CPE301 – SPRING 2019

Design Assignment 4, Part 1

Student Name: James Skelly
Student #: 2000945485
Student Email: skellj1@unlv.nevada.edu
Primary Github address: https://github.com/skellj1/submission_da
Directory: skellj1/submission_da

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.

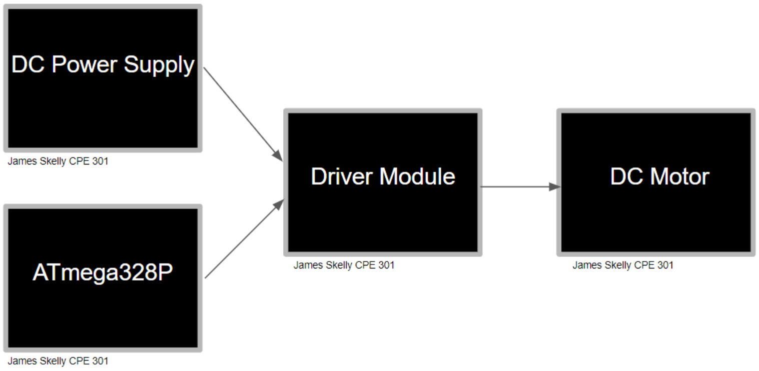
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.

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

Components used include Atmel Studio 7, Xplained Mini board with ATmega328P, DC Motor, Tektronix DC power supply, breadboard, multi-function shield, fan attachment for motor, and motor driver module.



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

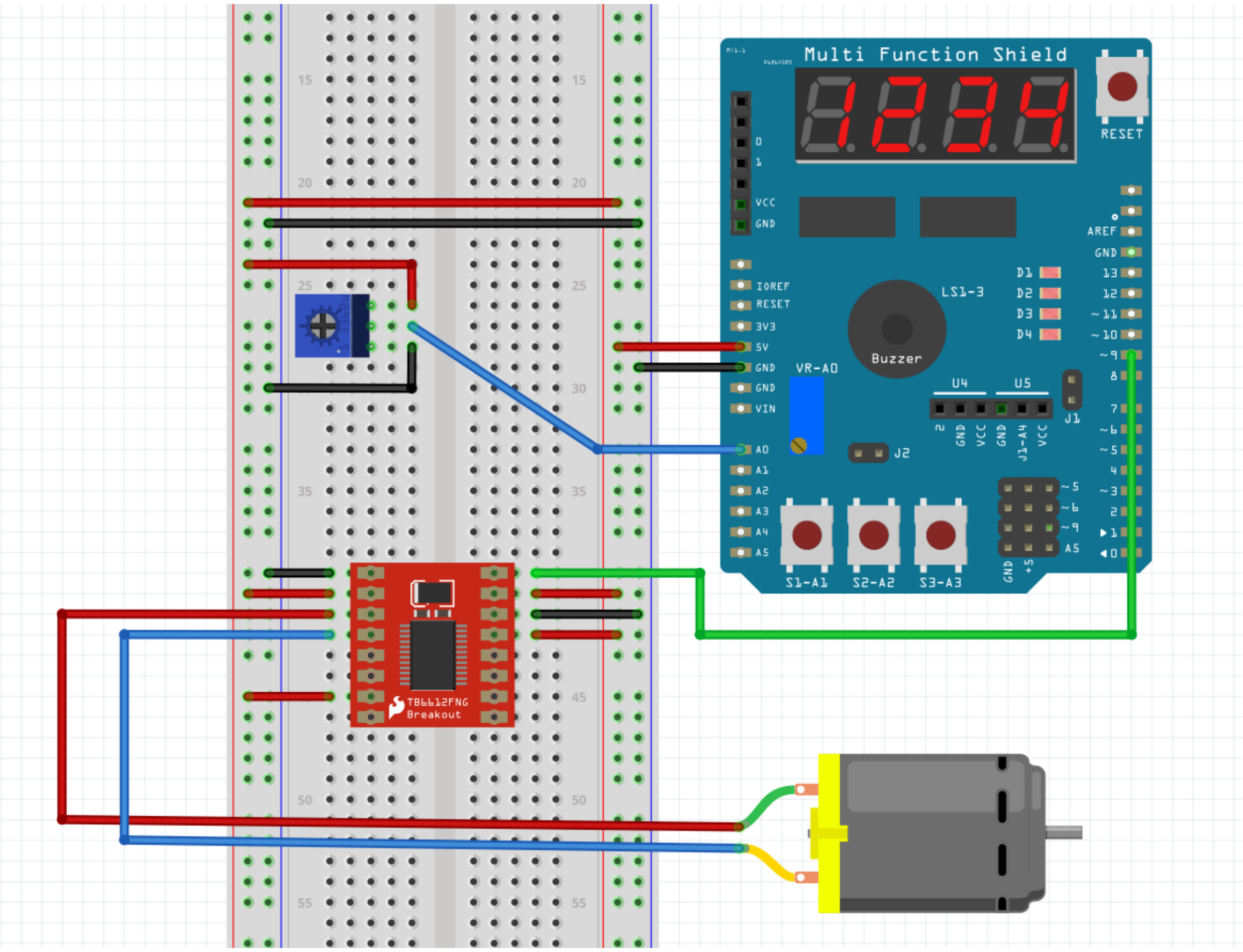
```c
/*
 * DA4A_JS.c
 * Created: 4/11/2019 10:54:25 AM
 * Author : James Skelly
 */

// define clock for delay function, include headers
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

int adcVal;                     // holds value of ADC
char toggle = 0;                // initialize toggle variable

int main(void)
{
        DDRB = 0xFF;                    // PORTB as output
        DDRC = 0x00;                    // PORTC as input
        PORTB |= 0xFF;                  // Set all of PORTB high initially

        // set timer1 operation mode and prescaler
        TCCR1A |= (1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);    // use PWM for output compare pins

        // OC1A/B, Fast PWM, Non-inverted mode
        TCCR1B |= (1<<WGM13)|(1<<WGM12)|(1<<CS11);              // timer1 pre-scaler set to 8
        ICR1 = 10000;                                           // set TOP value

        PORTC |= (1<<PORTC1);   // set portc initially high to wait for button press
        PCICR = (1<<PCIE1);     // set portc input capture enable bit
        PCMSK1 = (1<<PCINT9);   // enable pin change interrupt

        // Initialize ADC
        DIDR0 = 0x1;            // disable the digital input on ADC0 pin
        ADMUX = (1<<REFS0);     // set ADC reference pin to PC0 (Potentiometer)
        ADCSRA |= (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);// enable ADC using system clock
        ADCSRB = 0x0;           // free running mode

        sei();          // globally enable interrupts

        while(1);       // wait for pin change interrupt request
}

ISR(PCINT1_vect){

        if(!(PINC & (1 << PINC2)))                              // if button is pressed...
        {
                _delay_ms(200);                                 // debounce button pressed
                while(!(PINC & (1 << PINC2)));
                if(toggle == 1){
                        PORTB |= (1 << PORTB1);         // set PB1 high
                        PORTB &= ~(1 << PORTB5);        // turn on LED D1
                        ADCSRA |= (1 << ADSC);          // start conversion
                        while((ADCSRA&(1<<ADIF))==0);   // wait for conversion to finish
                        adcVal = ADC & 0x03FF;          // extract right 10-bits of ADC register
                        OCR1A = 10*adcVal;              // OCR1A value for duty cycle
                }
                else if(toggle == 0){
                        OCR1A = 0;                      // set output compare value to 0
                        PORTB &= ~(1 << PORTB1);        // set PB1 low
                        PORTB |= (1 << PORTB5);         // turn of LED D1
                }

                toggle ^= 1;                            // toggle the variable used to toggle
        }
}
```

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

Not applicable for this DA.

4. SCHEMATICS

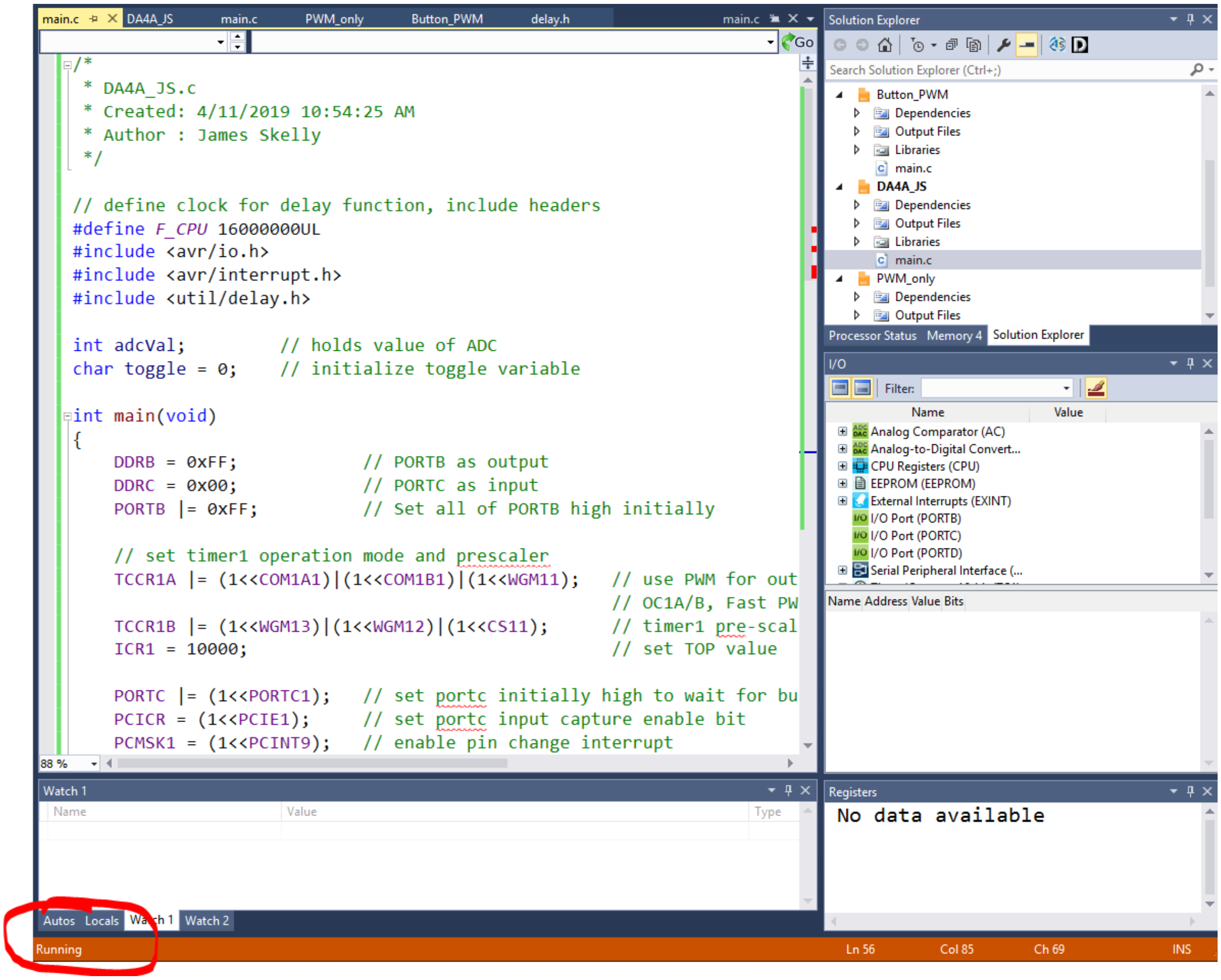
The schematic below was created using Fritzing. The Multi Function Shield PWM pin 9 is connected to the Xplained Mini at pin PB1 of the microcontroller. The signal on this pin is the pulse width modulated output signal generated by timer1 in PWM mode. The driver module utilizes the TB6612FNG motor driver IC to drive the DC motor. The blue potentiometer has 5V connected across it and the value read out is connected to A0 on the shield, which is PC0 on the microcontroller. This pin is used as the input to the Analog-to-Digital converter, which converts the analog voltage signal output of the potentiometer into a 10-bit binary (digital) value.



The driver does not provide any more voltage to the motor, but instead, provides more current to drive the motor and overcome the internal resistance and rotational inertia. A power supply was used to power the driver.

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

There was no real output to show on Atmel Studio in this assignment. All output is to be seen on the oscilloscope (PWM DC) and the motor itself.

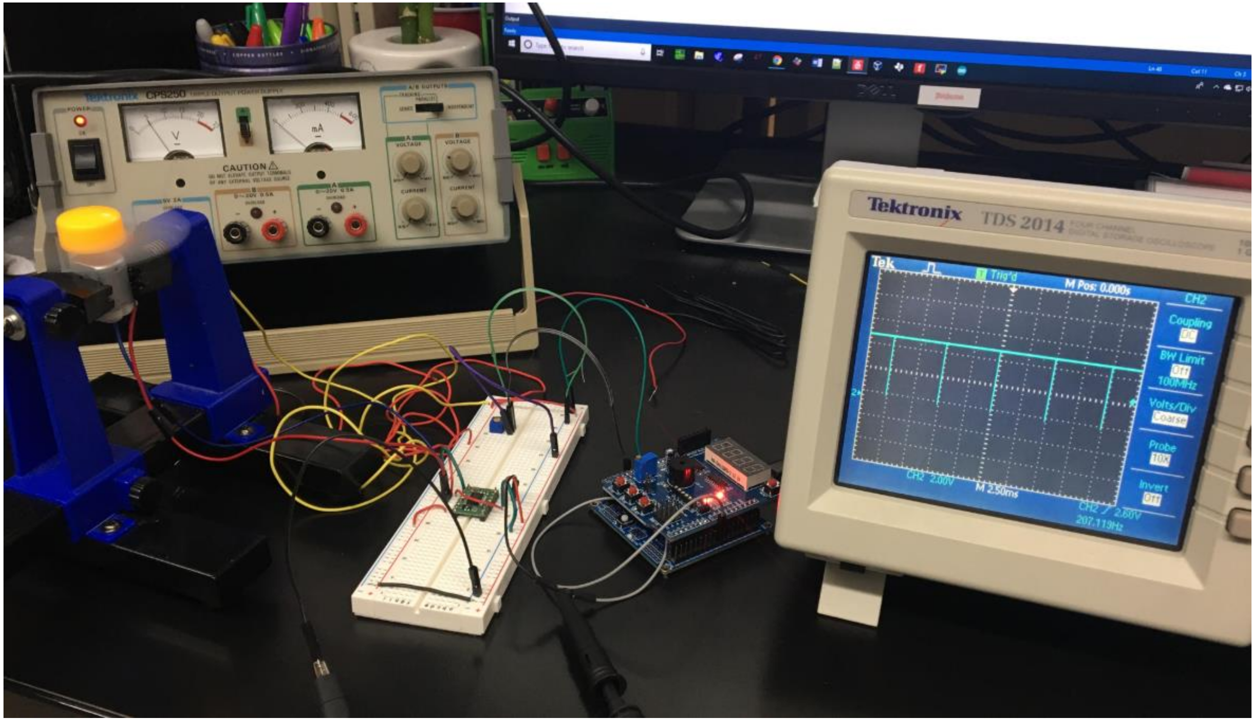


```
/*
 * DA4A_JS.c
 * Created: 4/11/2019 10:54:25 AM
 * Author : James Skelly
 */

// define clock for delay function, include headers
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

int adcVal;          // holds value of ADC
char toggle = 0;     // initialize toggle variable

int main(void)
{
    DDRB = 0xFF;                // PORTB as output
    DDRC = 0x00;                // PORTC as input
    PORTB |= 0xFF;              // Set all of PORTB high initially

    // set timer1 operation mode and prescaler
    TCCR1A |= (1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);   // use PWM for out
                                                    // OC1A/B, Fast PW
    TCCR1B |= (1<<WGM13)|(1<<WGM12)|(1<<CS11);      // timer1 pre-scal
    ICR1 = 10000;                                   // set TOP value

    PORTC |= (1<<PORTC1);   // set portc initially high to wait for bu
    PCICR = (1<<PCIE1);     // set portc input capture enable bit
    PCMSK1 = (1<<PCINT9);   // enable pin change interrupt
```

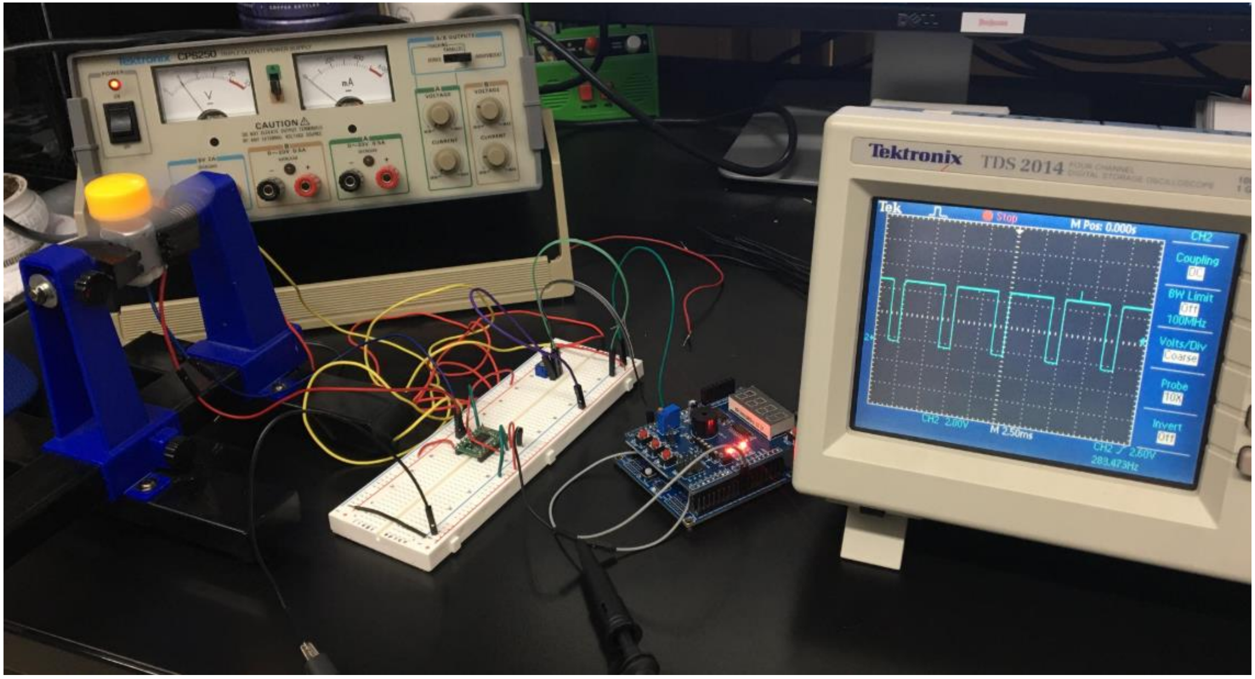
No data available

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

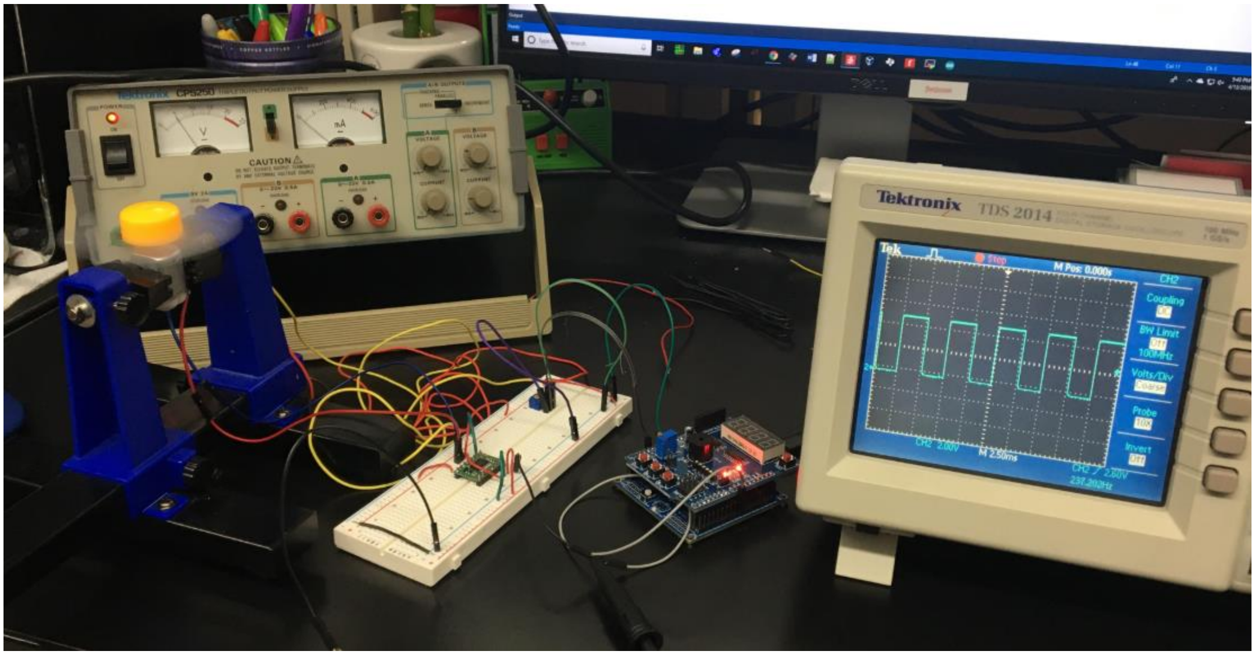
95% Duty Cycle



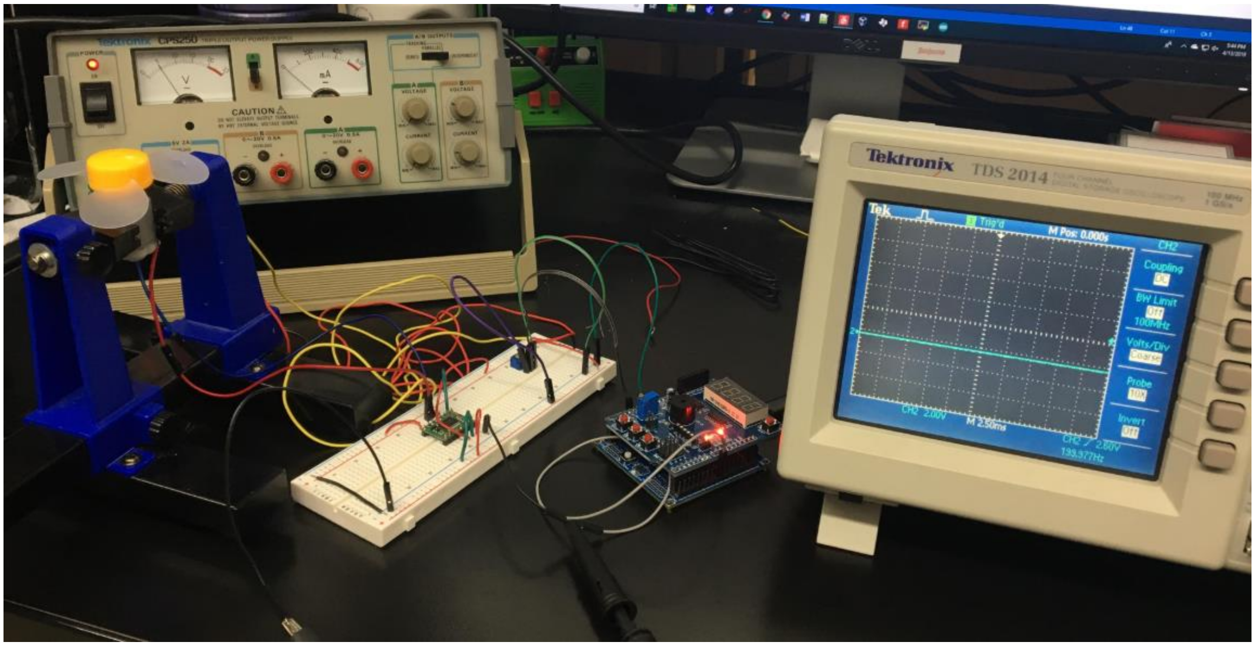
80% Duty Cycle



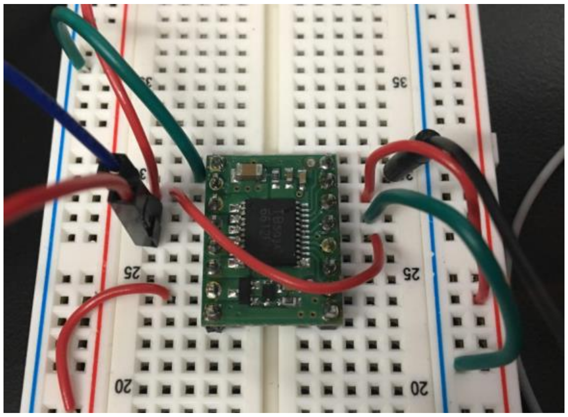
50% Duty Cycle



0% Duty Cycle



Power Supply and Driver Module



7. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=m1eQXuU82uc

8. GITHUB LINK OF THIS DA

https://github.com/skellj1/submission_da

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

"This assignment submission is my own, original work".
James W. Skelly