CPE301 – SPRING 2019

Design Assignment 4B

Student Name: Rocky Yasuaki Gonzalez
Student #: 5003229733
Student Email: gonzar14@unlv.nevada.edu
Primary Github address: https://github.com/rockyg1995/ihswppdar.git
Directory: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA4B

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND FLOW DIAGRAMS

Atmega328PB Xplained Mini
Micro USB Cable (Power Supply)
Breadboard
Multifunction Shield (Potentiometer)
ULN2003 w/ LEDs
Stepper Motor
Servo Motor
Female to Female Wires

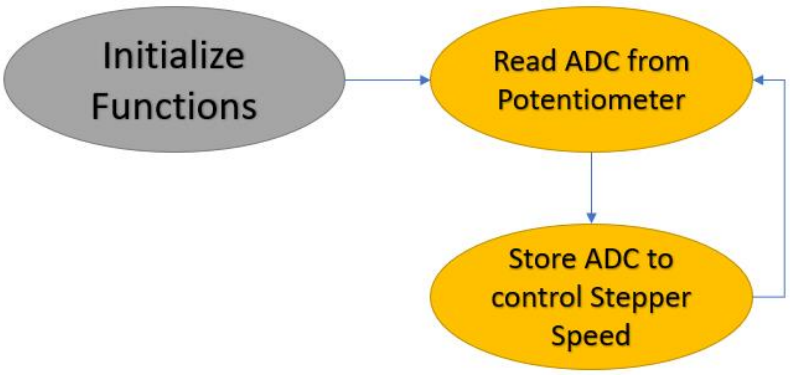

Figure 1 – Flow Chart for Coding Algorithm in Task 1

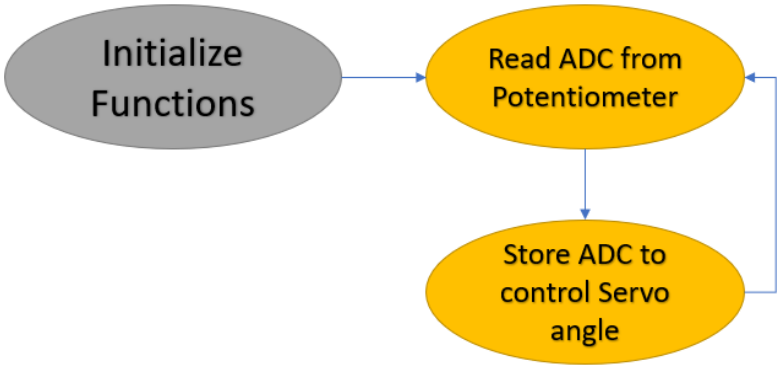

Figure 2 – Flow Chart for Coding Algorithm in Task 2

2. INITIAL/DEVELOPED CODE OF TASK 1

```
/*
 * DA4B_Task_1.c
 *
 * Created: 4/13/2019 3:36:09 PM
 * Author: RYG95
 */
#define F_CPU 16000000UL    // Frequency of Xplained Mini (16MHz)
#include <avr/io.h>         // Standard AVR Library
#include <stdio.h>          // AVR library containing printf functions
#include <avr/interrupt.h>  // AVR library containing interrupt functions
#include <util/delay.h>     // AVR library containing _delay_ms() function
```

```
#define BAUDRATE 9600                                          // Baudrate (bps)
#define BAUD_PRESCALLER ((F_CPU / (BAUDRATE * 16UL)) - 1)      // Baudrate Prescaler

void Timer0_init(void);                    // Initialize Timer1 properties
void adc_init(void);                       // Initialize Analog to Digital Converter
void read_adc(void);                       // Read value from ADC

void USART_init(void);                     // Initialize USART
unsigned char USART_receive(void);         // Receive Serial data from UDR0
void USART_send(unsigned char data);       // Send individual char in UDR0
void USART_putstring(char* StringPtr);     // Break string into chars and send

volatile float adc_val;                    // Store ADC Value representing Analog Voltage
char outs[20];                             // Store float values into array of chars

int main(void) {

	DDRC = 0x00;                           // Inputs: PC7:0 (PC0 Input ADC, PC1 Switch)
	PORTC = (1<<DDC1);                     // Set pull-up for PC1
	DDRB |= (1 << DDB0)|(1 << DDB1)|(1 << DDB2)|(1 << DDB3);      // Outputs: PB3:0

	Timer0_init();                         // Call the Timer0/PWM initialization
	adc_init();                            // Call the ADC initialization code
	USART_init();                          // Call the USART initialization code
	USART_putstring("Connected!\r\n"); // Pass 'Connected!' to function
	_delay_ms(125);                        // Wait a bit

	while (1) {                            // Infinite loop
		read_adc();                        // Read ADC Value
		snprintf(outs,sizeof(outs),"%3f", adc_val);// Store 'adc_val' into 'outs'
		USART_putstring(outs);             // Pass 'outs' to function
		_delay_ms(250);                    // delay

		if (adc_val >= 242) {              // If 'adc_val >= 242'...
			OCR0A = 242;                   // Cap Duty Cycle at 95%
			USART_putstring(", 95%\r\n");              //
		} else if ((adc_val<242) & (adc_val >= 5)) {    // If between range
			OCR0A = adc_val;                           // DC matches adc_val
			USART_putstring(", ");
			adc_val = (adc_val/255)*100;
			snprintf(outs,sizeof(outs),"%3f", adc_val);
			USART_putstring(outs);
			USART_putstring("%\r\n");
		} else if (adc_val < 5) {                      // If 'adc_val < 5'...
			OCR0A = 0;                                 // Drop Duty Cycle to 0%
			USART_putstring(", 0%\r\n");
		}

		PORTB=0x09;
		while (TCNT0 != OCR0A);
		PORTB=0x0C;
		while (TCNT0 != OCR0A);
		PORTB=0x06;
		while (TCNT0 != OCR0A);
		PORTB=0x03;
		while (TCNT0 != OCR0A);
	}
	return 0;
```

```
}
//---------------------------------------------------------------------
void Timer0_init(void) {                        // Initialize Timer0 properties
      TCCR0A |= (1 << WGM01);                   // Set fast PWM Mode
      TCCR0B |= (1 << CS02) | (1 << CS00);      // Set prescaler to 256 and starts PWM
}
//---------------------------------------------------------------------
void USART_init(void) {                         // Initialize USART properties
      UBRR0H = (uint8_t)(BAUD_PRESCALLER >> 8); // Store Upper Baudrate into UBRR0H
      UBRR0L = (uint8_t)(BAUD_PRESCALLER);      // Store Lower Baudrate into UBRR0L
      UCSR0B = (1 << RXEN0) | (1 << TXEN0);     // Enable Receiver and Enable Transmitter
      UCSR0C = (3 << UCSZ00);                   // Set UCSZ02:1 as 8-bit character data
}

unsigned char USART_receive(void) {             // Receive ASCII value from UDR0
      while (!(UCSR0A & (1 << RXC0)));          // Check RXC0 is 'High' to break loop
      return UDR0;                              // Return serial into unsigned char data
}

void USART_send(unsigned char data) {           // Transmit ASCII value into UDR0
      while (!(UCSR0A & (1 << UDRE0)));         // Check UDRE0 if 'High' to break loop
      UDR0 = data;                              // Store unsigned char data into UDR0
}

void USART_putstring(char* StringPtr) {         // Break string into chars and send
      while (*StringPtr != 0x00) {              // Keep Looping until String Completed
            USART_send(*StringPtr);             // Send char pointed by string pointer
            StringPtr++;                        // Increment pointer next char location
      }
}
//---------------------------------------------------------------------
void adc_init(void) {
      DIDR0 = 0x3F;                     // Disable Digital Input

      ADMUX = (0<<REFS1)|(1<<REFS0)|    // Reference Selection, AVcc - Ext cap at AREF
      (0<<ADLAR)|                       // ADC Left Adjust Result for 10-bit result
      (0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0);// Analog Channel Selection 'ADC0' (PC0)

      ADCSRA = (1<<ADEN)|               // ADC Enable
      (0<<ADSC)|                        // ADC Start Conversion
      (0<<ADATE)|                       // ADC Auto Trigger Enable
      (0<<ADIF)|                        // ADC Interrupt Flag
      (0<<ADIE)|                        // ADC Interrupt Enable
      (1<<ADPS2)|(0<<ADPS1)|(1<<ADPS0); // ADC Prescaler Select Bits '32'
}

void read_adc(void) {
      unsigned char i = 4;              // Set 'i' for iterations
      adc_val = 0;                      // set float 'adc_val'
      while (i--) {                     // Decrement 'i' until 4 samples take
            ADCSRA |= (1<<ADSC);        // If ADSC is high (ADC Start Conversion)...
            while (ADCSRA & (1<<ADSC)); // Start the ADC Conversion
            adc_val += ADC;             // Store the analog value on of current adc_val
            _delay_ms(50);              // delay 50ms for sampling
      }
      adc_val = (adc_val/4);            // Average of 4 samples taken into adc_val

      adc_val = (adc_val - 512)/2;      // Lets 10-bit Analog voltage become 8-bit value
```

```
}
```

3. INITIAL/DEVELOPED CODE OF TASK 2

```
/*
 * DA4B_Task_2.c
 *
 * Created: 4/19/2019 8:16:33 PM
 * Author: RYG95
 */

#define F_CPU 16000000UL    // Frequency of Xplained Mini (16MHz)
#include <avr/io.h>         // Standard AVR Library
#include <stdio.h>          // AVR library containing printf functions
#include <avr/interrupt.h>  // AVR library containing interrupt functions
#include <util/delay.h>     // AVR library containing _delay_ms() function

#define BAUDRATE 9600                                       // Baudrate (bps)
#define BAUD_PRESCALLER ((F_CPU / (BAUDRATE * 16UL)) - 1)   // Baudrate Prescaler

void Timer1_init(void);                 // Initialize Timer1 properties
void adc_init(void);                    // Initialize Analog to Digital Converter
void read_adc(void);                    // Read value received from ADC

void USART_init(void);                  // Initialize USART
unsigned char USART_receive(void);      // Receive Serial data from UDR0
void USART_send(unsigned char data);    // Send char data into UDR0
void USART_putstring(char* StringPtr);  // Break string into chars, send

volatile float adc_val;                 // Stores ADC Value
char outs[20];                          // Store float value into array of chars

int main(void) {

	DDRC = 0x00;                        // PC7:0 Inputs (PC0 Read ADC, PC1 Switch)
	PORTC = (1<<DDC1);                  // Set pull-up for PC1
	DDRB |= (1 << 1);                   // OC1A for ATmega328PB = PB1

	Timer1_init();                      // Call the Timer1/PWM initialization
	adc_init();                         // Call the ADC initialization code
	USART_init();                       // Call the USART initialization code
	USART_putstring("Connected!\r\n"); // Pass "Connected!" to function
	_delay_ms(125);                     // Wait a bit

	while (1) {                         // Infinite loop
		read_adc();                     // Read ADC Value
		ICR1 = 39999;                   // Set TOP count for timer1 in ICR1
		snprintf(outs,sizeof(outs),"%3f\r\n", adc_val);
		USART_putstring(outs);                  // Pass "outs" to function
		if (adc_val >= 3999) {                  // If 'adc_val >= 3999'...
			TCNT1 = 0;                          // Reset Timer1
			OCR1A = 3999;                       // Set timer1 to position 180 deg
			_delay_ms(100);                     // delay
		} else if ((adc_val < 3999) & (adc_val >= 1999)) {     // If between range
			TCNT1 = 0;                          // Reset Timer1
			OCR1A = adc_val;                    // Set timer1 to position ADC
			_delay_ms(100);                     // delay
```

```
		} else if (adc_val < 1999) {		// If 'adc_val < 1999'...
			TCNT1 = 0;			// Reset Timer1
			OCR1A = 1999;			// Set timer1 to position 0 deg
			_delay_ms(100);			// delay
		}
	}
}
//-----------------------------------------------------------------------
void Timer1_init(void) {				// Initialize Timer0 properties
	TCCR1A = (1 << COM1A1) | (1 << WGM11);
	TCCR1B = (1 << WGM13) | (1<< WGM12) | (1 << CS11);
}
//-----------------------------------------------------------------------
void USART_init(void) {					// Initialize USART properties
	UBRR0H = (uint8_t)(BAUD_PRESCALLER >> 8);	// Store Upper Baudrate in UBRR0H
	UBRR0L = (uint8_t)(BAUD_PRESCALLER);		// Store Lower Baudrate in UBRR0L
	UCSR0B = (1 << RXEN0) | (1 << TXEN0);		// Enable Receiver and Transmitter
	UCSR0C = (3 << UCSZ00);				// Set UCSZ02:1 as 8-bit char
}

unsigned char USART_receive(void) {			// Function to receive ASCII UDR0
	while (!(UCSR0A & (1 << RXC0)));		// Check RXC0 'High' to break loop
	return UDR0;					// Return serial into char data
}

void USART_send(unsigned char data) {			// Transmit ASCII value into UDR0
	while (!(UCSR0A & (1 << UDRE0)));		// Check UDRE0 'High' break loop
	UDR0 = data;					// Store char data into UDR0
}

void USART_putstring(char* StringPtr) {			// Break string into chars, send()
	while (*StringPtr != 0x00) {			// Loop until String Completed
		USART_send(*StringPtr);			// Send char by string pointer
		StringPtr++;				// Increment pointer to next char
	}
}
//-----------------------------------------------------------------------
void adc_init(void) {
	DIDR0 = 0x3F;				// Disable Digital Input

	ADMUX = (0<<REFS1)|(1<<REFS0)|		// Reference Selection, AVcc - Ext cap at AREF
	(0<<ADLAR)|				// ADC Left Adjust Result for 10-bit result
	(0<<MUX0);				// Analog Channel Selection Bits 'ADC0' (PC0)

	ADCSRA = (1<<ADEN)|			// ADC Enable
	(0<<ADSC)|				// ADC Start Conversion
	(0<<ADATE)|				// ADC Auto Trigger Enable
	(0<<ADIF)|				// ADC Interrupt Flag
	(0<<ADIE)|				// ADC Interrupt Enable
	(1<<ADPS2)|(1<<ADPS0);			// ADC Prescaler Select Bits '32'
}

void read_adc(void) {
	unsigned char i = 1;			// Set 'i' for iterations
	adc_val = 0;				// set float 'adc_val'
	while (i--) {				// Decrement 'i' until 4 samples take
		ADCSRA |= (1<<ADSC);		// If ADSC is high (ADC Start Conversion)...
		while (ADCSRA & (1<<ADSC)); // Start the ADC Conversion
```

```
            adc_val += ADC;             // Store the analog value on of current adc_val
            _delay_ms(50);              // delay 50ms for sampling
        }
        adc_val = (adc_val*4);          // Store  ADC value as readable
}
```

4. SCHEMATICS

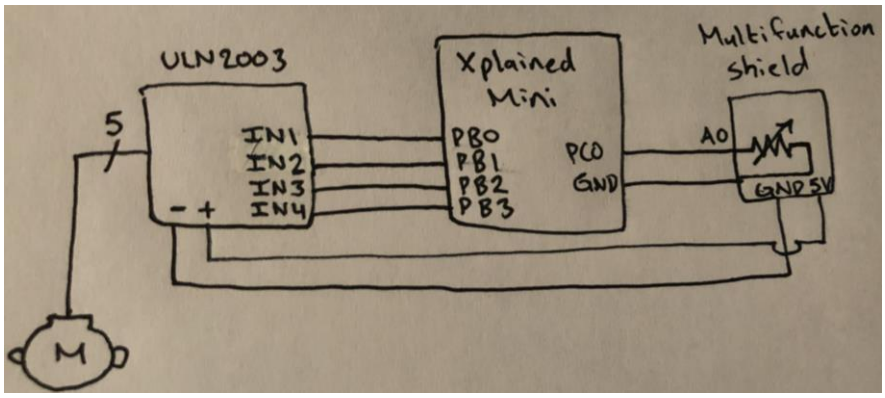

Figure 3 – Schematic of Stepper Motor and Potentiometer ADC

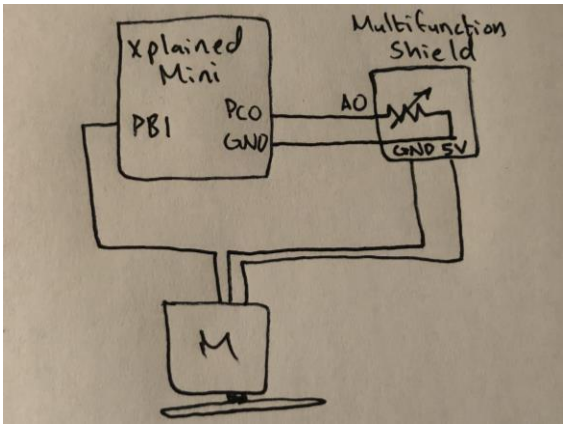

Figure 4 – Schematic of Servo Motor and Potentiometer ADC

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

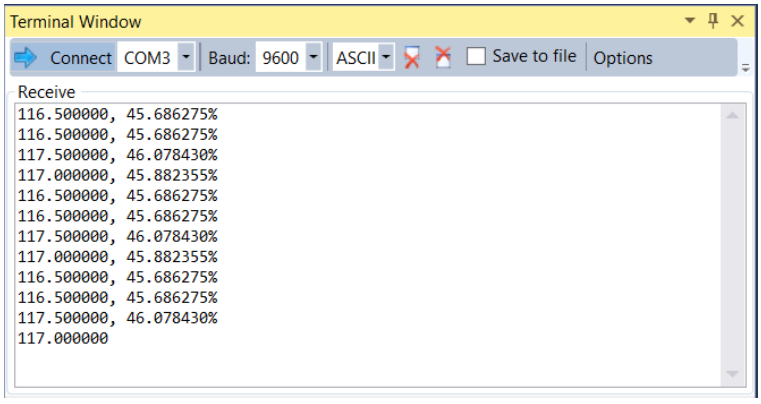

Figure 5 – Output Terminal for Controlling Stepper Motor

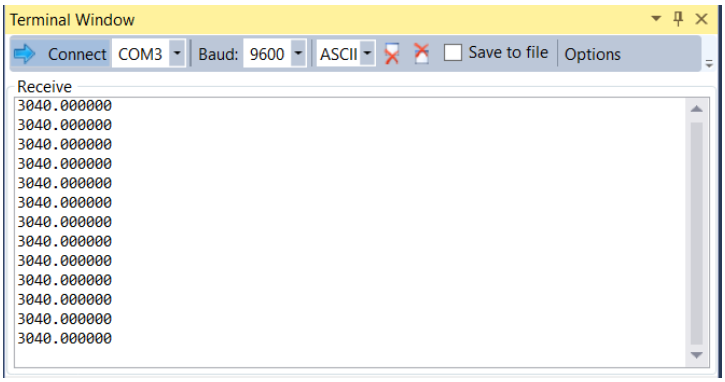

Figure 6 – Output Terminal for Controlling Servo Motor

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

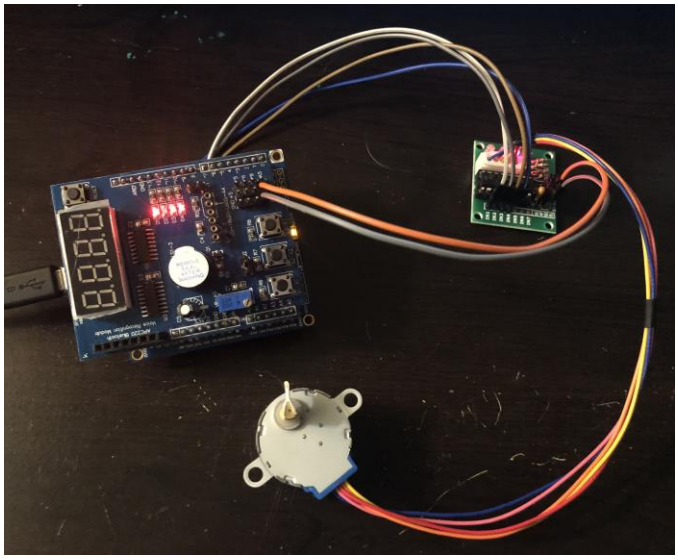

Figure 7 – Connecting Stepper Motor to Potentiometer

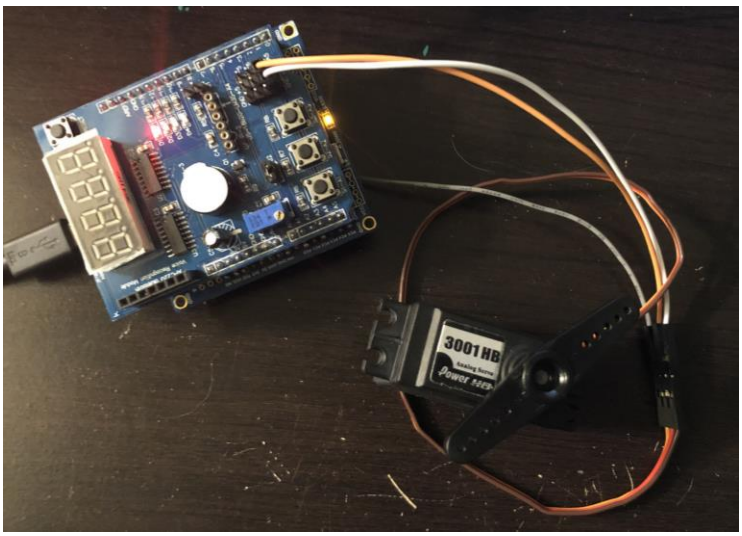

Figure 8 – Connecting Servo Motor to Potentiometer

7. VIDEO LINKS OF EACH DEMO

https://youtu.be/CwPMm-LuCxk

8. GITHUB LINK OF THIS DA

https://github.com/rockyg1995/ihswppdar/tree/master/DesignAssignments/DA4B

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

"*This assignment submission is my own, original work*".
Rocky Gonzalez