CPE301 – SPRING 2019

Design Assignment 4B

Student Name: Henry Mesa

Student #: 1008099273

Student Email: mesah1@unlv.nevada.edu

Primary Github address: https://github.com/mesah1

Directory: https://github.com/mesah1/submissions

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

List of Components used

ATmega328pb

Multi-Function Shield

Analog Servo 3001HB

Stepper Motor 28BYJ-48

MD08A

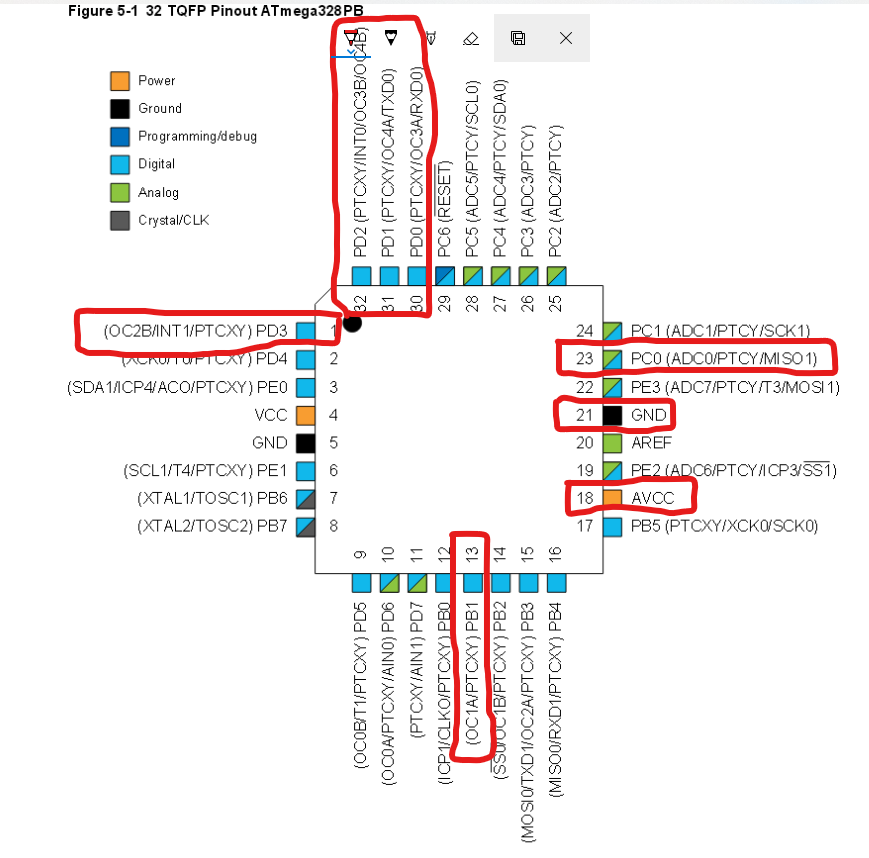
ULN2003AN

Power Supply

Breadboard

Wires

Block diagram with pins used in the Atmega328P



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

/\*

\* DA4B.c

\*

\* Created: 11/11/2019 9:23:33 AM

\* Author : Henry Mesa

\* THIS PROGRAM WILL CONTROL THE VELOCITY OF A STEPPER MOTOR

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

//Functions to manipulate ADC input

void read\_adc(void);

void adc\_init(void);

//ADC storage

volatile float adc;

int main(void)

{

adc\_init(); //Initializing ADC

//Configuring input and output pins

PORTC |= 0b00000001; //input pull-up enable PC0

DDRD = 0xFF; DDRB = 0xFF;//outputs

//Configuration of Timer1

TCCR1A |= (1 << COM1A1) |(1 << COM1B1) |(1 << WGM11); //non-inverted PWM

//PRESCALER = 64

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

// freq = 50Hz Period = 20ms

ICR1 = 4999;

while (1)

{

read\_adc(); //Read ADC value

*\_delay\_ms*(25);

while (1)

{

//Stepper Motor Instructions

PORTD = 0b01100110;

*\_delay\_ms*(50);

PORTD = 0b11001100;

*\_delay\_ms*(50);

PORTD = 0b10011001;

*\_delay\_ms*(50);

PORTD = 0b00110011;

*\_delay\_ms*(50);

if ((adc >= 973) && (adc < 1024 )) //when adc value is at its max

{ //delay the least

*\_delay\_ms*(10);

}

else if ((adc < 972) && (adc >= 768)) //decreasing pwm as resistor

{ //value decreases

*\_delay\_ms*(50);

}

else if ((adc < 767) && (adc >= 51)) //decreasing pwm as resistor

{ //value decreases

*\_delay\_ms*(100);

}

else

{

PORTD = 0b00000000;//if resistor value is less than 5% then pwm is 0%

*\_delay\_ms*(100);

}

}

}

}

/\* INITIALIZING ADC \*/

void adc\_init(void){

//ADC ENABLE AND SET-UP

ADMUX = (0 << REFS1)| // external cap at AREF

(1 << REFS0)| // AVcc - Reference Selection INTERNAL

(1 << 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 128

(0 << ADPS1)|

(1 << ADPS0);

}

/\* READ ADC PINS \*/

void read\_adc(void){

unsigned char i = 4;

adc = 0;

while (i--)

{

ADCSRA |= (1 << ADSC);

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

adc+= ADC;

*\_delay\_ms*(50);

}

adc = adc / 8; // Average a few samples

}

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

/\*

\* DA4B.c

\*

\* Created: 11/11/2019 9:23:33 AM

\* Author : Henry

\* THIS PROGRAM CONTROLS DIRECTION OF MOVEMENT OF A SERVO MOTOR

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

//Functions to manipulate ADC input

void read\_adc(void);

void adc\_init(void);

//ADC storage

volatile float adc;

int main(void)

{

adc\_init(); //Initializing ADC

//Configuring input and output pins

PORTC |= 0b00000001; //input pull-up enable PC0

DDRD = 0xFF; DDRB = 0xFF;//outputs

//Configuration of Timer1

TCCR1A |= (1 << COM1A1) |(1 << COM1B1) |(1 << WGM11); //non-inverted PWM

//PRESCALER = 64

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

// freq = 50Hz Period = 20ms

ICR1 = 4999;

while (1)

{

read\_adc(); //Read ADC value

*\_delay\_ms*(25);

//Servo Motor Command

OCR1A = adc;

*\_delay\_ms*(250);

}

}

/\* INITIALIZING ADC \*/

void adc\_init(void){

//ADC ENABLE AND SET-UP

ADMUX = (0 << REFS1)| // external cap at AREF

(1 << REFS0)| // AVcc - Reference Selection INTERNAL

(1 << 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 128

(0 << ADPS1)|

(1 << ADPS0);

}

/\* READ ADC PINS \*/

void read\_adc(void){

unsigned char i = 4;

adc = 0;

while (i--)

{

ADCSRA |= (1 << ADSC);

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

adc+= ADC;

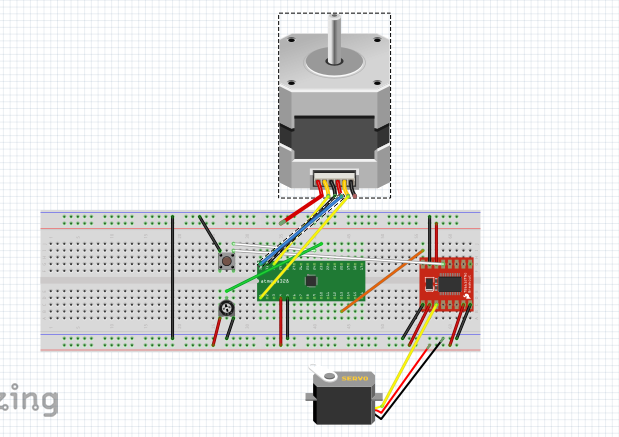
*\_delay\_ms*(50);

}

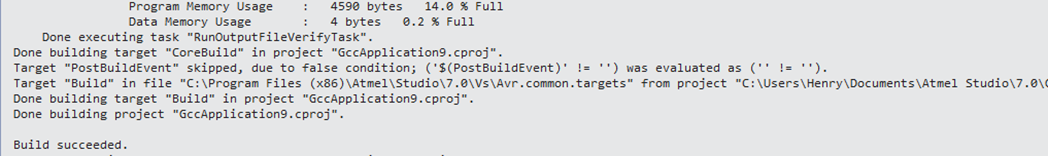
adc = adc / 8; // Average a few samples

}

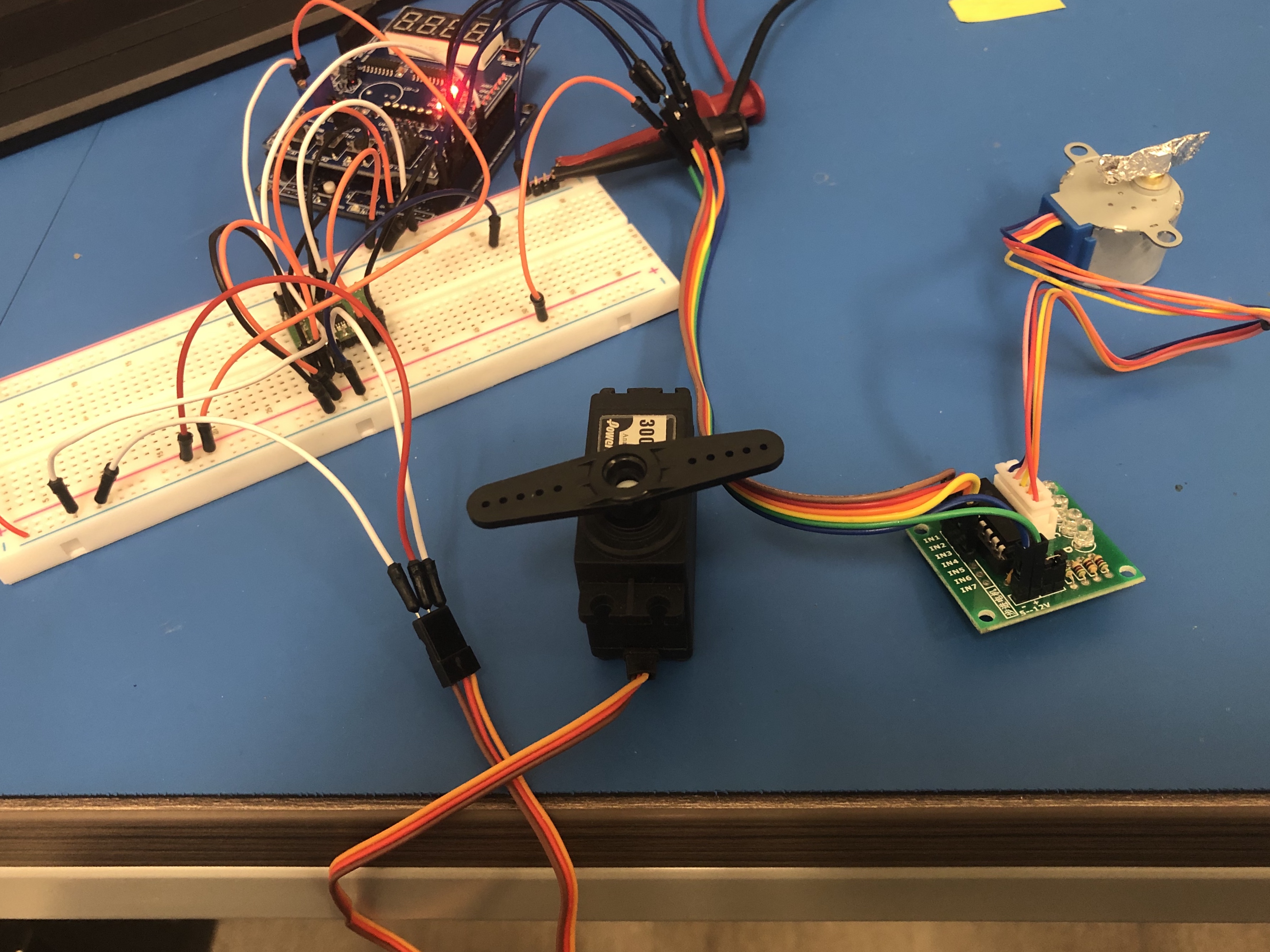
1. **SCHEMATICS**



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



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



1. **VIDEO LINKS OF EACH DEMO**

<https://youtu.be/4UEZMuFaITc>

1. **GITHUB LINK OF THIS DA**

<https://github.com/mesah1/submissions/tree/master/DA4B>

**Student Academic Misconduct Policy**

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

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

Henry Mesa