Bryan Takemoto

CPE301 – SPRING 2018

Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

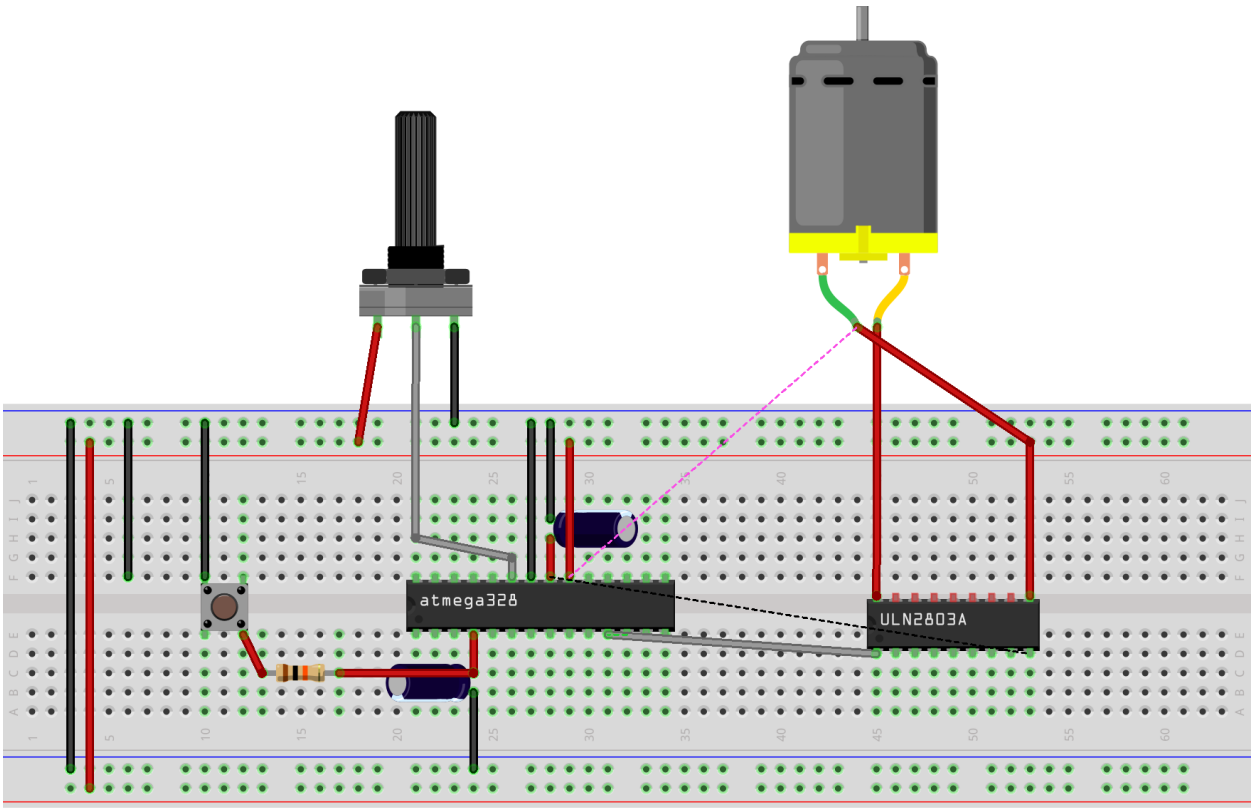
The student understands that all required components should be submitted in complete for grading of this assignment.

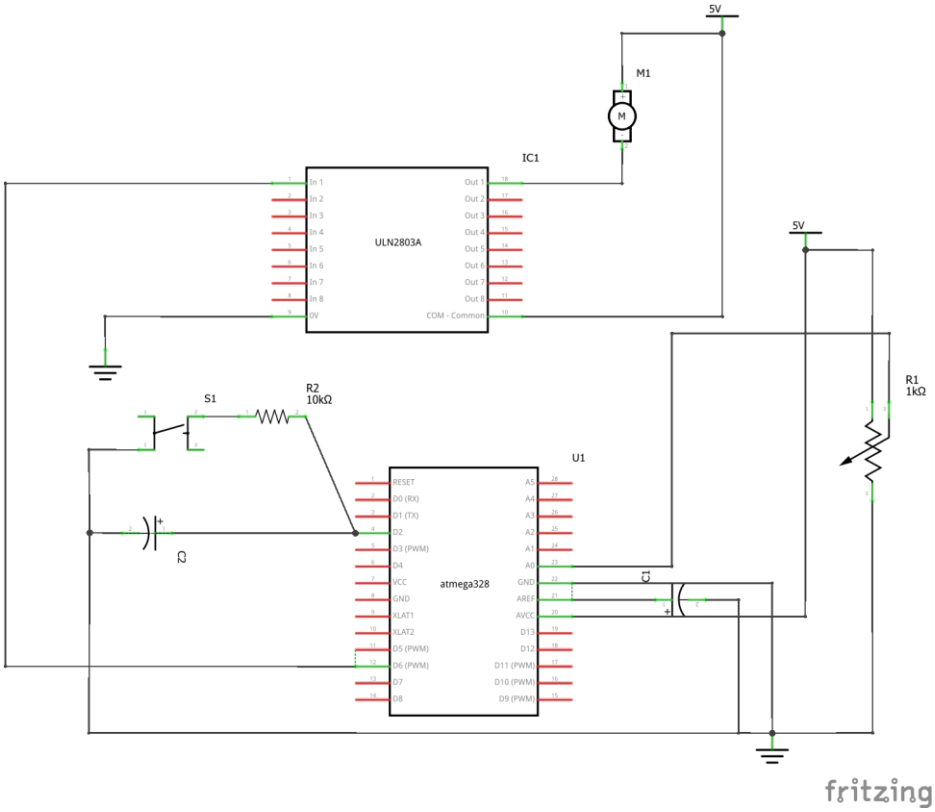
NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST		
2.	TASK 1 [DC MOTOR] SCHEMATIC AND BREADBOARD		
3.	TASK 1 [DC MOTOR] C CODE		
4.	TASK 1 [DC MOTOR] FLOW CHART		
5.	TASK 2 [STEPPER MOTOR] SCHEMATIC AND BREADBOARD		
6.	TASK 2 [STEPPER MOTOR] C CODE		
7.	TASK 2 [STEPPER MOTOR] FLOW CHART		
8.	TASK 3 [SERVO MOTOR] SCHEMATIC AND BREADBOARD		
9.	TASK 3 [SERVO MOTOR] C CODE		
10.	TASK 3 [SERVO MOTOR] FLOW CHART		
11.	VIDEO LINKS TO ALL TASKS		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components used:
Switch
Capacitors and Resistors
DC Motor
Stepper Motor
Servo Motor
Potentiometer

2. TASK 1 [DC MOTOR] SCHEMATIC AND BREADBOARD

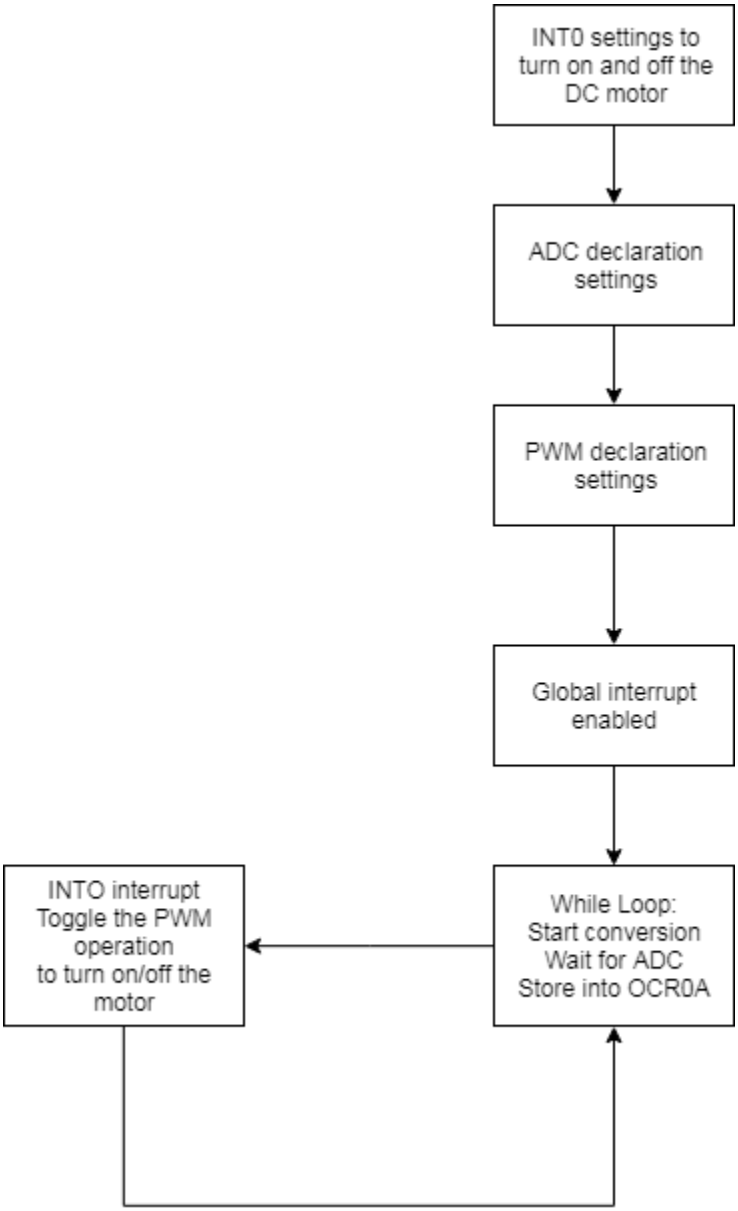

atmega328
ULN2803A
fritzing


5V
M1
IC1
ULN2803A
5V
R1
1kΩ
S1
R2
10kΩ
U1
atmega328
fritzing

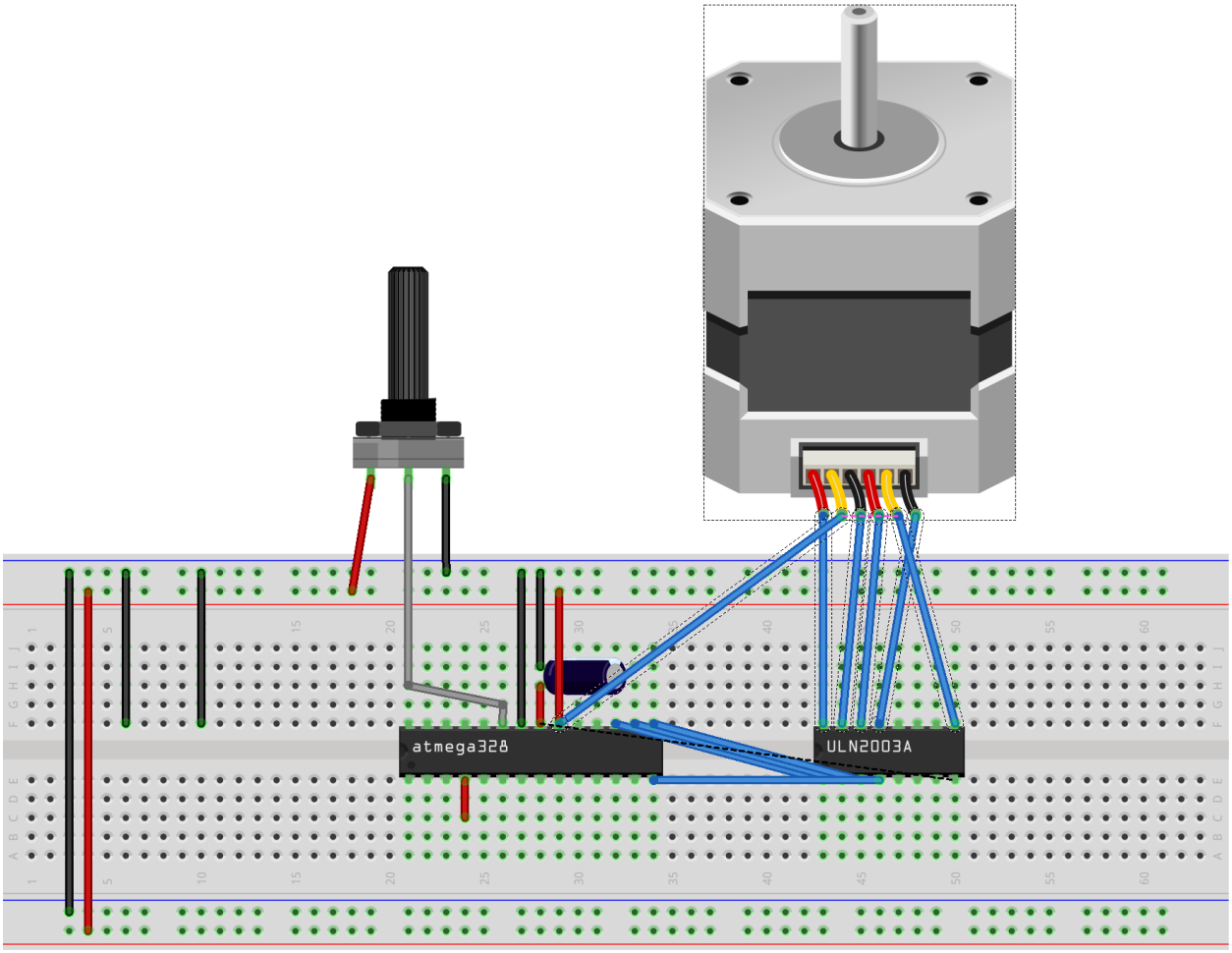
3. TASK 1 [DC MOTOR] C CODE

```
#include <avr/io.h>
#include <stdint.h>
#include <avr/interrupt.h>

int main(void)
{
	volatile uint16_t ADCvalue;
	DDRD |= (1 << 6);                                   // PD.6 is output [PWM]
	PORTD |= (1 << 2);                                  // Pull up the resistor for INT0
	EIMSK |= (1 << INT0);                               // Enable interrupt for INT0
	EICRA |= 0x2;                                       // Trigger on falling edge

	// ADC settings
	ADMUX |= (1 << REFS0);      // AVcc with external capacitor at AREF pin
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);  // Set prescalar to 128
	ADCSRA |= (1 << ADEN);      // Enable ADC

	// PWM Settings
	DDRD = (1 << 6);                                    // Set OC0A as output (PD6)
	TCCR0A |= (1 << COM0A1) | (1 << COM0A0);            // Inverting mode
	TCCR0A |= (1 << WGM01) | (1 << WGM00);              // Fast PWM mode
	TCCR0B |= (1 << CS02) | (1 << CS00);                // Fosc/1024

	sei();

    while (1)
    {
		ADCSRA |= (1 << ADSC);                      // Start conversion
		while((ADCSRA & (1 << ADIF)) == 0);         // Wait for conversion
		ADCvalue = ADC >> 2;
		ADCvalue = 0xFF - ADCvalue;
		OCR0A = ADCvalue;                           // Adjust motor speed

    }
	return 0;
}

// Switch to turn on and off the motor
ISR(INT0_vect)
{
	TCCR0A ^= (1 << COM0A1) | (1 << COM0A0);  // Turn off motor
	EIFR = (1 << INTF0);                      // Clear interrupt flag
}
```

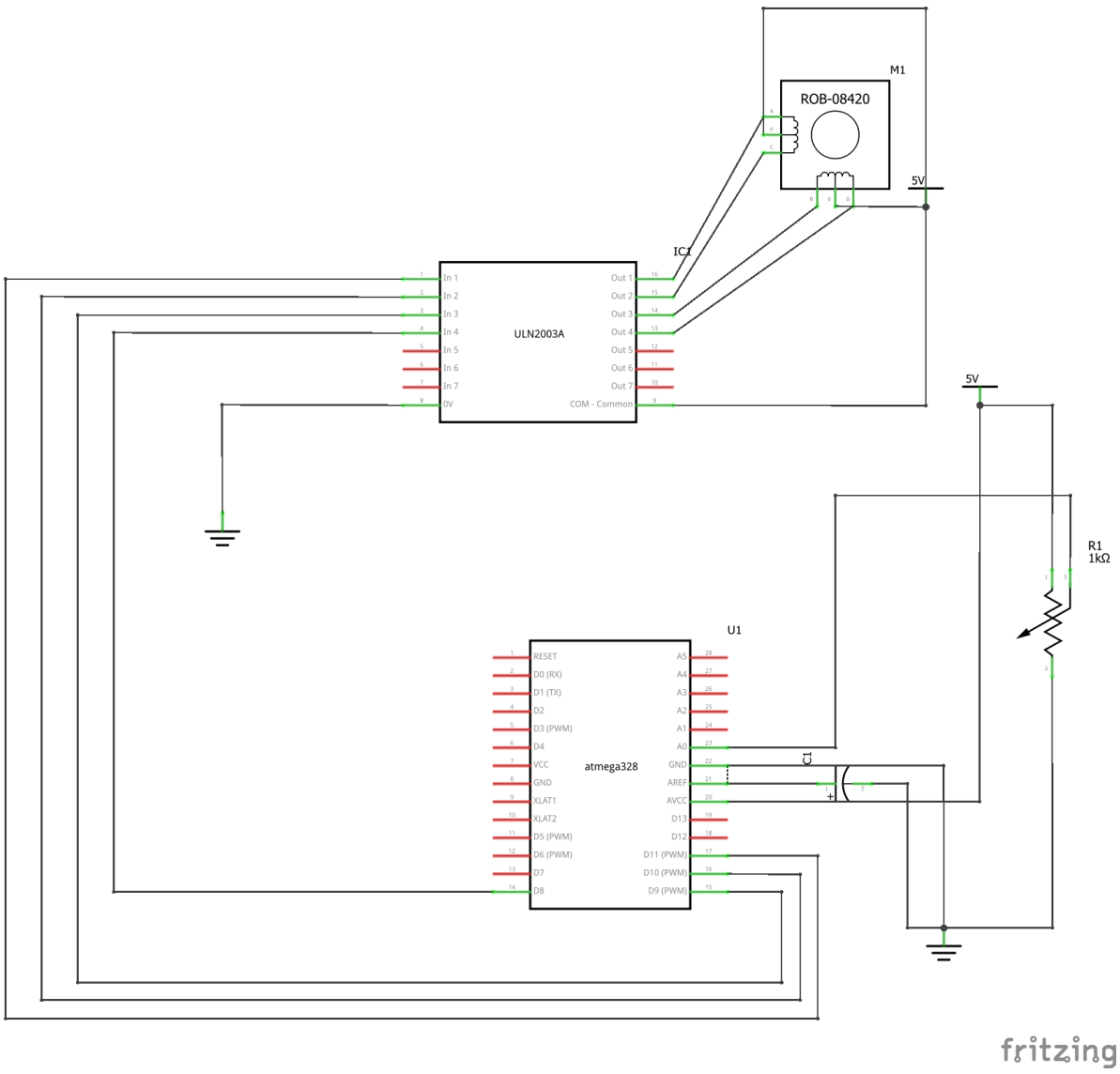
4. TASK 1 [DC MOTOR] FLOW CHART



5. TASK 2 [STEPPER MOTOR] SCHEMATIC AND BREADBOARD



fritzing

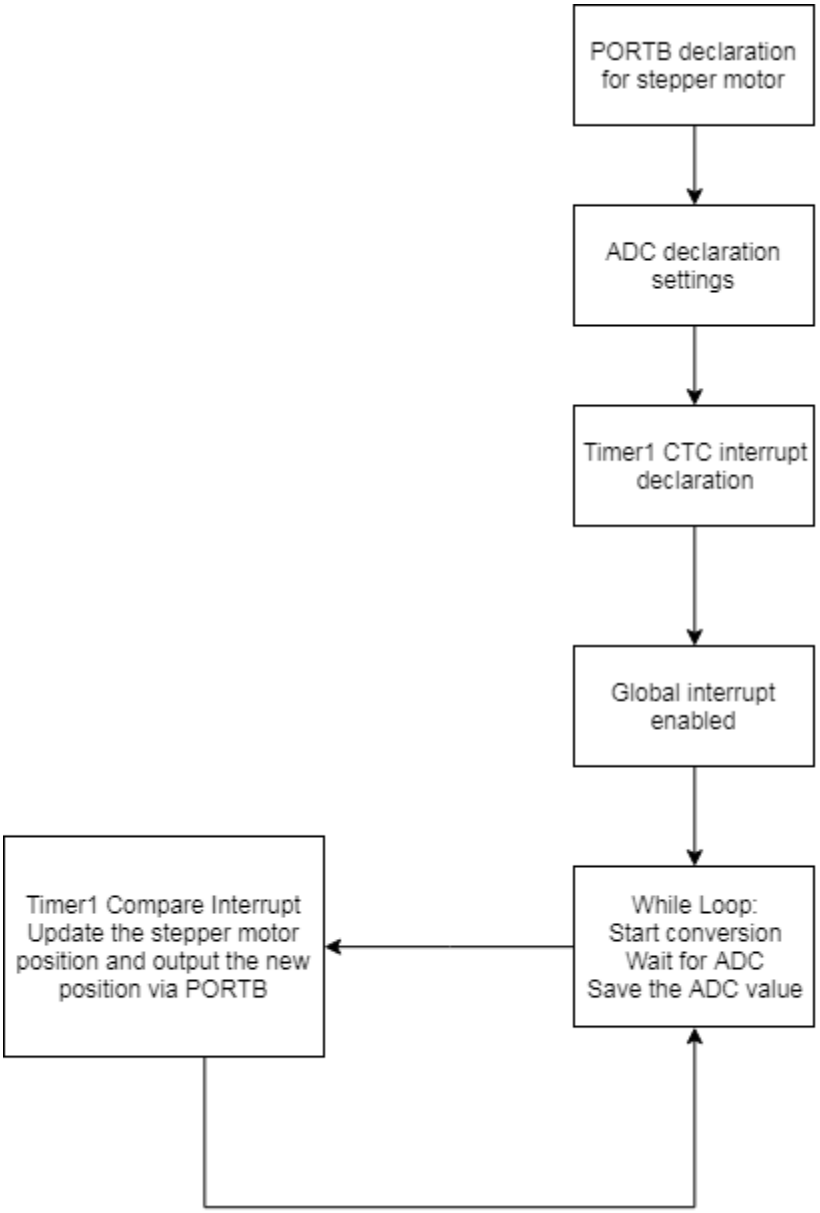


6. TASK 2 [STEPPER MOTOR] C CODE

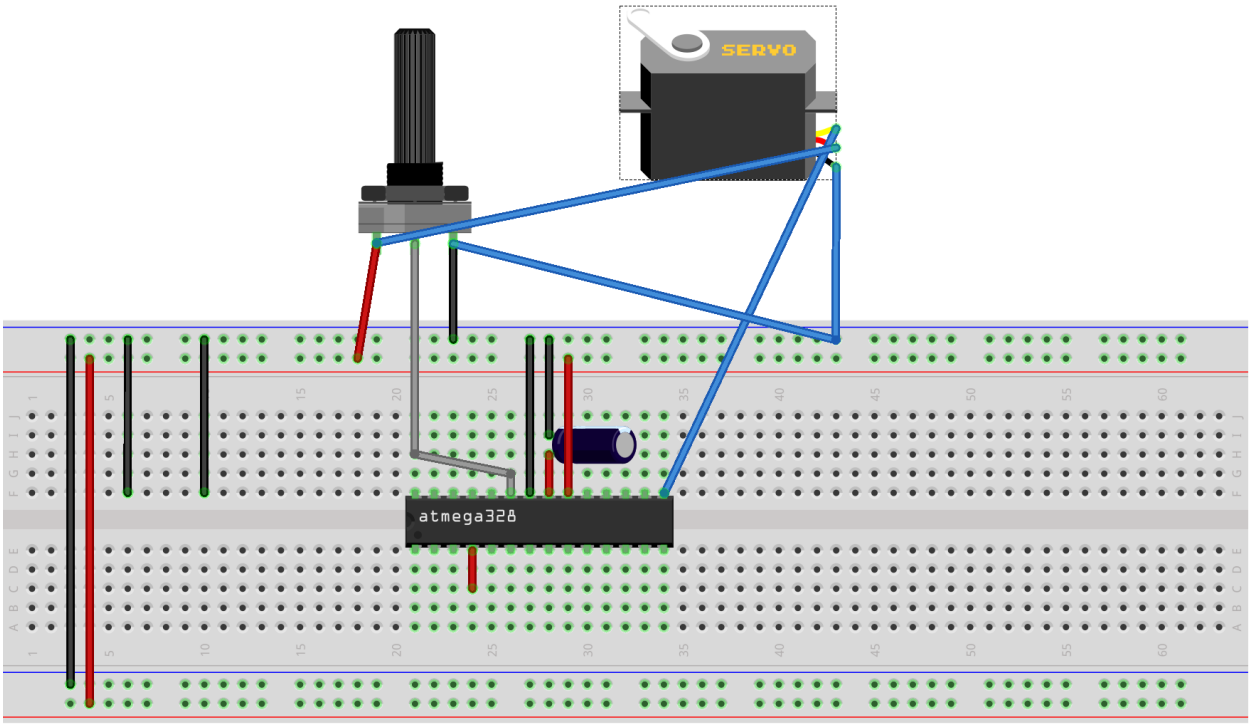
```
#include <avr/io.h>
#include <stdint.h>
#include <avr/interrupt.h>

volatile uint8_t stepperPos;
volatile uint16_t ADCvalue;

int main(void)
{
	// Port initialization
	DDRB |= 0x0F;
	PORTB = 0x00;

	// ADC settings
	ADMUX |= (1 << REFS0);
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
```

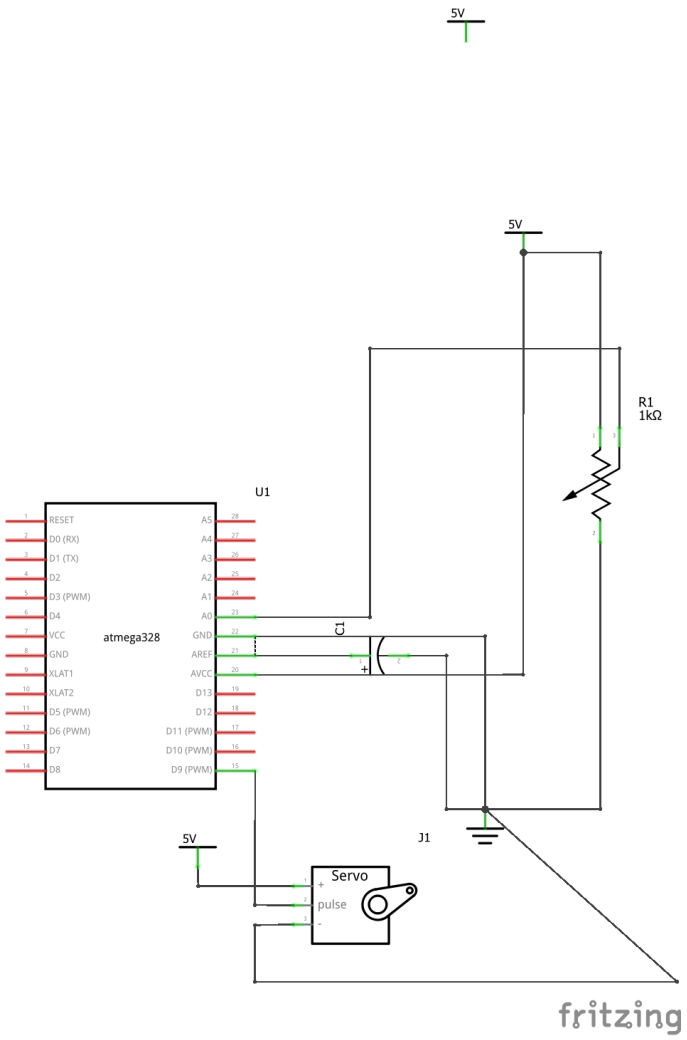
```
        ADCSRA |= (1 << ADEN);

        // Timer1 settings
        TCCR1B |= (1 << WGM12);           // CTC mode
        TCCR1B |= (1 << CS11);            // Prescalar of 8
        TIMSK1 |= (1 << OCIE1A);          // CTC interrupt
        TCNT1 = 0x00;
        OCR1A = 65535;

        // Initialize stepperPos
        stepperPos = 0x06;

        sei();
    while (1)
    {
            ADCSRA |= (1 << ADSC);                          // Start conversion
            while((ADCSRA & (1 << ADIF)) == 0);             // Wait for conversion
            ADCvalue = ADC << 5;
    }
        return 0;
}

// CTC ISR
ISR(TIMER1_COMPA_vect)
{
        OCR1A = 65535 - ADCvalue;         // Speed of the stepper
        if(ADCvalue > 32)
        {
            switch(stepperPos)            // Set next state of stepper
            {
                case 0x06:
                    stepperPos = 0x0C;
                    break;
                case 0x0C:
                    stepperPos = 0x09;
                    break;
                case 0x09:
                    stepperPos = 0x03;
                    break;
                case 0x03:
                    stepperPos = 0x06;
                    break;
                default:
                    break;
            }
            PORTB = stepperPos;
        }
        else
            PORTB = 0x00;
        TIFR1 |= (1 << OCF1A);      // Clear flag
}
```

7. TASK 2 [STEPPER MOTOR] FLOW CHART



8. TASK 3 [SERVO MOTOR] SCHEMATIC AND BREADBOARD


SERVO
atmega328
fritzing

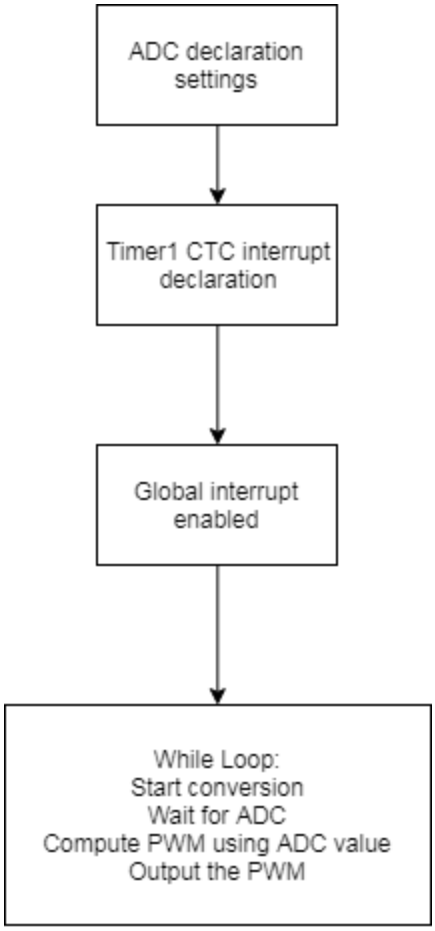


9. TASK 3 [SERVO MOTOR] C CODE

```
#include <avr/io.h>
#include <stdint.h>

int main(void)
{
	// ADC settings
	ADMUX |= (1 << REFS0);			// AVcc with external capacitor at AREF pin
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);  // Set prescalar to 128
	ADCSRA |= (1 << ADEN);                                 // Enable ADC

	// PWM Settings
	DDRB = (1 << 1);                                // Set OC1A as output (PB1)
	ICR1 = 5000;                                    // Set top
	TCCR1A |= (1 << COM1A1) | (1 << COM1B1);        // Set OC1A high on compare
	TCCR1A |= (1 << WGM11);
	TCCR1B |= (1 << WGM12) | (1 << WGM13);          // Fast PWM Mode
	TCCR1B |= (1 << CS11) | (1 << CS10);            // Clocked divided by 64

    while (1)
```

```
    {
            ADCSRA |= (1 << ADSC);                  // Start ADC conversion
            while((ADCSRA & (1 << ADIF)) == 0);     // Wait for conversion
            OCR1A = (ADC/3) + 200;                  // Assign ADC value to PWM
        }
}
```

10. TASK 3 [SERVO MOTOR] FLOW CHART



11. VIDEO LINKS TO ALL TASKS

Task 1: https://youtu.be/eb3BTntd4Lc
Task 2: https://youtu.be/4yAEPqhDWnU
Task 3: https://youtu.be/4R--y87nlpA

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

"*This assignment submission is my own, original work*".
Bryan Takemoto