CPE301 – SPRING 2019

Design Assignment 4B

Student Name: Ali Asadi

Student #: 5004200625

Student Email: aliasadivet@gmail.com

Primary Github address: https://github.com/aliasadivet/Submitions\_DA\_AliAsadi

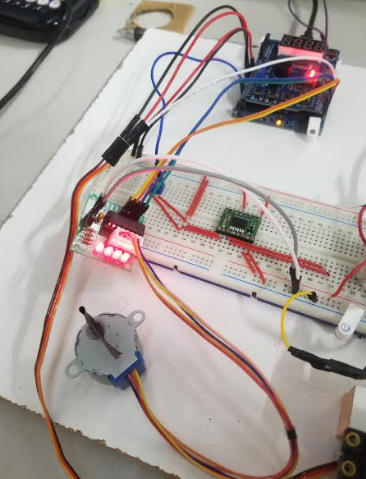
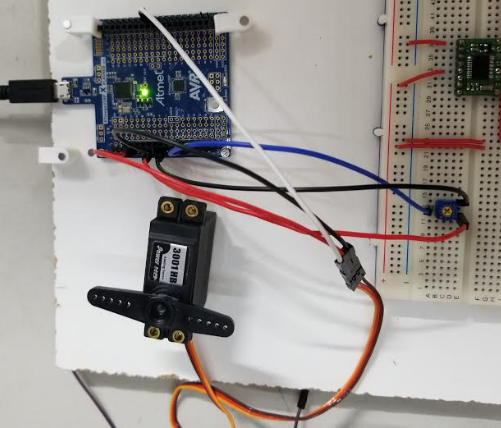
Directory: DessignAssignments

In this assignment we need to write a C program to control the speed of the DC Motor using a potentiometer connected to any of the analog-in port. An interrupt on a button will stop and start the motor at each click. The minimum speed of the motor should be 0 when pot is minimum and maximum should be 95% of PWM value.

List of Components used:

1. 5V DC Stepper motor
2. 3001 HB Servo motor
3. 1 kOhm Potentiometer
4. Atmega 328p Xplained Mini board
5. Breadboard
6. 22G jumper wires

The following is a screenshot of stepper motor and servo motor connections.



Programming:

/\*

\* DA4B.c

\*

\* Created: 4/22/2019 12:46:51 PM

\* Author : Ali Asadi

\*/

#include <inttypes.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/sleep.h>

#include <util/delay.h>

void Timer0\_init();

void Timer1\_init();

void Timer2\_init();

void ADC\_init();

void Port\_init();

*uint16\_t* ReadADC(*uint8\_t* );

void INT0\_init();

void Servo();

void Stepper();

void Motor();

const char steps[] = {0x30,0x90,0xC0,0x60}; // winding engerzising sequence

*uint16\_t* delay = 0; // Stepper Motor Speed

volatile *uint8\_t* Motor\_ON = 0; // Brushed Motor state

ISR(TIMER0\_COMPA\_vect)

{

static *int8\_t* step = 0; // initial step

static *uint16\_t* delay\_expired = 0; // delay timer

if(delay\_expired++ > delay) // check if specified delay expired

{

PORTD = (PORTD&0x0F)|steps[step++]; // Engerzise stepper motor winding

if(step > 3) step = 0; // if step exceeds maximum number of sequence then reset

delay\_expired = 0; // reset delay timer

}

}

ISR(INT0\_vect)

{

*uint8\_t* tmp = PIND & \_BV(PD2); // read the PIN D2

*\_delay\_ms*(10); // debounce Delay

if(tmp == (PIND & \_BV(PD2))) // check if the PIN is stable

Motor\_ON = Motor\_ON ? 0 : 1 ; // if stable toggle the Motor between ON or OFF state

}

int main()

{

Port\_init(); // init the PORT

ADC\_init(); // ADC enable

INT0\_init(); // init the Interrupt for the button

Timer0\_init(); // Timer 0 for Stepper Motor

Timer1\_init(); // Timer 1 for Servo Motor

Timer2\_init(); // Timer 2 for Motor

sei(); // enable global interrupt

while (1)

{

//Motor();

Servo();

//Stepper();

}

return 0;

}

void Port\_init()

{

DDRB = \_BV(PB0)|\_BV(PB1)|\_BV(PB2)|\_BV(PB3); // DC and Servo Motor Pin Connection

DDRD = \_BV(PD7)|\_BV(PD6)|\_BV(PD5)|\_BV(PD4); // Stepper Motor pin connection

PORTB = \_BV(PB0); // set high

PORTD = \_BV(PD2);

}

void Timer0\_init()

{

TCCR0A = \_BV(WGM01); // Timer 0 CTC mode

TCCR0B = \_BV(CS01)|\_BV(CS00); // Clk / 64 , 16M / 64 = 4uS

OCR0A = 249; // 4u \* 250 = 1mS delay generation

TIMSK0 = \_BV(OCIE0A); // Enable Output compare match A interrupt

}

void Timer1\_init()

{

TCCR1A = \_BV(COM1B1)|\_BV(WGM11)|\_BV(WGM10); // non-inverting mode, Fast PWM mode

TCCR1B = \_BV(WGM13)|\_BV(WGM12)|\_BV(CS11)|\_BV(CS10);// Fast PWM mode with OCR1A as TOP, clk / 64 = 4uS

OCR1A = 4999; // 5000 \* 4u = 20mS

}

void Timer2\_init()

{

TCCR2A = \_BV(COM2A1)|\_BV(WGM20); // non-inverting mode, Phase Correct PWM

TCCR2B = \_BV(CS21)|\_BV(CS20); // Clk / 32 = 8uS

}

void ADC\_init()

{

ADMUX = \_BV(REFS0); // AVcc as Reference

ADCSRA = \_BV(ADEN)|\_BV(ADPS0); // Enable ADC, Clk / 2

}

*uint16\_t* ReadADC(*uint8\_t* Channel)

{

ADMUX = (ADMUX & 0xF0)|(Channel & 0x0F); // Select channel

ADCSRA |= \_BV(ADSC); // start conversion

while(!(ADCSRA & \_BV(ADIF))); // wait for completion

ADCSRA |= \_BV(ADIF); // clear flag

return ADC; // return the converted value

}

void INT0\_init()

{

EICRA = \_BV(ISC01); // falling edge

EIMSK = \_BV(INT0); // Enable Interrupt 0

}

void Servo()

{

OCR1B = ((ReadADC(1) / 1024.0) \* 400) + 125; // Read ADC Convert to value in the range of 125 - 525 to get 0.5mS to 2mS

}

void Stepper()

{

delay = 1024 - ReadADC(2); // inverse the read value

}

void Motor()

{

if(Motor\_ON)

OCR2A = (ReadADC(0) / 1024.0) \* 242; // Motor duty cycle from 0 - 95

else

OCR2A = 0; // turn OFF

}

The execution results are posted on YouTube, and can be found on link below:

<https://www.youtube.com/watch?v=iL3hGHAPK3Q>