Ovs2 = 0;
}
// Overflow ISR
ISR(TIMER1_OVF_vect) {
  // increment overflow counter
  T1Ovs2++;
}

int main(void) {
  char outs[72];
  USART_Init(9600);
  USART_SendString("Connected!\r\n"); // we're alive!
  InitTimer1();
  StartTimer1();
  USART_SendString("TIMER1 ICP Running \r\n");
  /* set PD2 and PD3 as input */
  DDRD &= ~(1 << DDD2);                          /* Make INT0 pin as Input */
  DDRD &= ~(1 << DDD3);                          /* Make INT1 pin as Input */
  PORTD |= (1 << DDD2) | (1 << DDD3);            // turn On the Pull-up
  DDRD |= (1 << DDD6) | (1 << DDD4) | (1 << DDD5); /* Make OC0 pin as Output */
  // We are manually setting the direction
  PORTD |= (1 << DDD5);              // CW Direction Set
  PORTD &= ~(1 << DDD4);             // CW Direction Set
  EIMSK |= (1 << INT0) | (1 << INT1); /* enable INT0 and INT1 */
  MCUCR |= (1 << ISC01) | (1 << ISC11) |
           (1 << ISC10); /* INT0 - falling edge, INT1 - raising edge */
  sei();                 /* Enable Global Interrupt */
  // WE are not using the ADC for speed - just manually setting the value
  ADC_Init(); /* Initialize ADC */
  TCNT0 = 0;  /* Set timer0 count zero */
  TCCR0A |= (1 << WGM00) | (1 << WGM01) | (1 << COM0A1);
  TCCR0B |=
      (1 << CS00) | (1 << CS02); /* Set Fast PWM with Fosc/64 Timer0 clock */
  OCR0A = 30;
  /* ready start value */
  while (1) {
    // Convert ticks to RPM
       // send Speed value to LCD or USART
     USART_SendString("Tick;Period;Frequency  ");
     snprintf(outs, sizeof(outs), "%f ", (float)revTickAvg); // print it
     USART_SendString(outs);
     USART_SendString(" \r\n");
  }
}
```

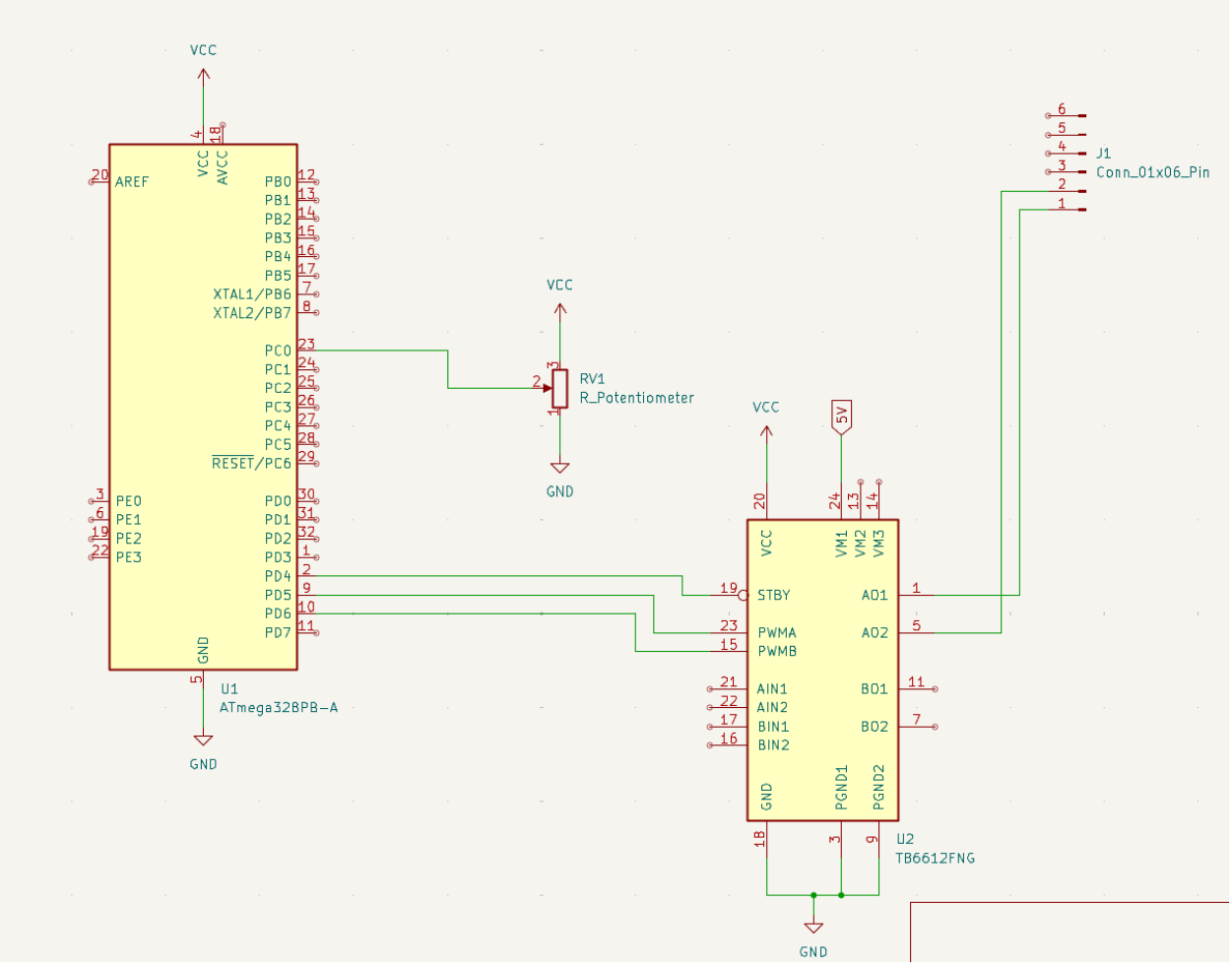
3. DEVELOPED/MODIFIED CODE OF TASK 2/A from TASK 1/A

```
void pwm0_init()
{
	// Fast PWM, non-inverting on OC0A, prescaler at 64
	TCCR0A = (1 << WGM01) | (1 << WGM00) | (1 << COM0A1);
	TCCR0B = (1 << CS01) | (1 << CS00);
	DDRD |= (1 << PIND6); // PD6 (OC0A) as output
}

void icp1_init()
{
	TCCR1A = 0;
	// Noise cancel, rising edge, prescaler = 8
	TCCR1B = (1 << ICES1) | (1 << ICNC1) | (1 << CS11);
	TIMSK1 = (1 << ICIE1); // ICP interrupt
	sei(); // Enable global interrupts
}

ISR(TIMER1_CAPT_vect)
{
	uint16_t now = ICR1;
	per = now - lastCapt;
	lastCapt = now;
	newPer = 1;
}

void usart_init(uint16_t ubrr)
{
	UBRR0H = ubrr >> 8;
	UBRR0L = ubrr;
	UCSR0B = (1 << RXEN0) | (1 << TXEN0) | (1 << RXCIE0);
	UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

ISR(USART0_RX_vect)
{
	char buffer[4];
	uint8_t idx = 0;
	char c = UDR0;
	if (c >= '0' && c <= '9' && idx < 3 )
	{
		buffer[idx++] = c;
	}
	else if ((c == '\r' || c == '\n') && idx > 0)
	{
		buffer[idx] = 0;
		setPoint = (uint16_t)atoi(buffer);
		override = 1;
		idx = 0;
	}
}
```

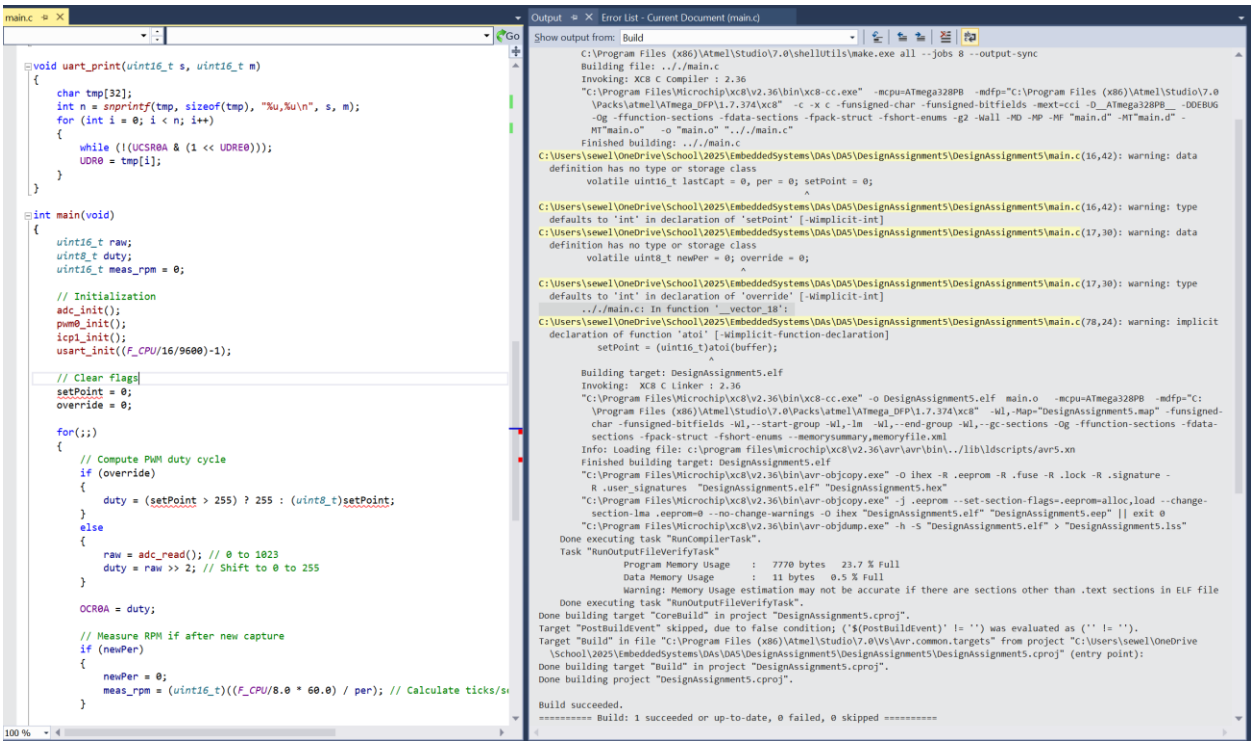
```
void uart_print(uint16_t s, uint16_t m)
{
	char tmp[32];
	int n = snprintf(tmp, sizeof(tmp), "%u,%u\n", s, m);
	for (int i = 0; i < n; i++)
	{
		while (!(UCSR0A & (1 << UDRE0)));
		UDR0 = tmp[i];
	}
}

int main(void)
{
	uint16_t raw;
	uint8_t duty;
	uint16_t meas_rpm = 0;

	// Initialization
	adc_init();
	pwm0_init();
	icp1_init();
	usart_init((F_CPU/16/9600)-1);

	// Clear flags
	setPoint = 0;
	override = 0;

	for(;;)
	{
		// Compute PWM duty cycle
		if (override)
		{
			duty = (setPoint > 255) ? 255 : (uint8_t)setPoint;
		}
		else
		{
			raw    = adc_read(); // 0 to 1023
			duty = raw >> 2; // Shift to 0 to 255
		}

		OCR0A = duty;

		// Measure RPM if after new capture
		if (newPer)
		{
			newPer = 0;
			meas_rpm = (uint16_t)((F_CPU/8.0 * 60.0) / per); // Calculate ticks/sec
		}

		// Send measurement for plotting
		uart_print(duty, meas_rpm);
```

```
            _delay_ms(50); // Print every 50ms
        }
}
```

4. SCHEMATICS



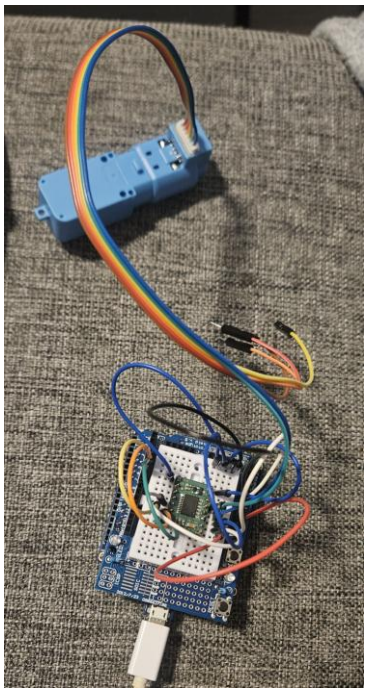
5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



```
void uart_print(uint16_t s, uint16_t m)
{
    char tmp[32];
    int n = snprintf(tmp, sizeof(tmp), "%u,%u\n", s, m);
    for (int i = 0; i < n; i++)
    {
        while (!(UCSR0A & (1 << UDRE0)));
        UDR0 = tmp[i];
    }
}

int main(void)
{
    uint16_t raw;
    uint8_t duty;
    uint16_t meas_rpm = 0;

    // Initialization
    adc_init();
    pwm0_init();
    icp1_init();
    usart_init((F_CPU/16/9600)-1);

    // Clear flags
    setPoint = 0;
    override = 0;

    for(;;)
    {
        // Compute PWM duty cycle
        if (override)
        {
            duty = (setPoint > 255) ? 255 : (uint8_t)setPoint;
        }
        else
        {
            raw = adc_read(); // 0 to 1023
            duty = raw >> 2; // Shift to 0 to 255
        }

        OCR0A = duty;

        // Measure RPM if after new capture
        if (newPer)
        {
            newPer = 0;
            meas_rpm = (uint16_t)((F_CPU/8.0 * 60.0) / per); // Calculate ticks/s
        }
```

```
C:\Program Files (x86)\Atmel\Studio\7.0\shellUtils\make.exe all --jobs 8 --output-sync
Building file: ../../main.c
Invoking: XC8 C Compiler : 2.36
"C:\Program Files\Microchip\xc8\v2.36\bin\xc8-cc.exe"  -mcpu=ATmega328PB  -mdfp="C:\Program Files (x86)\Atmel\Studio\7.0
  \Packs\atmel\ATmega_DFP\1.7.374\xc8"  -c -x c -funsigned-char -funsigned-bitfields -mext=cci -D__ATmega328PB__ -DDEBUG
  -Og -ffunction-sections -fdata-sections -fpack-struct -fshort-enums -g2 -Wall -MD -MP -MF "main.d" -MT"main.d" -
  MT"main.o"  -o "main.o" "../../main.c"
Finished building: ../../main.c
C:\Users\sewel\OneDrive\School\2025\EmbeddedSystems\DAs\DA5\DesignAssignment5\DesignAssignment5\main.c(16,42): warning: data
  definition has no type or storage class
       volatile uint16_t lastCapt = 0, per = 0; setPoint = 0;
                                                ^
C:\Users\sewel\OneDrive\School\2025\EmbeddedSystems\DAs\DA5\DesignAssignment5\DesignAssignment5\main.c(16,42): warning: type
  defaults to 'int' in declaration of 'setPoint' [-Wimplicit-int]
C:\Users\sewel\OneDrive\School\2025\EmbeddedSystems\DAs\DA5\DesignAssignment5\DesignAssignment5\main.c(17,30): warning: data
  definition has no type or storage class
       volatile uint8_t newPer = 0; override = 0;
                            ^
C:\Users\sewel\OneDrive\School\2025\EmbeddedSystems\DAs\DA5\DesignAssignment5\DesignAssignment5\main.c(17,30): warning: type
  defaults to 'int' in declaration of 'override' [-Wimplicit-int]
    ../../main.c: In function '__vector_18':
C:\Users\sewel\OneDrive\School\2025\EmbeddedSystems\DAs\DA5\DesignAssignment5\DesignAssignment5\main.c(78,24): warning: implicit
  declaration of function 'atoi' [-Wimplicit-function-declaration]
         setPoint = (uint16_t)atoi(buffer);
                              ^
Building target: DesignAssignment5.elf
Invoking:  XC8 C Linker : 2.36
"C:\Program Files\Microchip\xc8\v2.36\bin\xc8-cc.exe" -o DesignAssignment5.elf  main.o   -mcpu=ATmega328PB  -mdfp="C:
  \Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.7.374\xc8"  -Wl,-Map="DesignAssignment5.map" -funsigned-
  char -funsigned-bitfields -Wl,--start-group -Wl,-lm  -Wl,--end-group -Wl,--gc-sections -Og -ffunction-sections -fdata-
  sections -fpack-struct -fshort-enums --memorysummary,memoryfile.xml
Info: Loading file: c:\program files\microchip\xc8\v2.36\avr\avr\bin\../lib\ldscripts/avr5.xn
Finished building target: DesignAssignment5.elf
"C:\Program Files\Microchip\xc8\v2.36\bin\avr-objcopy.exe" -O ihex -R .eeprom -R .fuse -R .lock -R .signature -
  R .user_signatures  "DesignAssignment5.elf" "DesignAssignment5.hex"
"C:\Program Files\Microchip\xc8\v2.36\bin\avr-objcopy.exe" -j .eeprom --set-section-flags=.eeprom=alloc,load --change-
  section-lma .eeprom=0 --no-change-warnings -O ihex "DesignAssignment5.elf" "DesignAssignment5.eep" || exit 0
"C:\Program Files\Microchip\xc8\v2.36\bin\avr-objdump.exe" -h -S "DesignAssignment5.elf" > "DesignAssignment5.lss"
Done executing task "RunCompilerTask".
Task "RunOutputFileVerifyTask"
        Program Memory Usage     :   7770 bytes   23.7 % Full
        Data Memory Usage        :   11 bytes   0.5 % Full
        Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "DesignAssignment5.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\sewel\OneDrive
  \School\2025\EmbeddedSystems\DAs\DA5\DesignAssignment5\DesignAssignment5\DesignAssignment5.cproj" (entry point):
Done building target "Build" in project "DesignAssignment5.cproj".
Done building project "DesignAssignment5.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



Kept receiving this error, switched ports, tried on different computer, checked connections, and tried different hat. Problem still persisted. Encoder may be broken

7. VIDEO LINKS OF EACH DEMO

Task 1:
Task 2:
…

8. GITHUB LINK OF THIS DA

https://github.com/sewelr2/DA-Submissions/tree/master/DA5

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

"*This assignment submission is my own, original work*".
Ryan Sewell