CPE301 – SPRING 2019

Design Assignment 4A

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\DA4A

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 CONNECTION BLOCK DIAGRAM w/ PINS

Atmega328PB Xplained Mini
Micro USB Cable (Power Supply)
Breadboard
x2 100nF Capacitors
10uH Inductor
Wire Connectors
DC Motor
L293
Pushbutton Switch
1k resistor
1k Potentiometer

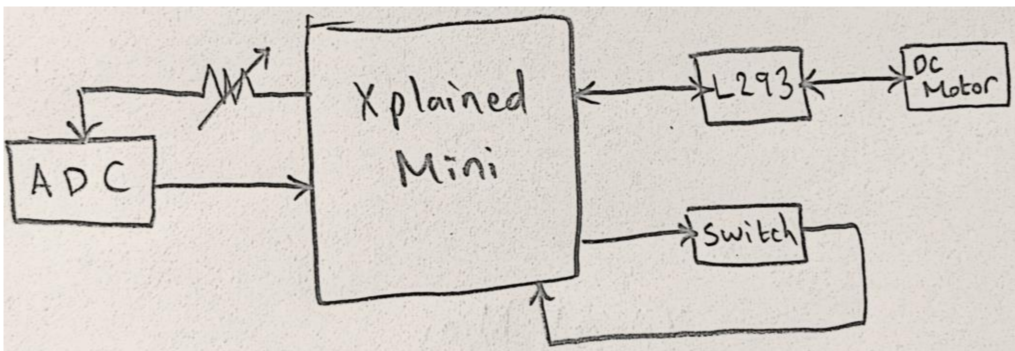

Figure 1 – Block Diagram

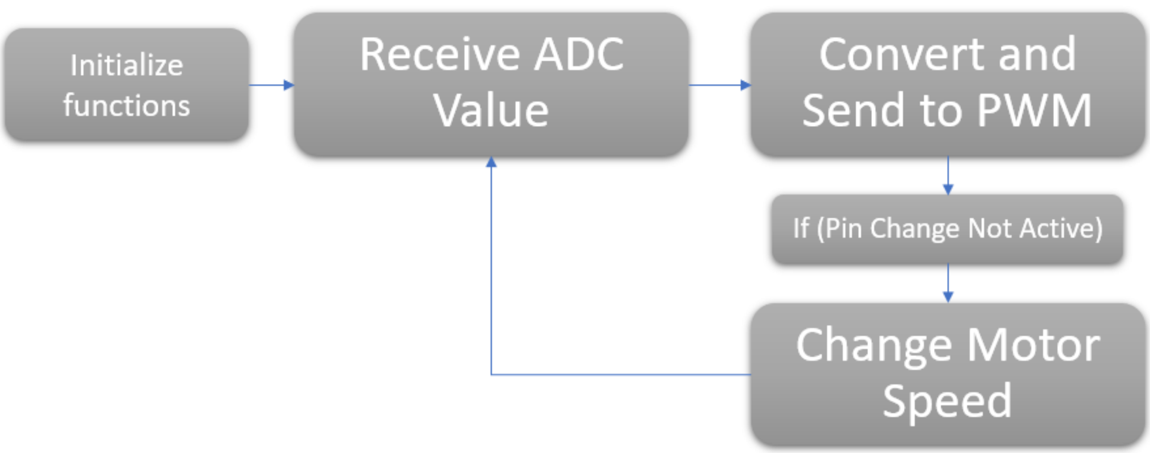

Figure 2 – Flow Chart for Coding Algorithm

2. INITIAL/DEVELOPED CODE OF TASK

```
/*
 * DA4A.c
 *
 * Created: 4/13/2019 3:36:09 PM
 * Author: RYG95
```

```c
 */

#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 in Bits per second (bps)
#define BAUD_PRESCALLER ((F_CPU / (BAUDRATE * 16UL)) - 1)     // Baudrate UBRR0H:UBRR0L

//Declaration of our functions
void Timer0_init(void);            // Function to Initialize Timer1 properties
void adc_init(void);               // Function to initialize Analog to Digital Converter
void read_adc(void);               // Function to read temperature received from ADC

void USART_init(void);                       // Function to initialize USART
unsigned char USART_receive(void);           // Function to receive Serial data from UDR0
void USART_send(unsigned char data);         // Function to send individual char into UDR0
void USART_putstring(char* StringPtr);       // Function to break string into chars and send

volatile float adc_val;                      // Stores ADC Value representing Analog Voltage
char outs[20];                               // 'outs[]' used to store integer and float
                                             // values into array of chars size 20

int main(void) {

       DDRC = 0x00;                // All of PC7:0 Inputs (PC0 Input ADC, PC1 Switch)
       PORTC = (1<<DDC1);                    // Set pull-up for PC1
       DDRD |= (1 << DDD6)|(1 << DDD7);      // PD6 and PD7 are Outputs (DC Motor)
       PORTD |= (1 << DDD6)|(0 << DDD7);     // PD6 is HIGH, PD7 is LOW (DC Motor)
//     PCMSK1 = (1<<PCINT9);                 // Set Pin Change Interrupt on PC1
//     PCICR |= (1<<PCIE1);
       sei();                                // Enable Global Interrupts

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

       while (1) {                           // Infinite loop
              read_adc();                    // Read value of ADC Value
              snprintf(outs,sizeof(outs),"%3f", adc_val);// Stores 'adc_temp' in 'outs'
              USART_putstring(outs);         // Pass 'outs' to function to send chars
              _delay_ms(250);

              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)) {     // Else, if ‘5<adc_val<242’
                     OCR0A = adc_val;                          // Duty Cycle ‘%’ = 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) {                        // Else, if ' adc_val < 5'
```

```c
                        OCR0A = adc_val;                        // Drop Duty Cycle to 0%
                        USART_putstring(", 0%\r\n");
                }
        }
        return 0;
}
//------------------------------------------------------------------------
/*
ISR(PCINT1_vect) {                  // Pin Change Interrupt to toggle DC MOTOR
        _delay_ms(250);             // Prevent debounce error
        while (PINC1 != 0);         // Stuck in loop until switch let go
        _delay_ms(250);             // Prevent debounce error
        PORTD ^= (1 << DDD7);       // Toggle PD7 (DC Motor) Break/Spin
        };
*/
//------------------------------------------------------------------------
void Timer0_init(void) {                          // Initialize Timer0 properties
        TCCR0A |= (1 << COM0A1);                  // Set non-inverting mode
        TCCR0A |= (1 << WGM01) | (1 << WGM00);    // Set fast PWM Mode
        TCCR0B |= (1 << CS02);                    // Set prescaler to 256 and starts PWM
}
//------------------------------------------------------------------------
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 Enable Transmitter
        UCSR0C = (3 << UCSZ00);                   // Set UCSZ02:1 as 8-bit character data
}

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

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

void USART_putstring(char* StringPtr) {           // Function to break string into chars
        while (*StringPtr != 0x00) {              // Keep Looping until String Completed
                USART_send(*StringPtr);           // Send string
                StringPtr++;                      // Increment to next char array location
        }
}
//------------------------------------------------------------------------
void adc_init(void) {
        DIDR0 = 0x3F;                     // Disable Digital Input

        ADMUX = (0<<REFS1)|(1<<REFS0)|    // Reference Select Bits, 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 Bits 'ADC0'

        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. SCHEMATICS

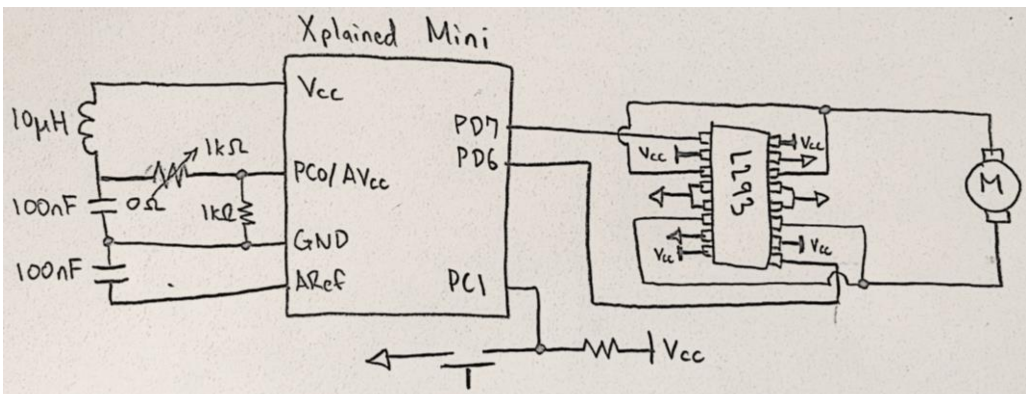

Figure 3 – Schematic of DC Motor and Potentiometer ADC

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

Since the Atmega328P/PB Xplained Mini has a built in AVR RX/TX module, we can send our serial data into a terminal which is able to display the temperature reading of ADC when connected to a temperature sensor (LM35) sending analog data into ADC channel 5. The output is portrayed in *Figure 4*:

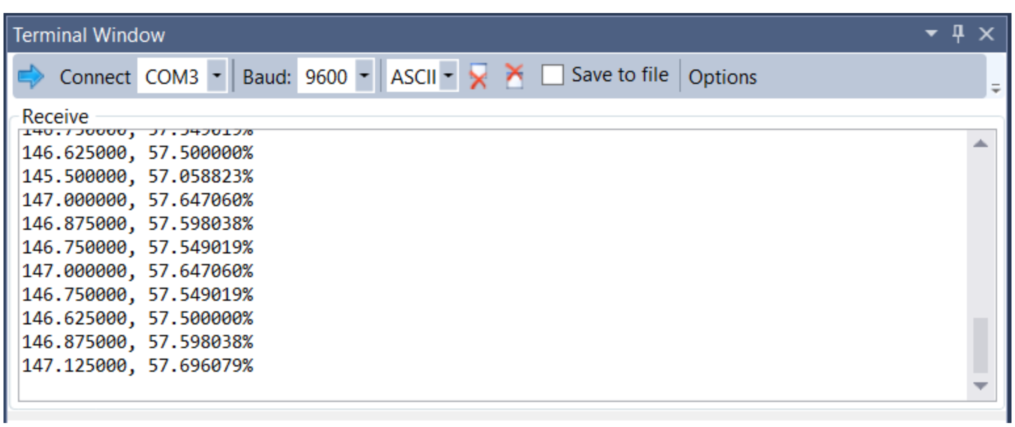

Figure 4 – Output Terminal of Serial Data

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

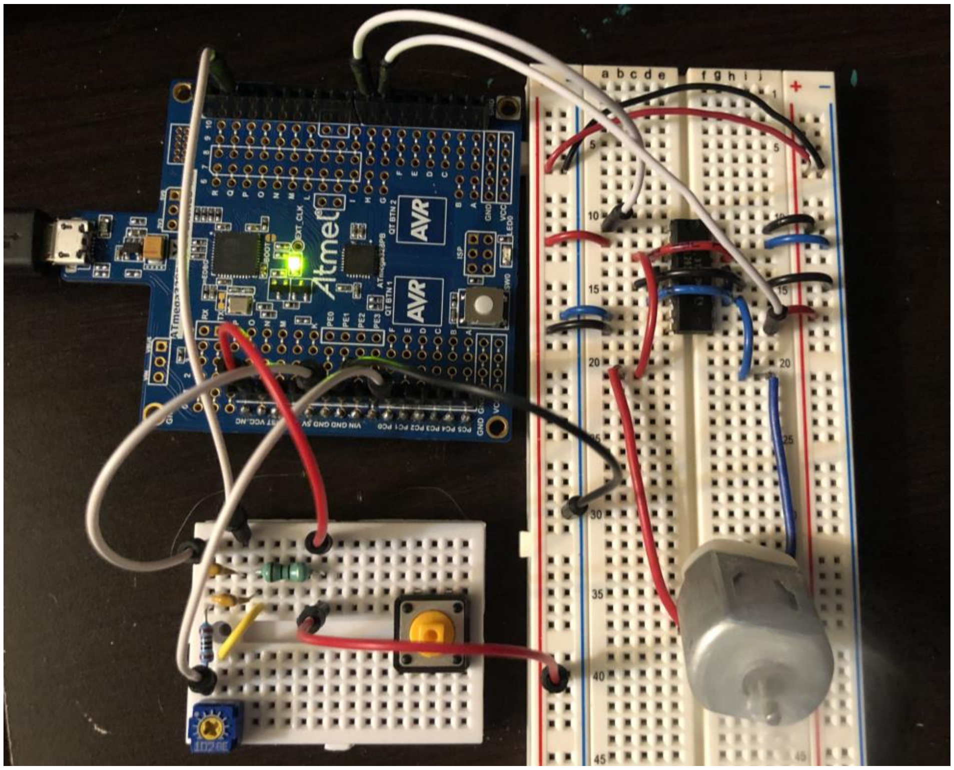

Figure 5 – Xplained Mini connected to Potentiometer and DC Motor

6. VIDEO LINKS OF EACH DEMO

https://youtu.be/KNZYYGoeRr4

7. GITHUB LINK OF THIS DA

https://github.com/rockyg1995/ihswppdar/tree/master/DesignAssignments/DA4A

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

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