Eilat Avidan

CPE301 – SPRING 2018

Design Assignment 3

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E		
4.	SCHEMATICS		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
5.	SCREENSHOT OF EACH DEMO		
6.	VIDEO LINKS OF EACH DEMO		
7.	GOOGLECODE LINK OF THE DA		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

- Atmega328P
- Switch
- Breadboard
- Resistors
- Power supply
- 3 types of motors: DC, servo, and stepper.
- Potentiometer
- ULN2003 (works with the stepper motor)

2. INITIAL/DEVELOPED CODE OF TASK 1/A

DC Motor code:

```
#include <avr/io.h>
#define  F_CPU 8000000UL
#include <avr/interrupt.h>
#include <util/delay.h>



int main()
{

	ADCSRA = 0x87;			//ADC enable , prescaler 128
	ADMUX = 0x60;			//AVcc , left justified
	DDRB = 0xFF;			//PORTB output
	PORTD |= (1<<2);		//set up pull up resistor
	OCR1A = 0;			//0% duty cycle initially
	TCCR1B = 0x0D;			//prescaler of 1024
	TCCR1A = 0x83;			//non-inverting mode, fast PWM 10 bit
	EIMSK |= (1<<INT0);		//enable external interrupt 0
	EICRA |= (1<<ISC01);		//falling edge trigger
	sei();

	while (1)
	{
		ADCSRA |= (1<<ADSC);			//start conversion
		while ((ADCSRA & (1<<ADIF)) == 0)
		{
			//wait for conversion to finish
		}

	}
}
ISR (INT0_vect)
{

	EIFR |= (1<<INTF0);		//reset flag
		if((PORTB & 0b00000001) == 0b00000000)
			PORTB |= (1<<0);
			else
```

```
                PORTB &= ~(1<<0);

            if(ADCH > 220){
            OCR1A = 600;          //95% duty cycle
            _delay_ms(2000);
            }
            else
            OCR1A = 0;            //0% duty cycle

}
```

Stepper Motor code:

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/interrupt.h>

void delay_count();
int ADCvalue;

int main(void)
{
	DDRC = 0x00;                        //set port c as an input
	DDRD = 0xFF;                        //set port d as an output
	ADMUX = 0x00;                       //use ADC0 connected to the potentiometer
	ADMUX |= (1<<REFS0);          //AVcc
	ADMUX |= (1<<ADLAR);          //right adjust
	ADCSRA |= (1<<ADPS2) |(1<<ADPS1) |(1<<ADPS0); //prescaler of 128
	ADCSRA |= (1<<ADEN);          //ADC enable
	ADCSRA |= (1<<ADATE);               //auto trigger enable
	ADCSRB = 0;                                 //free mode
	ADCSRA |= (1<<ADIE);          //enable interrupt
	ADCSRA |= (1<<ADSC);          //start conversion
	sei();                                      //enable interrupt

	while (1)
	{
		{
			PORTD = 0x03;         //clockwise direction
			delay_count();;
			PORTD = 0x42;
			delay_count();
			PORTD = 0xC0;
			delay_count();
			PORTD = 0x81;
			delay_count();

		}

	}
}


void delay_count ()
{
```

```
	TCNT0 = 0;                                        //initial count value
	OCR0A = ADCvalue/7;                          //compare value
	TCCR0A = (1<<WGM01);                  // Configure timer0 for CTC mode.
	TCCR0B = (1<<CS02) |  (1<<CS00) ;  // prescaler 1024
	while ((TIFR0 & (1<<OCR0A)) == 0);
	TCCR0B = 0;                                              //stop timer
	TIFR0 = (1<<OCR0A);                                //reset flag

}

ISR(ADC_vect)
{
	ADCvalue = ADCH;                    // Output ADCH to ADCvalue variable
}
```

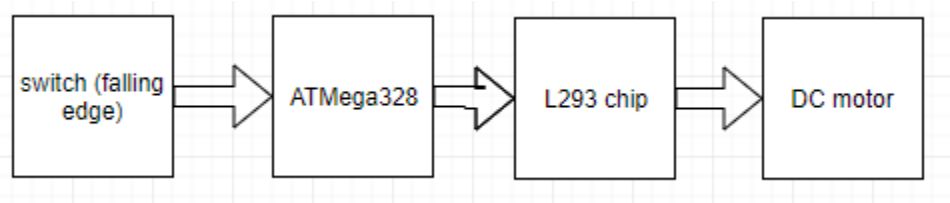
Servo Motor code:

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/interrupt.h>

int ADCvalue;


int main(void)
{
	TCCR1A |= (1<<COM1A1) | (1<<COM1B1)| (1<<WGM11);                    //TIMER1
	TCCR1B |= (1<<WGM13)| (1<<WGM12)| (1<<CS11)|(1<<CS10);        //prescaler 64
	ICR1 = 2500;          //fPWM = 50Hz, period 20ms
	DDRB= 0xFF;
	DDRC = 0x00;          //set port c as an input
	DDRD = 0xFF;          //set port d as an output
	ADMUX = 0x00;         //use ADC0 connected to the potentiometer
	ADMUX |= (1<<REFS0); //AVcc
	ADMUX |= (1<<ADLAR); //right adjust
	ADCSRA |= (1<<ADPS2) |(1<<ADPS1) |(1<<ADPS0); //prescaler of 128
	ADCSRA |= (1<<ADEN);        //ADC enable
	ADCSRA |= (1<<ADATE);              //auto trigger enable
	ADCSRB = 0;          //free mode
	ADCSRA |= (1<<ADIE);        //enable interrupt
	ADCSRA |= (1<<ADSC);
	OCR1A = 63; //0 degree, min value
	sei();
	while (1)
	{
		OCR1A = ADCvalue; //motor gets ADC value
		_delay_ms(10);

	}
}

ISR(ADC_vect)
{
	ADCvalue = ADCH;                    // Output ADCH to ADC value
```

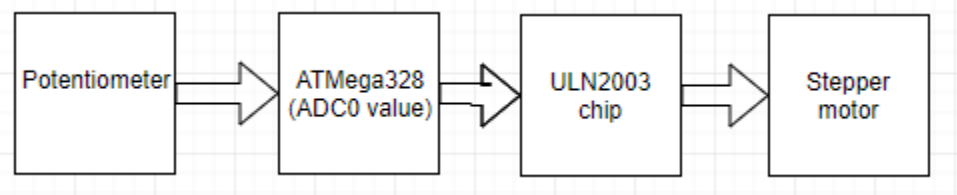
```
}
```

3. Flow chart

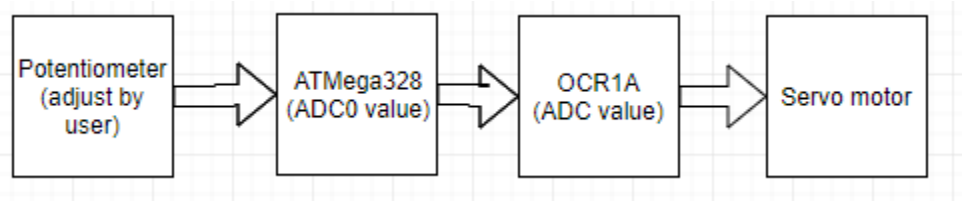
For the DC motor-



For the Stepper motor-



For the Servo motor-



4. SCREENSHOT OF EACH DEMO (BOARD SETUP)

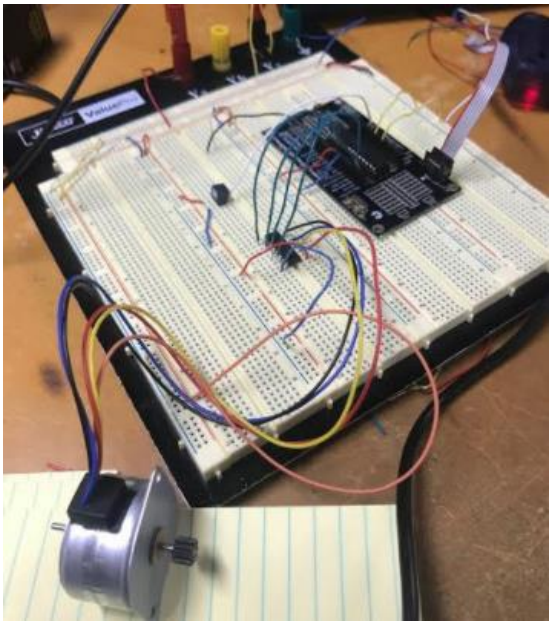


Figure 1 Set up of the stepper motor

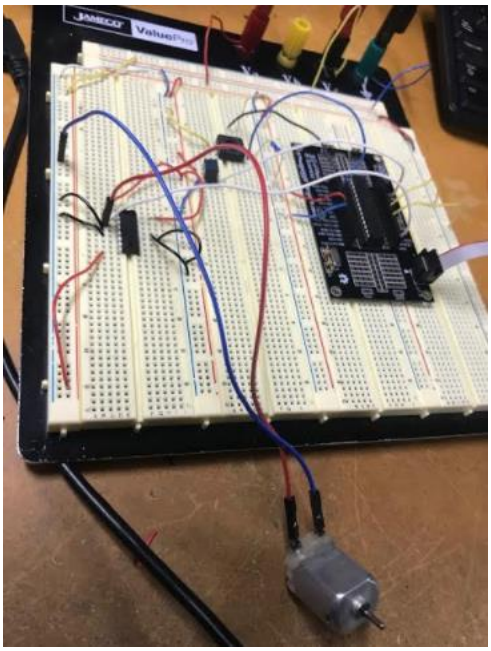


Figure 2 Set up of the DC motor

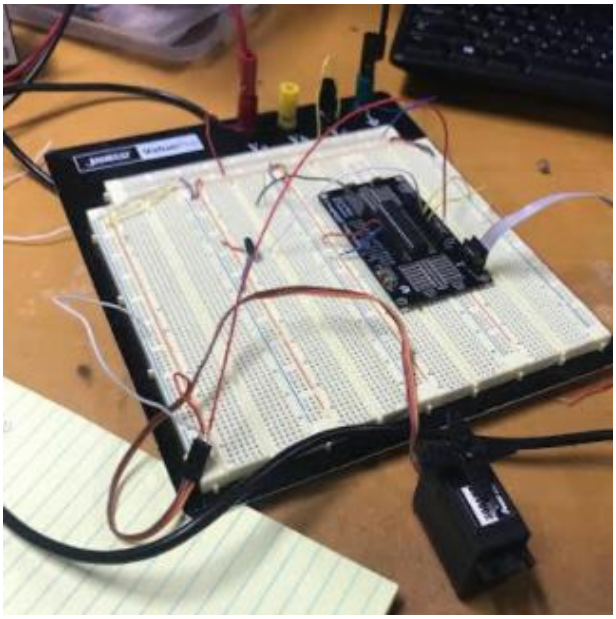


Figure 3 set up of servo motor

5. VIDEO LINKS OF EACH DEMO

DC motor- https://youtu.be/WrnR4ehkQR4
Stepper motor- https://youtu.be/2YX0L3S_kJg
Servo motor- https://youtu.be/iK1KQTjyQqI

6. GITHUB LINK OF THIS DA

git@github.com:EilatAvidan/microcon.git

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

"*This assignment submission is my own, original work*".
Eilat Avidan