CPE301 – SPRING 2019
Design Assignment 4B

Student Name: Yannick Kengne Tatcha
Student #: 5003294512
Student Email: kengneta@unlv.nevada.edu
Primary Github address: **Vasty1995/submission_da**
Directory: DA4B

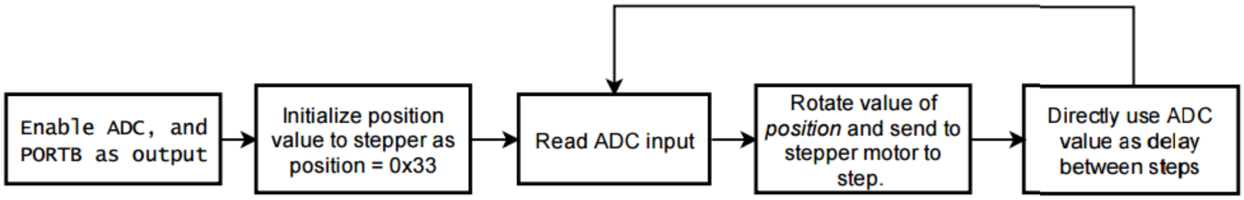
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

- Atmega328P
- Servo motor
- Stepper motor
- PC
- 5V power supply

2. DEVELOPED CODE

1- TASK 1 in AVR C

Flow chart

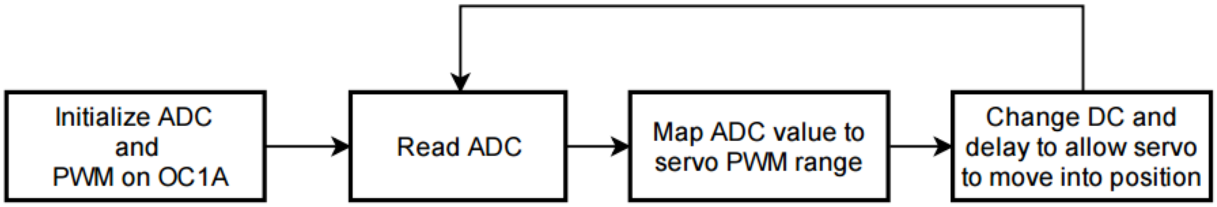

Enable ADC, and PORTB as output
Initialize position value to stepper as position = 0x33
Read ADC input
Rotate value of position and send to stepper motor to step.
Directly use ADC value as delay between steps

```
/*
 * DA4BT1.c
 *
 * Created: 4/19/2019 1:07:39 PM
 * Author : YKengne
 */

#define F_CPU 8000000UL // XTAL = 8MHZ

#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>

#define BAUDRATE     9600
#define ASYNCH_NORM_PRESCALER (F_CPU/16/BAUDRATE - 1)

void ADC0init();                                 // Initialize ADC0 input
```

```
unsigned short readADC();                 // read ADC0 analog input and return it
void delay_ms(unsigned int);              // shell procedule to call _delay_ms on variable
input
void step_clockwise(unsigned int, unsigned int);// step stepper motor desired number of
times with delay
unsigned char rotateLeft(unsigned char); // rotate bits of input to the left.
int USART0_sendChar(char, FILE*);  // Write character to USART0
void usart0_init (void);                  // Initialize USART0

FILE USART0_stream = FDEV_SETUP_STREAM(USART0_sendChar, NULL, _FDEV_SETUP_WRITE);

// Current position signal of stepper motor
unsigned char positionSig = 0x33;
int main()
{
       unsigned short adcVal;
       DDRB = 0xFF; // make portB output pins.

       stdout = &USART0_stream;    // change standard output to point to a USART stream

       usart0_init();                            // Initialize USART0 for debugging and
monitoring
       ADC0init();                               // Initialize ADC0 input

       while (1)
       {
              adcVal = readADC();                // read ADC0;
              step_clockwise(1, adcVal);  // Step stepper motor 1 step with an adcVal
delay
              printf("ADC Value: %u | Position signal: 0x%X\n", adcVal, positionSig); //
print monitoring message
       }
}
void usart0_init (void)
/*
 * Procedure to initialize USART0 asynchronous with enabled RX/TX, 8 bit data,
 * no parity, and 1 stop bit.
*/
{
       UCSR0B = (1<<TXEN0)  | (1<<RXEN0);// enable transmit/receive
       UCSR0C = (1<<UCSZ01) | (1<<UCSZ00);       // asynchronous, 8N1
       UBRR0L = ASYNCH_NORM_PRESCALER;           // Set prescaler based on desired
baudrate
}
int USART0_sendChar(char data, FILE *stream)
/*
 * Procedure to send a single character over USART0. If character is linefeed, reset
 * line.
 * Assumes ASCII code.
 */
{
       if(data == '\n')     // If character is linefeed,
       {                                  // First send return.
              while(! (UCSR0A & (1<<UDRE0)) );
              UDR0 = '\r';
       }
       while(! (UCSR0A & (1<<UDRE0)) ); // Wait for last data to be transmitted.
       UDR0 = data;  // send data
       return 0;
}
unsigned char rotateLeft(unsigned char x)
/*
 * Given an unsigned character x, rotateLeft will do a logic rotatation of
 * the bits of x to the right.
 */
{
       unsigned char shiftIn = 0;
       if ((x & 0x80) == 0x80)
              shiftIn = 0x01;
       return ((x<<1) | shiftIn);
}
void step_clockwise(unsigned int steps, unsigned int delay)
/*
 * Given the unsigned integers steps, and delay, step_clockwise will send the appropriate
 * signal to PORTB[7:0] to step a stepper motor in the clockwise direction.
 * A global variable positionSig must be initialized to 0x33.
 */
{
       for (; steps > 0; steps--)  // loop steps times.
       {
              positionSig = rotateLeft(positionSig);   // Rotate value of positionSig
              PORTB = positionSig;                            // send data to PORTB
              delay_ms(delay);                                       // Delay a given
value of milliseconds.
       }
}
unsigned short readADC()
// readADC will read the adcValue after it has been calculated.
{
       ADCSRA |= (1<<ADSC);                     // Begin conversion
       while((ADCSRA & (1<<ADIF)) == 0 );// Wait for conversion to finish.
       return ADC;
}
void ADC0init()
// ADC0init will initialize analog input on ADC0, set voltage reference to Vcc, with
// data right justified on data register.
{
       DDRC   &= ~(0<<DDC0);
       ADCSRA = 0x87;                    // Make ADC enable and select ck/128
       ADMUX  = (1<<REFS0); // VCC reference, ADC0 single ended input
       // data will be right-justified
}
void delay_ms(unsigned int count)
/*
 * Procedure to perform a delay based on an unsigned short
 * since the _delay_ms macro will not accept parameters
 * other than constant values.
 */
{
       int i;
       for(i = 0; i < count; i++)
              _delay_ms(1);
}
```

2- TASK 2 in AVR C

Flow chart of task 2

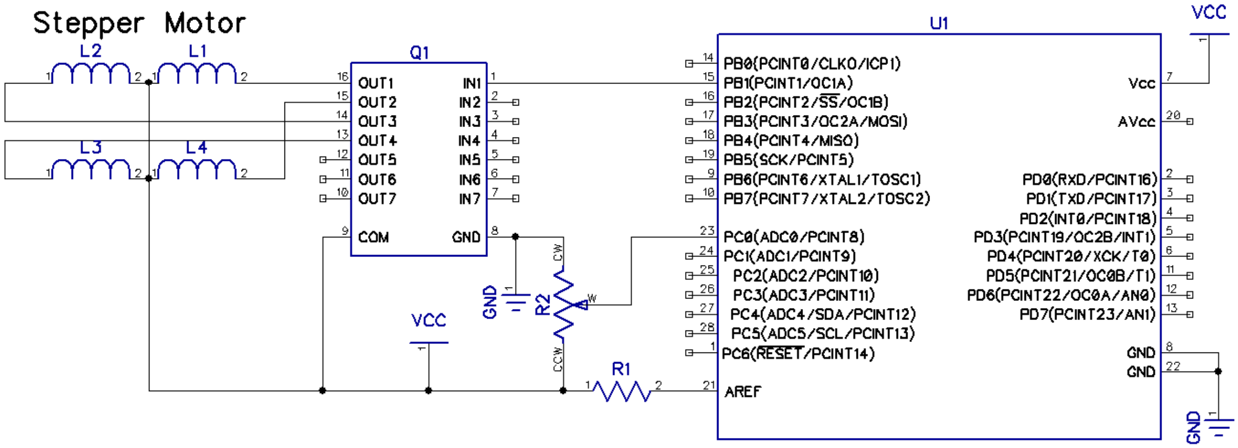

Initialize ADC and PWM on OC1A
Read ADC
Map ADC value to servo PWM range
Change DC and delay to allow servo to move into position

```
/*
 * DA4BT2.c
 *
 * Created: 4/19/2019 1:10:50 PM
 * Author : YKengne
 */

#define F_CPU 8000000UL // XTAL = 8MHZ

#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>

#define SERVO_MIN    65
#define SERVO_MAX    285
#define BAUDRATE     9600
#define ASYNCH_NORM_PRESCALER (F_CPU/16/BAUDRATE - 1)

void ADC0init();                                 // Initialize ADC0 input
unsigned short readADC();                 // read ADC0 analog input and return it
void delay_ms(unsigned int);              // shell procedule to call _delay_ms on variable
input
void PWM_OC1A_init();                            // Initialize PWM on OC1A at 50Hz
int USART0_sendChar(char, FILE*);  // Send character on USART0
void usart0_init (void);                  // Initialize USART0

FILE USART0_stream = FDEV_SETUP_STREAM(USART0_sendChar, NULL, _FDEV_SETUP_WRITE);

int main()
{
       unsigned short adcVal;      // Variable to store input ADC Value
       unsigned short newVal;      // new value calculated based on a range for servo
       DDRB = 0xFF; // make portB output pins.

       ADC0init();                 // Initialize ADC0 input
       PWM_OC1A_init();     // initialize pwm  on OC1A
       usart0_init();              // Initialize USART0 for debugging and monitoring

       stdout = & USART0_stream;   // change standard output to point to a USART stream

       while (1)
       {
              adcVal = readADC();               // read ADC0;
              // Map ADC value to a range from 0 to SERVO_MAX
              newVal = (unsigned short)((float)adcVal / ((1UL<<10) - 1) * SERVO_MAX);
              printf("adcVal = %u\n", adcVal);   // Print monitoring data
              printf("\tnewVal = %u\n", newVal);
              if (newVal <= SERVO_MIN)    // If newVal is less than minimum servo value (0
degrees)
                     OCR1A = SERVO_MIN;         // then set OCR1A to minimum value.
              else
              {
                     OCR1A = newVal;                   // else, update OCR1A to change
duty cycle.
              }
              _delay_ms(50);                           // Delay to allow servo to move
       }
}

int USART0_sendChar(char data, FILE *stream)
/*
 * Procedure to send a single character over USART0. If character is linefeed, reset
 * line.
 * Assumes ASCII code.
 */
{
       if(data == '\n')
       {
              while(! (UCSR0A & (1<<UDRE0)) );
              UDR0 = '\r';
       }
       while(! (UCSR0A & (1<<UDRE0)) );
       UDR0 = data;
       return 0;
}

void usart0_init (void)
/*
 * Procedure to initialize USART0 asynchronous with enabled RX/TX, 8 bit data,
 * no parity, and 1 stop bit.
*/
{
       UCSR0B = (1<<TXEN0)  | (1<<RXEN0);// enable transmit/receive
       UCSR0C = (1<<UCSZ01) | (1<<UCSZ00);      // asynchronous, 8N1
       UBRR0L = ASYNCH_NORM_PRESCALER;          // To set 9600 baud rate with 8MHz clock
}

unsigned short readADC()
/*
 * Procedure to send a single character over USART0. If character is linefeed, reset
 * line.
 * Assumes ASCII code.
 */
{
       ADCSRA |= (1<<ADSC);                     // Begin conversion
       while((ADCSRA & (1<<ADIF)) == 0 );// Wait for conversion to finish.
       return ADC;
}

void ADC0init()
```

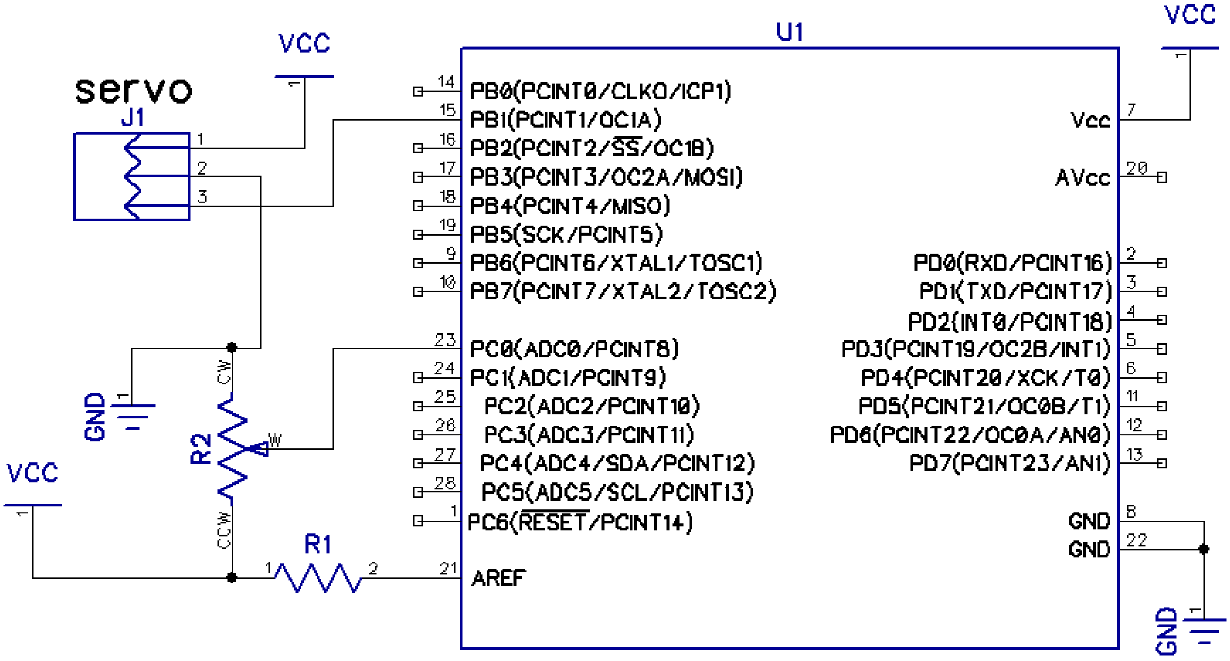
```
// ADC0init will initialize analog input on ADC0, set voltage reference to Vcc, with
// data right justified on data register.
{
       DDRC   &= ~(0<<DDC0);
       ADCSRA = 0x87;                    // Make ADC enable and select ck/128
       ADMUX  = (1<<REFS0); // VCC reference, ADC0 single ended input
       // data will be right-justified
}

void delay_ms(unsigned int count)
/*
 * Procedure to perform a delay based on an unsigned short
 * since the _delay_ms macro will not accept parameters
 * other than constant values.
 */
{
       int i;
       for(i = 0; i < count; i++)
              _delay_ms(1);
}

void PWM_OC1A_init()
{
       //Set PORTB1 pin as output
       DDRB |= (1<<DDB1);   // make OC1A as output.
       // Output compare mode on OC1A. Fast PWM with top = ICR1.
       // Clear OC1A on Compare match and set at bottom.
       TCCR1A |= 
(1<<COM1A1)|(0<<COM1A0)|(0<<COM1B1)|(0<<COM1B0)|(0<<FOC1A)|(0<<FOC1B)|(1<<WGM11)|(0<<WGM1
0);
       // Start timer with prescaler 64
       TCCR1B |= 
(0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(1<<WGM12)|(0<<CS12)|(1<<CS11)|(1<<CS10);
       ICR1 = 2499;  // F_CPU / (N * F_pwm) - 1, where N is the prescaler = 64, and F_pwm 
is the desired 50Hz frequency.
}
```

3. SCHEMATICS

Task 1

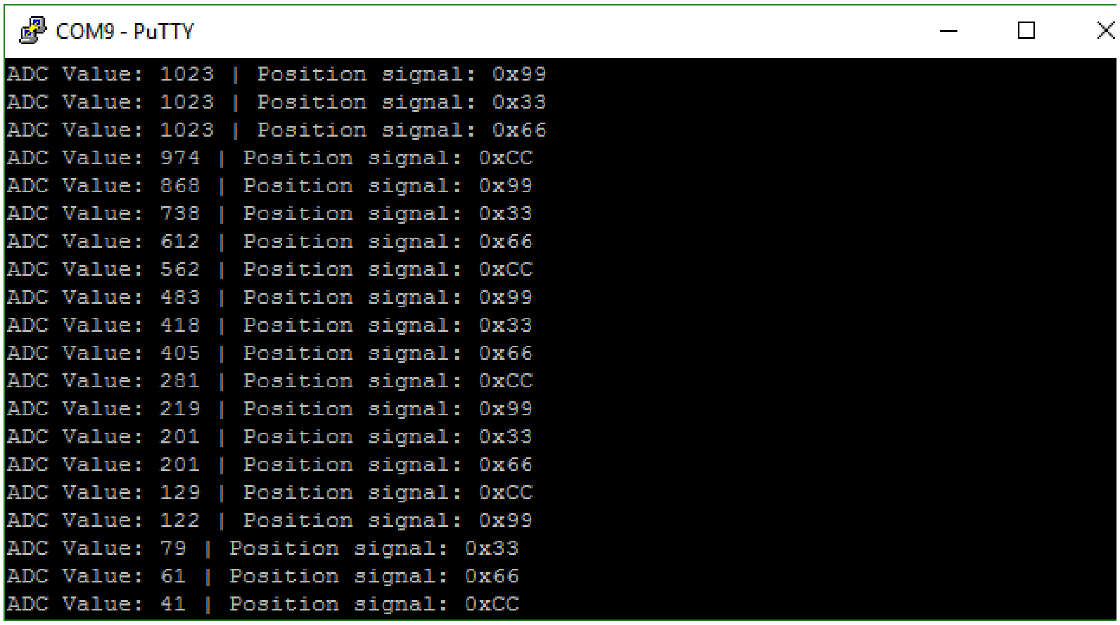


Task 2



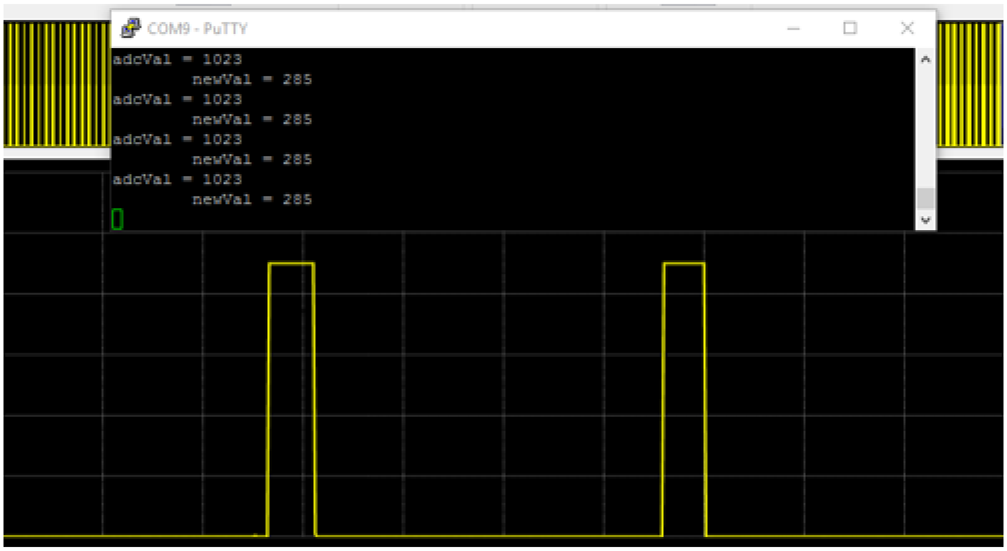
4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task1:



```
ADC Value: 1023 | Position signal: 0x99
ADC Value: 1023 | Position signal: 0x33
ADC Value: 1023 | Position signal: 0x66
ADC Value: 974 | Position signal: 0xCC
ADC Value: 868 | Position signal: 0x99
ADC Value: 738 | Position signal: 0x33
ADC Value: 612 | Position signal: 0x66
ADC Value: 562 | Position signal: 0xCC
ADC Value: 483 | Position signal: 0x99
ADC Value: 418 | Position signal: 0x33
ADC Value: 405 | Position signal: 0x66
ADC Value: 281 | Position signal: 0xCC
ADC Value: 219 | Position signal: 0x99
ADC Value: 201 | Position signal: 0x33
ADC Value: 201 | Position signal: 0x66
ADC Value: 129 | Position signal: 0xCC
ADC Value: 122 | Position signal: 0x99
ADC Value: 79 | Position signal: 0x33
ADC Value: 61 | Position signal: 0x66
ADC Value: 41 | Position signal: 0xCC
```

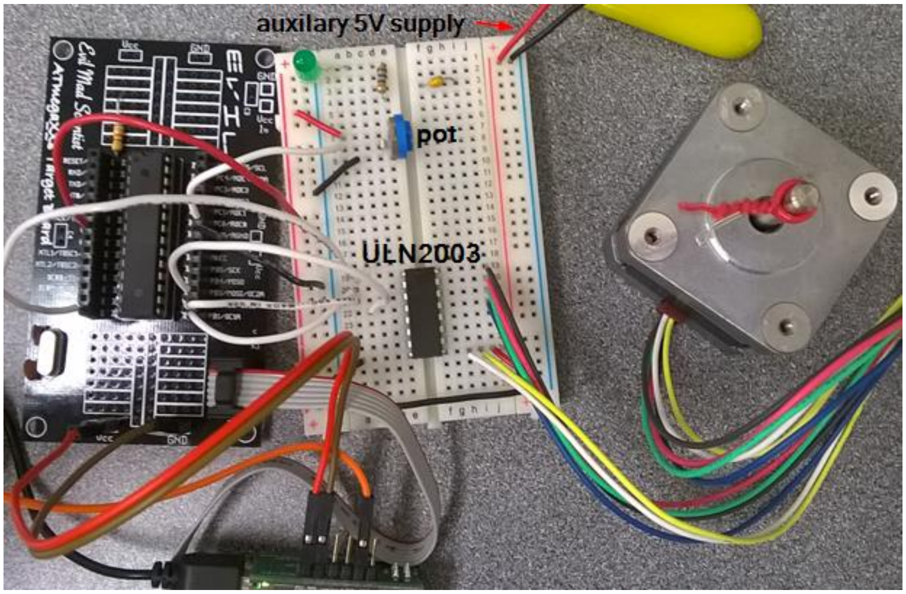
Task 2:



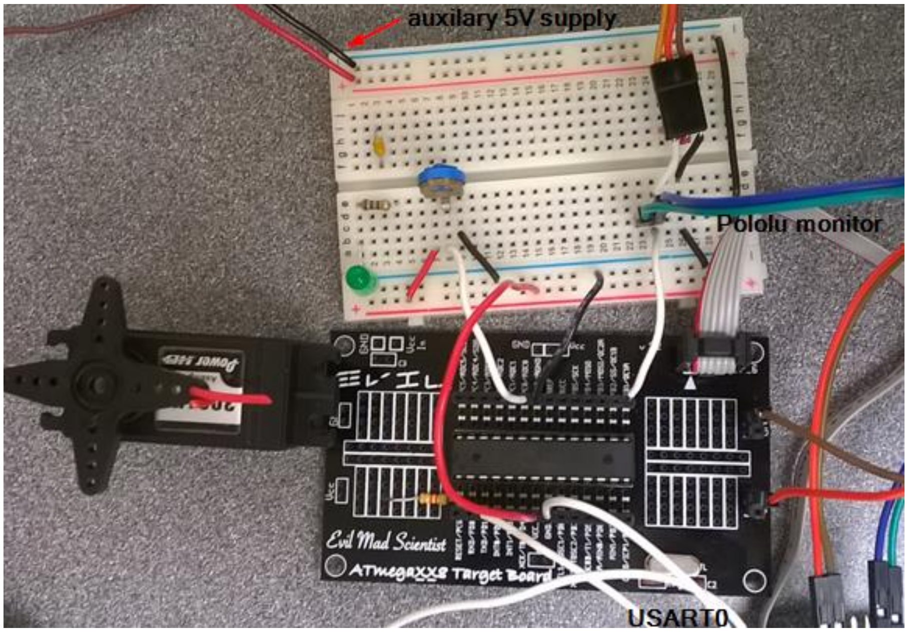
```
adcVal = 1023
          newVal = 285
adcVal = 1023
          newVal = 285
adcVal = 1023
          newVal = 285
adcVal = 1023
          newVal = 285
```

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Tak1:



Task2:



6. VIDEO LINKS OF EACH DEMO

7. GITHUB LINK OF THIS DA

https://github.com/Vasty1995/submission_da/tree/master/DA4B

Student Academic Misconduct Policy

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

"*This assignment submission is my own, original work*". Yannick Kengne Tatcha