CPE301 – SPRING 2025

Design Assignment 6

Student Name: Ryan Sewell

Student #: 8000473785

Student Email: sewelr2@unlv.nevada.edu

Primary Github address: https://github.com/sewelr2

Directory: DA-Submissions/DA6

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.
4. **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

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

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdint.h>

//—— your Wait() from reference (~20 ms) ———————————————————————

void Wait(void)

{

*uint8\_t* i;

for (i = 0; i < 50; i++)

{

*\_delay\_loop\_2*(0);

*\_delay\_loop\_2*(0);

*\_delay\_loop\_2*(0);

}

}

//—— hardware definitions ————————————————————————————————————

// Servo PWM on PB1/OC1A (Timer1)

#define SERVO\_DDR DDRB

#define SERVO\_PIN PINB1

// HC‑SR04 Trigger on PC1

#define TRIG\_DDR DDRC

#define TRIG\_PORT PORTC

#define TRIG\_PIN PINC1

// HC‑SR04 Echo on PD6

#define ECHO\_PINR PIND

#define ECHO\_PIN PIND6

//—— USART @9600, TX only ———————————————————————————————————

#define BAUD 9600

#define UBRR\_VAL ((*F\_CPU*/16/BAUD) - 1)

static void USART\_init(void)

{

UBRR0H = (*uint8\_t*)(UBRR\_VAL >> 8);

UBRR0L = (*uint8\_t*)UBRR\_VAL;

UCSR0B = (1<<TXEN0); // TX enable

UCSR0C = (1<<UCSZ01)|(1<<UCSZ00); // 8N1

}

static void USART\_send(char c)

{

while (!(UCSR0A & (1<<UDRE0)));

UDR0 = c;

}

static void USART\_print\_u16(*uint16\_t* x)

{

char buf[6];

*uint8\_t* i = 0;

if (x == 0)

{

USART\_send('0');

return;

}

while (x && i < sizeof(buf))

{

buf[i++] = '0' + (x % 10);

x /= 10;

}

while (i--)

USART\_send(buf[i]);

}

//—— Servo (Timer1 Fast PWM Mode 14, prescaler=64 → 50 Hz) ———————

static void servo\_init(void)

{

SERVO\_DDR |= (1<<SERVO\_PIN); // PB1 output

// COM1A1=1 non‑inverting OC1A, WGM11=1

TCCR1A = (1<<COM1A1)|(1<<WGM11);

// WGM13=1, WGM12=1, CS11=1, CS10=1 → prescaler=64, Mode 14

TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);

ICR1 = 4999; // TOP = 4999 → 50 Hz

}

// map 0–180° → 250–500 ticks (1 ms–2 ms @ 4 µs/tick)

static inline void servo\_setAngle(*uint8\_t* angle)

{

OCR1A = 250 + ((*uint32\_t*)angle \* 250) / 180;

}

//—— HC‑SR04 setup & measurement by polling TCNT1 —————————————

static void ultrasonic\_init(void)

{

TRIG\_DDR |= (1<<TRIG\_PIN); // trigger pin output

DDRD &= ~(1<<ECHO\_PIN); // echo pin input

PORTD &= ~(1<<ECHO\_PIN); // no pull‑up

}

static *uint16\_t* ultrasonic\_read\_raw(void)

{

*uint16\_t* start, end;

// 10 µs trigger pulse

TRIG\_PORT |= (1<<TRIG\_PIN);

*\_delay\_us*(10);

TRIG\_PORT &= ~(1<<TRIG\_PIN);

// wait for echo high

while (!(ECHO\_PINR & (1<<ECHO\_PIN)));

start = TCNT1;

// wait for echo low

while (ECHO\_PINR & (1<<ECHO\_PIN));

end = TCNT1;

return end - start; // raw ticks (4 µs each)

}

//—— Main sweep —————————————————————————————————————————

int main(void)

{

USART\_init();

servo\_init();

ultrasonic\_init();

while (1)

{

// CW sweep: 0→180 in 2° steps

for (*uint8\_t* ang = 0; ang <= 180; ang += 2)

{

servo\_setAngle(ang);

Wait();

*uint16\_t* raw = ultrasonic\_read\_raw();

USART\_print\_u16(ang);

USART\_send(',');

USART\_print\_u16(raw);

USART\_send('\n');

}

// CCW sweep: 180→0

for (*int8\_t* ang = 180; ang >= 0; ang -= 2)

{

servo\_setAngle(ang);

Wait();

*uint16\_t* raw = ultrasonic\_read\_raw();

USART\_print\_u16(ang);

USART\_send(',');

USART\_print\_u16(raw);

USART\_send('\n');

}

}

}

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

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdint.h>

//—— your Wait() from reference (~20 ms) ———————————————————————

void Wait(void)

{

uint8\_t i;

for (i = 0; i < 50; i++)

{

\_delay\_loop\_2(0);

\_delay\_loop\_2(0);

\_delay\_loop\_2(0);

}

}

//—— hardware definitions ————————————————————————————————————

// Servo PWM on PB1/OC1A (Timer1)

#define SERVO\_DDR DDRB

#define SERVO\_PIN PINB1

// HC‑SR04 Trigger on PC1

#define TRIG\_DDR DDRC

#define TRIG\_PORT PORTC

#define TRIG\_PIN PINC1

// HC‑SR04 Echo on PD6

#define ECHO\_PINR PIND

#define ECHO\_PIN PIND6

// USART0 TX only @9600 baud

#define BAUD 9600

#define UBRR\_VAL ((F\_CPU/16/BAUD) - 1)

// SPI pins for MAX7219 (7‑seg driver)

#define SPI\_DDR DDRB

#define SPI\_PORT PORTB

#define SPI\_MOSI PINB3

#define SPI\_SCK PINB5

#define SPI\_SS PINB2

//—— SPI0 + MAX7219 routines ———————————————————————————————

void SPI\_init(void)

{

SPI\_DDR |= (1<<SPI\_MOSI)|(1<<SPI\_SCK)|(1<<SPI\_SS);

SPI\_PORT |= (1<<SPI\_SS);

SPCR0 = (1<<SPE0)|(1<<MSTR0)|(1<<SPR00); // SPI0 enabled, Master, Fosc/16

}

void max7219\_send(uint8\_t reg, uint8\_t data)

{

SPI\_PORT &= ~(1<<SPI\_SS);

SPDR0 = reg;

while (!(SPSR0 & (1<<SPIF0)));

SPDR0 = data;

while (!(SPSR0 & (1<<SPIF0)));

SPI\_PORT |= (1<<SPI\_SS);

}

void max7219\_init(void)

{

max7219\_send(0x09, 0x0F); // decode mode: digits 0–3

max7219\_send(0x0A, 0x0F); // intensity

max7219\_send(0x0B, 0x03); // scan limit: 4 digits

max7219\_send(0x0C, 0x01); // normal operation

max7219\_send(0x0F, 0x00); // display test: off

}

static const uint16\_t pow10[4] = {1, 10, 100, 1000};

void displayNumber(uint16\_t num)

{

for (uint8\_t d = 0; d < 4; d++)

{

uint8\_t val = (num / pow10[d]) % 10;

max7219\_send(d + 1, val);

}

}

//—— USART0 TX only ———————————————————————————————————————

void USART\_init(void)

{

UBRR0H = (uint8\_t)(UBRR\_VAL >> 8);

UBRR0L = (uint8\_t)UBRR\_VAL;

UCSR0B = (1<<TXEN0);

UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);

}

void USART\_send(char c)

{

while (!(UCSR0A & (1<<UDRE0)));

UDR0 = c;

}

void USART\_print\_u16(uint16\_t x)

{

char buf[6];

uint8\_t i = 0;

if (x == 0) { USART\_send('0'); return; }

while (x && i < sizeof(buf))

{

buf[i++] = '0' + (x % 10);

x /= 10;

}

while (i--) USART\_send(buf[i]);

}

//—— Servo (Timer1 Fast PWM Mode 14, prescaler=64 → 50 Hz) ———————

void servo\_init(void)

{

SERVO\_DDR |= (1<<SERVO\_PIN);

TCCR1A = (1<<COM1A1)|(1<<WGM11);

TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);

ICR1 = 4999;

}

static inline void servo\_setAngle(uint8\_t angle)

{

OCR1A = 250 + ((uint32\_t)angle \* 250) / 180;

}

//—— HC‑SR04 setup & measurement by polling TCNT1 —————————————

void ultrasonic\_init(void)

{

TRIG\_DDR |= (1<<TRIG\_PIN);

DDRD &= ~(1<<ECHO\_PIN);

PORTD &= ~(1<<ECHO\_PIN);

}

uint16\_t ultrasonic\_read\_raw(void)

{

uint16\_t start, end;

TRIG\_PORT |= (1<<TRIG\_PIN);

\_delay\_us(10);

TRIG\_PORT &= ~(1<<TRIG\_PIN);

while (!(ECHO\_PINR & (1<<ECHO\_PIN)));

start = TCNT1;

while (ECHO\_PINR & (1<<ECHO\_PIN));

end = TCNT1;

return end - start;

}

//—— Main sweep with 7‑SEG display —————————————————————————

int main(void)

{

USART\_init();

SPI\_init();

max7219\_init();

servo\_init();

ultrasonic\_init();

while (1)

{

uint16\_t min\_raw = 0xFFFF;

// CW: display each raw reading

for (uint8\_t ang = 0; ang <= 180; ang += 2)

{

servo\_setAngle(ang);

Wait();

uint16\_t raw = ultrasonic\_read\_raw();

if (raw < min\_raw) min\_raw = raw;

// log on USART

USART\_print\_u16(ang);

USART\_send(',');

USART\_print\_u16(raw);

USART\_send('\n');

// show current reading on 7‑seg

displayNumber(raw);

}

// CCW: display lowest reading from CW scan

for (int8\_t ang = 180; ang >= 0; ang -= 2)

{

servo\_setAngle(ang);

Wait();

displayNumber(min\_raw);

}

}

1. **SCHEMATICS**

A computer screen shot of a circuit board

AI-generated content may be incorrect.

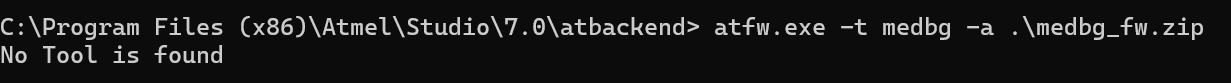
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

A screenshot of a computer

AI-generated content may be incorrect.

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

A close up of a circuit board

AI-generated content may be incorrect. 

**Board was bricked and have been unable to unbrick it. Photos above show what I was seeing.**

1. **VIDEO LINKS OF EACH DEMO**
2. **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