CPE301 – SPRING 2019

Design Assignment 4B

Student Name: Michael Johnson

Student #: 2000878717

Student Email: johnsm17@unlv.nevada.edu

Primary Github address: https://github.com/miggnuggets/submissions.git

Directory: Repository\_301

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Atmega328P

Potentiometer

UNL2003 Driver Breakout Board

Stepper Motor

Servo Motor

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

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdio.h>

int start\_value = 1699; // Start Timer1

// Function Declarations

void read\_adc(void);

void adc\_init(void);

volatile unsigned int adc\_value;

volatile int OVFCount; // Setup global overflow counter

volatile double float\_value;

volatile int RPMvalue;

float c = 1699; // Needed offset for Affine Transformation

char outs[20];

int main(void)

{

adc\_init(); // Start ADC

*\_delay\_ms*(125); // Wait

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

/\*

CTC Mode with Prescalar to 8

\*/

TIMSK1 |= (1<<OCIE1A);

/\*

Turn on interrupt Mask for OCR1A

\*/

OCR1A = start\_value; // Start Timer at longest possible value, 20,000us

sei();

//Test of timer

DDRD = 0xFF; // Set PortD as an output

OVFCount = 0; // Start global overflow counter

while (1)

{

read\_adc();

/\*

Affine Transformation

PWM output = adc\_value\*(39999-1699)/(1024-0) + 1699

\*/

float\_value = adc\_value;

RPMvalue = (float\_value\*37.4033) + c;

OCR1A = RPMvalue; // Load newest time measurement value

}

}

/\* Timer 1 Compare Interrupt Routine \*/

ISR(TIMER1\_COMPA\_vect)

{

OVFCount++; // Increment overflow flag counter

int count; // Start local counter

count = OVFCount; // Let local counter mimic global counter variable

// Step 1

if(count == 1){

PORTD = 0xE;

}

// Step 2

if(count == 2) {

PORTD = 0xC;

}

// Step 3

if(count == 3) {

PORTD = 0xD;

}

// Step 4

if(count == 4) {

PORTD = 0x9;

}

// Step 5

if(count == 5) {

PORTD = 0xB;

}

// Step 6

if(count == 6) {

PORTD = 0x3;

}

// Step 7

if(count == 7) {

PORTD = 0x7;

}

// Step 8

if(count == 8) {

PORTD = 0x6;

}

// Reset Counter if gone through all steps

if(count == 8){

OVFCount = 0; // Restart Stepper Counter

}

}

/\* ADC \*/

void adc\_init(void)

{

/\*\* Setup and enable ADC \*\*/

ADMUX = (0<<REFS1)| // Reference Selection Bits

(1<<REFS0)| // AVcc - external cap at AREF

(0<<ADLAR)| // ADC Left Adjust Result

(0<<MUX2)| // Analog Channel Selection Bits

(0<<MUX1)| // ADC0 (PC0 PIN23)

(0<<MUX0);

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)| // ADC Prescaler Select Bits

(0<<ADPS1)|

(1<<ADPS0);

}

/\* READ ADC PINS \*/

void read\_adc(void)

{

unsigned char i = 4;

adc\_value = 0;

while (i--)

{

ADCSRA |= (1<<ADSC);

while(ADCSRA & (1<<ADSC));

adc\_value+= ADC;

*\_delay\_ms*(50);

}

adc\_value = adc\_value / 4; // Average some samples

}

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

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

#define BAUDRATE 9600

#define BAUD\_PRESCALLER (((*F\_CPU* / (BAUDRATE \* 16UL))) - 1)

void read\_adc(void); // Function Declarations

void adc\_init(void);

void USART\_init( unsigned int ubrr );

void USART\_tx\_string( char \*data );

volatile unsigned int adc\_value;

volatile double float\_value;

volatile int PWMvalue;

float c = 97; // Needed offset for Affine Transformation

char outs[20];

int main(void)

{

// Configure Timer 1

TCCR1A |= (1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);

/\*

Non inverted PWM

Prescalar = 64

WGM Mode 14 - Fast PWM

\*/

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

ICR1 = 4999;

/\*

PWM Frequency = 50Hz, Period = 20ms standard for servo motor operation

\*/

DDRB |= (1<<PORTB1);

adc\_init(); // Start the ADC

USART\_init(BAUD\_PRESCALLER); // Start the USART

USART\_tx\_string("Connected!\r\n"); // we're alive!

*\_delay\_ms*(125); // wait a bit

while(1)

{

read\_adc();

/\*

Affine Transformation

PWM output = adc\_value\*(535-97)/(1024-0) + 97

\*/

float\_value = adc\_value;

PWMvalue = (float\_value\*0.4277343) + c;

*snprintf*(outs,sizeof(outs),"%3d\r\n", PWMvalue); // print it

USART\_tx\_string(outs);

OCR1A = PWMvalue;

*\_delay\_ms*(125); // Wait

}

}

/\* Start ADC \*/

void adc\_init(void)

{

/\*\* Setup and enable ADC \*\*/

ADMUX = (0<<REFS1)| // Reference Selection Bits

(1<<REFS0)| // AVcc - external cap at AREF

(0<<ADLAR)| // ADC Left Adjust Result

(0<<MUX2)| // Analog Channel Selection Bits

(0<<MUX1)| // ADC0 (PC0 PIN23)

(0<<MUX0);

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)| // ADC Prescaler Select Bits

(0<<ADPS1)|

(1<<ADPS0);

}

/\* READ ADC PINS \*/

void read\_adc(void)

{

unsigned char i = 4;

adc\_value = 0;

while (i--)

{

ADCSRA |= (1<<ADSC);

while(ADCSRA & (1<<ADSC));

adc\_value+= ADC;

*\_delay\_ms*(50);

}

adc\_value = adc\_value / 4; // Average some samples

}

/\* Start USART (RS-232) \*/

void USART\_init( unsigned int ubrr )

{

UBRR0H = (unsigned char)(ubrr>>8);

UBRR0L = (unsigned char)ubrr;

UCSR0B = (1 << TXEN0); // Enable RX, TX & RX interrupt

UCSR0C = (3 << UCSZ00); //asynchronous 8 N 1

}

/\* SEND A STRING TO THE RS-232 \*/

void USART\_tx\_string( char \*data )

{

while ((\*data != '\0'))

{

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

UDR0 = \*data;

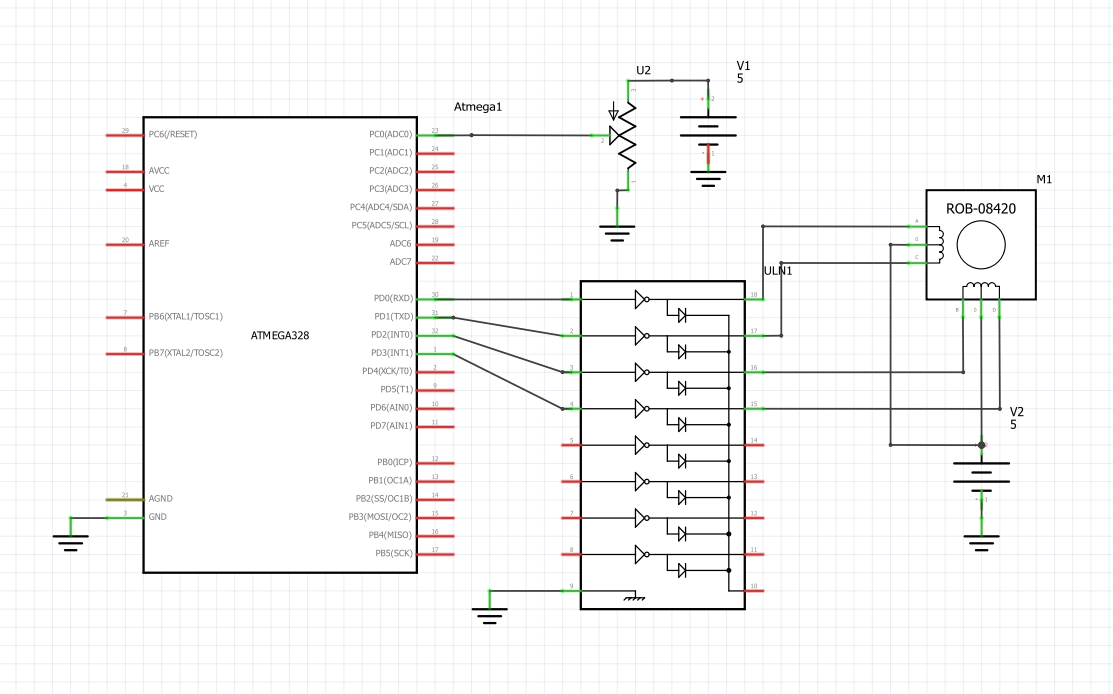
data++;

}

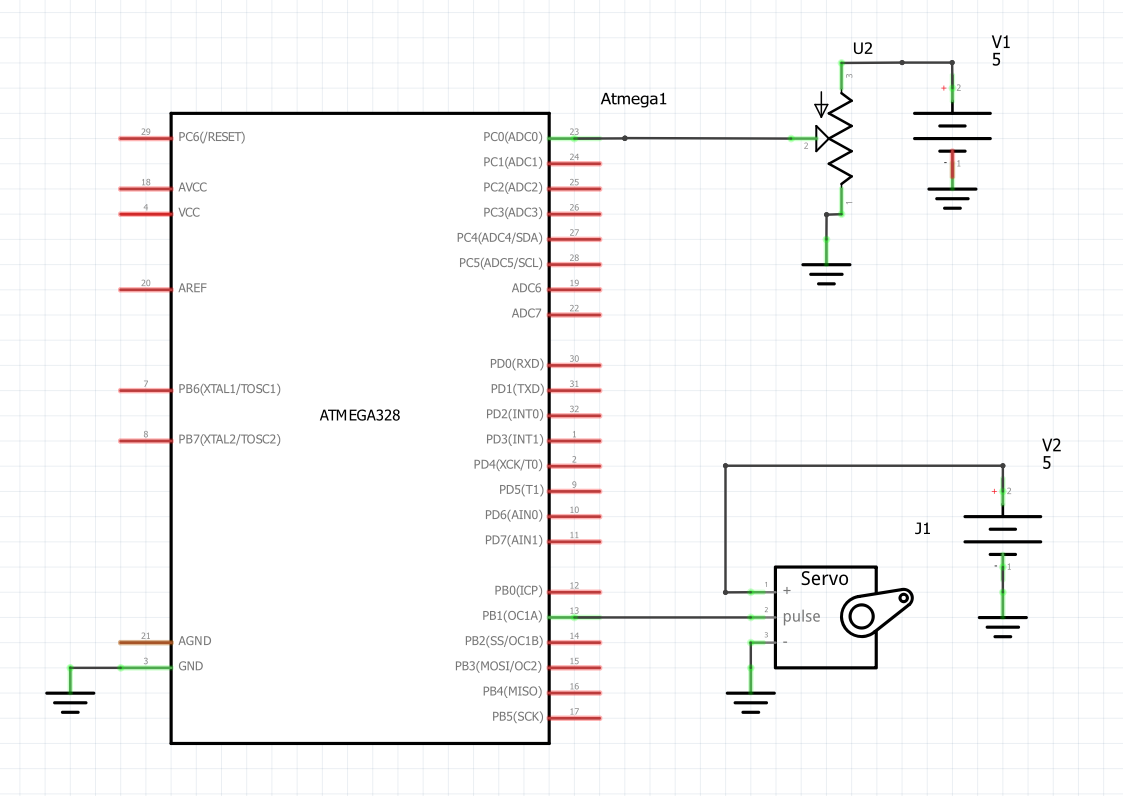
}

1. **SCHEMATICS**

Task 1:



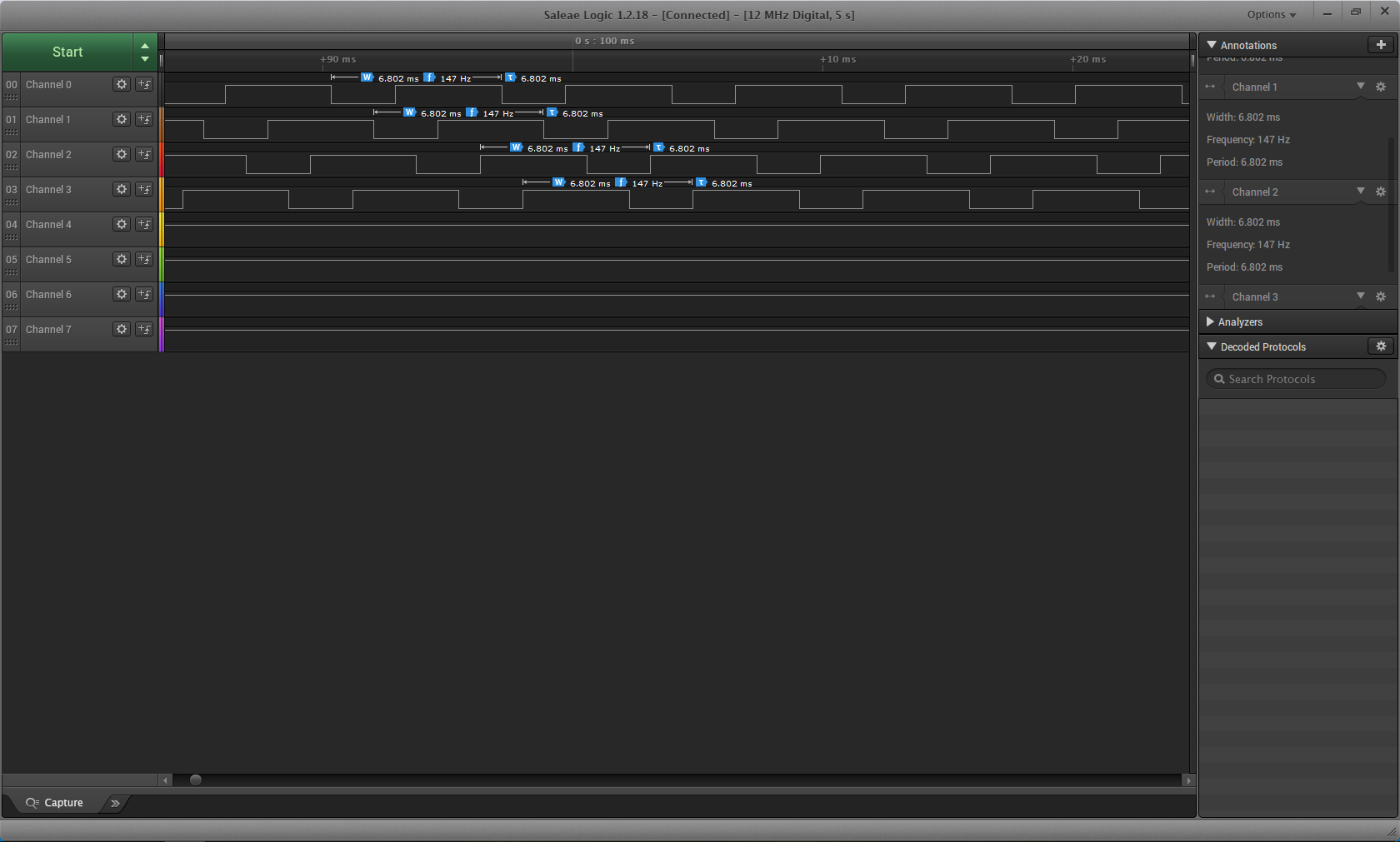
Task 2:



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

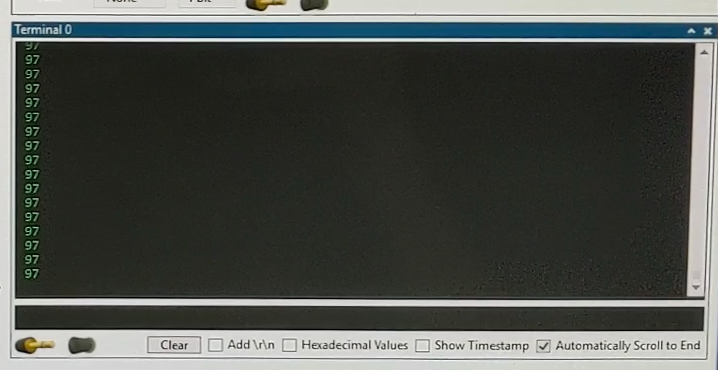
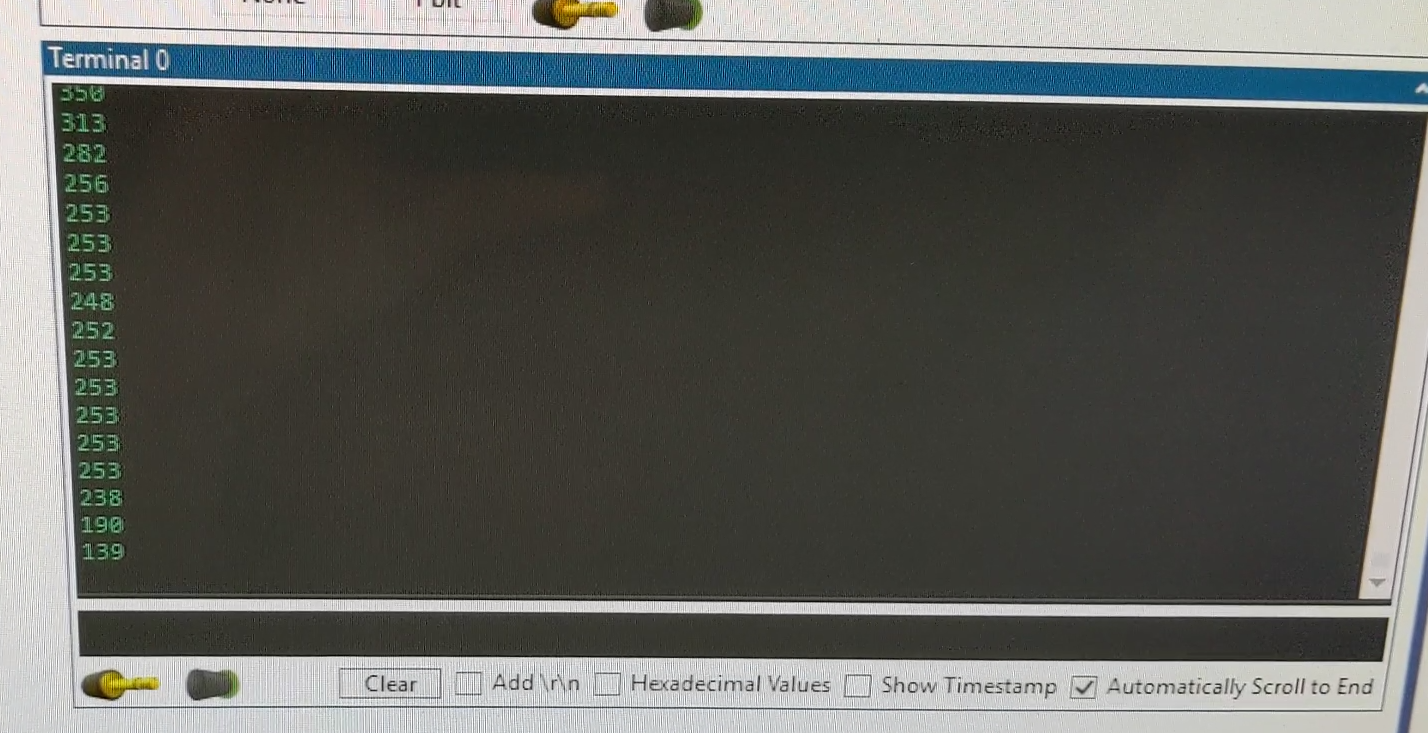
Task 1:

Output Waveform:



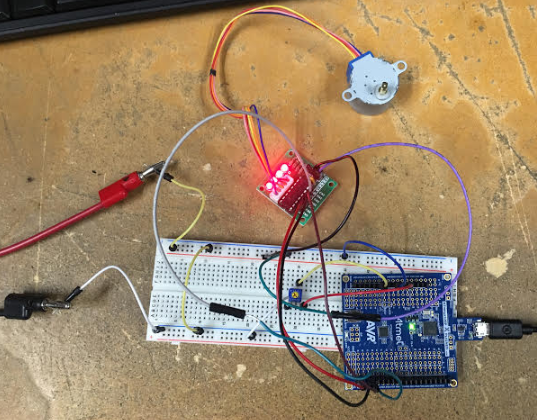
Task 2:

ADC from change in position

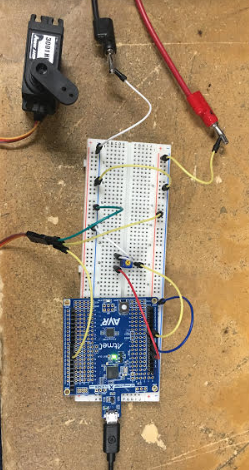


1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

Task 1:



Task 2:



1. **VIDEO LINKS OF EACH DEMO**

**https://youtu.be/LfuonSIHfHg**

1. **GITHUB LINK OF THIS DA**

[**https://github.com/miggnuggets/submissions.git**](https://github.com/miggnuggets/submissions.git)

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Michael Johnson