CPE301 – SPRING 2021

Midterm II

Student Name: Brian Wolak

Student #: 2000509437

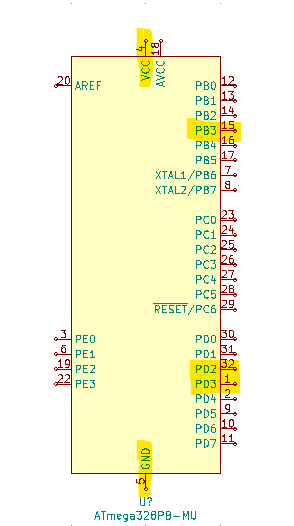
Student Email: wolak@unlv.nevada.edu

Primary Github address: <https://github.com/brianwolak/submission_da.git>

Directory: [submission\_da/MIDTERM\_2 at main · brianwolak/submission\_da (github.com)](https://github.com/brianwolak/submission_da/tree/main/MIDTERM_2)

**Task 1:**

In task one a C code is written to program the ATmega328pb to drive a DC motor in clockwise for 5 seconds and then counterclockwise for 5 seconds using a giving PWM. The code should repeat forever switching from clockwise to counterclockwise. For this experiment my geared motor has burned out and I was forced to use a regular DC motor in its place.



*ATmega328pb Ports Used in Task 1*

**Video Link:**

<https://youtu.be/uggxgGmAdAE>

**C Code:**

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdio.h>

int direction = 0;

volatile int count = 0;

//timer setup

void timer\_set(void){

//motor PWM timer

TCCR2A |= (0 << COM2A1) | (1 << COM2A0) | (1 << WGM21) | (1 << WGM20); //mode 7, OC2A toggle

TCCR2B |= (1 << WGM22) | (1 << CS22) | (1 << CS21) | (1 << CS20); //1024 prescale set

OCR2A = 0x10;

//delay timer

TCCR1A |= (0 << COM1A1) | (0 << COM1A0) | (0 << COM1B1) | (0 << COM1B0) | (0 << WGM11) | (0 << WGM10);

TCCR1B |= (0 << WGM13) | (1 << WGM12) | (1 << CS12) | (0 << CS11) | (1 << CS10);

TIMSK1 |= (1 << OCIE1A);

OCR1A = 0x3D07;

}

int main(void)

{

DDRD |= (1<<2) | (1<<3); //set PORTD output bits 2 & 3

DDRB |= (1<<3); //set PORTB output bit 3 (OC2A)

//initialize function call

timer\_set();

//turn on interrupts

sei();

while (1)

{

if (direction == 0){

PORTD &= ~(1<<2); //PINC2 low

PORTD |= (1<<3); //PINC3 high

}

if (direction == 1){

PORTD |= (1<<2); //PINC2 high

PORTD &= ~(1<<3); //PINC3 low

}

}

}

ISR(TIMER1\_COMPA\_vect){

count++; //increment 5s timer

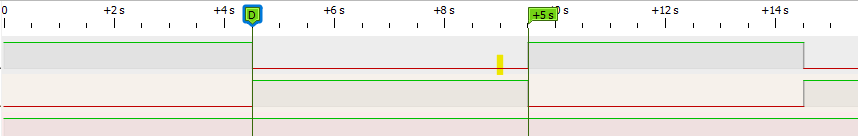
if (count == 5){ //check if 5s timer

direction ^= 1; //flip direction

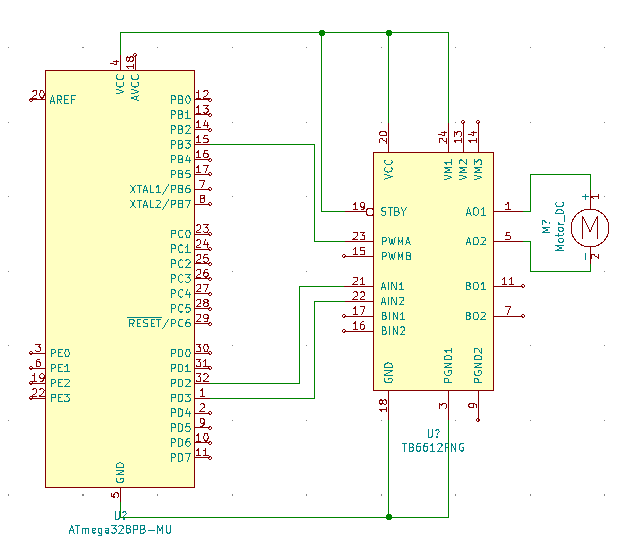
count = 0; //reset timer

}

}



*Midterm II Task 1 Pulseview showing 5s swap from CW to CCW rotation*



*Midterm II Circuit – Task 1*

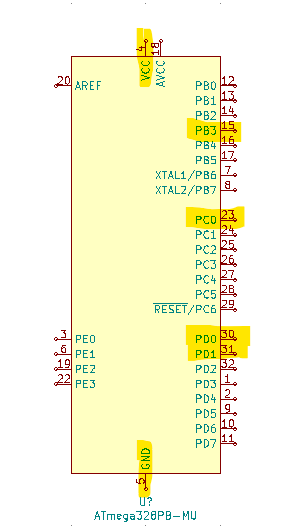
A picture containing electronics

Description automatically generated

*Midterm II Circuit – Task 1*

**Task 2:**

In task two, a potentiometer value (0~1023) is read using ADC from PC0 and translated to a (0~255) PWM duty cycle to control the speed of a DC motor. This task was achieved using Timer 2 as requested by the midterm document.



*ATmega328pb Ports Used in Task 2*

**Video Link:**

<https://youtu.be/BOGM-Dba--4>

**C Code:**

#define *F\_CPU* 16000000UL

#define BAUD 9600

#include <avr/io.h>

#include <util/setbaud.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdio.h>

volatile unsigned int pot; //temp read value from ADC

//ADC initialize PC0 input

void adc\_init(void){

ADMUX = (0 << REFS1) | (1 << REFS0) | (0 << ADLAR) | (0 << MUX3) | (0 << MUX2) |

(0 << MUX1) | (0 << MUX0);

ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADATE) | (0 << ADIF) | (0 << ADIE) | (1

<< ADPS2) | (1 << ADPS1) | (1 << ADPS0);

}

//read ADC value

void read\_adc(void){

ADCSRA |= (1 << ADSC);

while((ADCSRA & (1 << ADIF))==0); //wait for ADIF flag

ADCSRA |= (1 << ADIF); //reset ADIF flag

pot = ADC; //read value

}

//timer setup

void timer\_set(void){

TCCR2A |= (1 << COM2A1) | (1 << WGM21) | (1 << WGM20); //mode 3 fast PWM top 0xFF non-invert

TCCR2B |= (1 << CS22) | (1 << CS21) | (1 << CS20); //1024 prescale set

}

int main(void)

{

DDRB |= (1<<3); //set PORT B output PIN3

DDRD |= (1<<0) | (1<<1); //set PORTD outputs

//initialize function call

adc\_init();

//initialize function call

timer\_set();

PORTD |= (1<<0); //PINC0 high

//PORTD &= ~(1<<1); //PINC0 low

while (1)

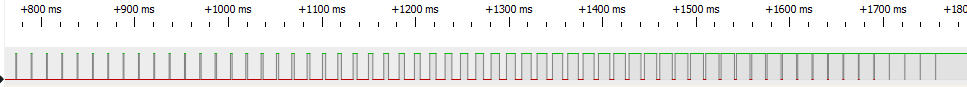
{

read\_adc(); //read ADC

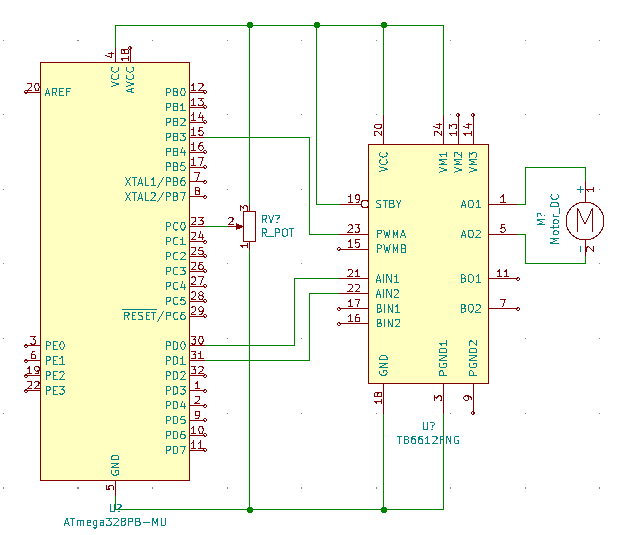
OCR2A = pot /4; //set OCR2A to pot2 value

}

}

**

*Midterm II Task 2 Pulseview showing a duty cycle sweep using the potentiometer*



*Midterm II Circuit – Task 2*

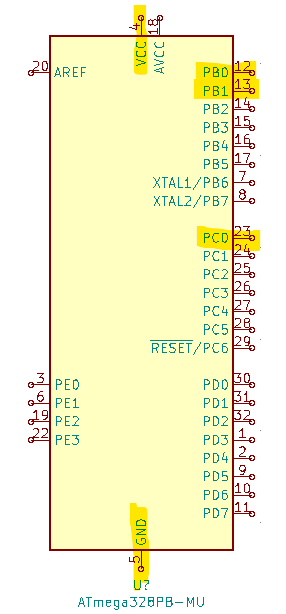
A picture containing electronics

Description automatically generated

*Midterm II Circuit – Task 2*

**Task 3:**

In task three the CCP capture pin is used to determine the speed of a DC motor using 1x mode. The motor speed is controlled using a 10k potentiometer connected to PINC 0 and read by ADC with the capture being read through PINB 0. Below you can see the output from the terminal as well as C code verifying functionality.



*ATmega328pb Ports Used in Task 3*

**Video Link:**

<https://youtu.be/gCH52aHjdlk>

**C Code:**

#define *F\_CPU* 16000000UL

#define BAUD 9600

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <util/setbaud.h>

#include <stdlib.h>

#include <stdio.h>

volatile *uint16\_t* pot; //global pot variable

volatile float rpm, rps; //rev per min, rev per sec

volatile *uint32\_t* tick\_rev; //ticks per rev

volatile *uint32\_t* rev\_count; //revolution count

volatile *uint16\_t* TIMER1ovf; //overflow value

volatile *uint32\_t* tick1; //temp tick value

volatile *uint32\_t* tick\_tot; //total ticks

char display[20]; //print variable to terminal

//ADC initialze using PC0 input

void adc\_init(void){

ADMUX = (0<<REFS1) | (1<<REFS0) | (0<<ADLAR) | (0<<MUX3) | (0<<MUX2) | (0 << MUX1) | (0 << MUX0);

ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);

}

//ADC read

void read\_adc(void){

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADIF))==0); //wait for flag

ADCSRA |= (1<<ADIF); //clear flag

pot = ADC; //read ADC

OCR1A = pot; //set OCR1A

}

//USART initialize

void USART\_init(void){

UBRR0H = *UBRRH\_VALUE*;

UBRR0L = *UBRRL\_VALUE*;

UCSR0C = \_BV(UCSZ01) | \_BV(UCSZ00);

UCSR0B = \_BV(RXEN0) | \_BV(TXEN0);

}

//print to terminal function

void USART\_tx\_string(char \*data){

while((\*data != '\0')){ //while data is not empty

while(!(UCSR0A & (1<<UDRE0))); //while data register is not empty

UDR0 = \*data; //UDR0 register gets data

data++; //next data value

}

}

//timer setup

void timer\_setup(void){

TCCR1A |= (1<<COM1A1) | (1<<WGM11) | (1<<WGM10); //mode 7 fast PWM

TCCR1B |= (1<<ICES1) | (1<<WGM12) | (1<<CS10); //prescale of 1

TIMSK1 |= (1<<ICIE1) | (1<<TOIE1); //set bits for interrupts

}

//RPM calculate

void rpm\_calc(void){

rps = 15625.0 / tick\_tot; //calc rpm

rpm = rps \* 60; //calc rps

if(rpm < 5) //check if RPM out of range

rpm = 0; //set RPM out of range

}

int main(void){

DDRB |= (1<<1); //output set

PORTB |= (1<<0); //pull up set

//setup functions

timer\_setup();

adc\_init();

USART\_init();

sei();

while (1){

read\_adc(); //read ADC and set OCR1A

rpm\_calc(); //calulate rpm

USART\_tx\_string("Motor RPM = ");

//display rpm to terminal

*snprintf*(display, sizeof(display), "%.2f ", (float)rpm);

USART\_tx\_string(display);

USART\_tx\_string(" \r\n");

}

}//TIMER1 capture ISR

ISR(TIMER1\_CAPT\_vect) {

tick1 = ICR1; //record tick value

tick\_tot = (*uint32\_t*)(tick1) + ((*uint32\_t*)TIMER1ovf \* 0x0400L);

rev\_count++; //increment revolution count

TCNT1 = 0; //reset TIMER1

TIMER1ovf = 0; //reset overflow

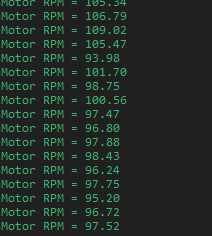
}

//TIMER1 overflow

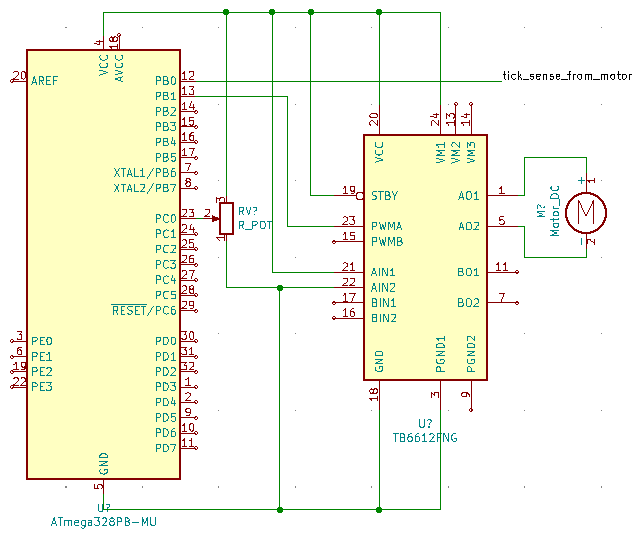
ISR(TIMER1\_OVF\_vect) {

TIMER1ovf++; //increment overflow

}



*TASK 3 Terminal output displaying motor RPM*



*Midterm II Circuit – Task 3*

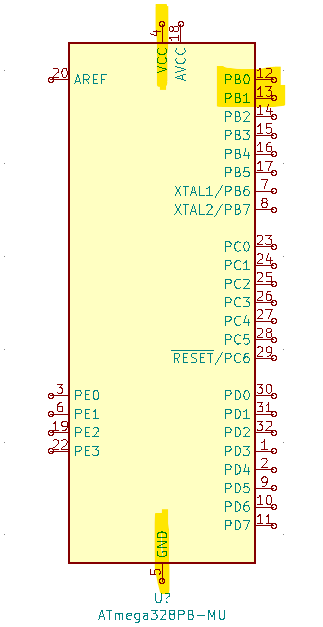
A picture containing electronics, adapter

Description automatically generated

*Midterm II Circuit – Task 3*

**Task 4:**

In task four the PID algorithm is used to control a DC motor speed, and maintain the motor’s speed adjusting for correction when needed. The motor PWM is supplied using OC1A as the output on PINB 1 and the capture is read through PINB0. Below is the C Code and SERIALPLOT outputs along with terminal output displaying the proper functionality.



*ATmega328pb Ports Used in Task 4*

**Video Link:**

<https://youtu.be/MDKxZyZVY60>

**C Code:**

#define *F\_CPU* 16000000UL //clock speed

#define BAUD 9600 //baud rate

#define IMEM 20

#define Pvalue 1.2

#define Ivalue 1.2

#define Dvalue 1.2

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/setbaud.h>

#include <stdlib.h>

#include <stdio.h>

volatile *uint16\_t* pot; //global ADC variable

volatile *uint32\_t* rev\_count, tick1, tick2, cap1,

cap2, cap3, cap4, avg1, avg2, tick\_tot;

int count = 0, edge = 0, finish = 0, tick\_count = 0;

volatile *int16\_t* duty = 0, duty\_max = 250;

volatile *int32\_t* mean\_I\_err[IMEM], PID = 0, err\_new = 0, err\_old = 0,

mean\_I\_error = 0.0, I\_err = 0.0,

P\_err = 0, D\_err = 0;

volatile *uint16\_t* TIMER1ovf; //overflow value

volatile float rpm, rps;

volatile double SETSPEED = 60, CURSPEED = 0;

char out[20], out1[20], out2[20]; //terminal print variables

//ADC initialize

void adc\_init(void){

ADMUX = (0<<REFS1) | (1<<REFS0) | (0<<ADLAR) | (0<<MUX3) | (0<<MUX2) | (0 << MUX1) | (0 << MUX0);

ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);

}

//read the ADC value

void read\_adc(void){

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADIF))==0); //wait for flag

ADCSRA |= (1<<ADIF); //clear flag

pot = ADC; //read ADC

}

//initialize USART

void USART\_init(void)

{

UBRR0H = *UBRRH\_VALUE*;

UBRR0L = *UBRRL\_VALUE*;

UCSR0C = \_BV(UCSZ01) | \_BV(UCSZ00);

UCSR0B = \_BV(RXEN0) | \_BV(TXEN0);

}

//print to terminal function

void USART\_tx\_string(char \*data){

while((\*data != '\0')){ //while data is not empty

while(!(UCSR0A & (1<<UDRE0))); //while data register is not empty

UDR0 = \*data; //UDR0 register gets data

data++; //next data value

}

}

//timer setup function

void timer\_setup(void){

//compare timer

TCCR2A |= (1<<WGM21) | (1<<WGM20);

TCCR2B |= (1<<CS22) | (1<<CS21) | ( 1<<CS20);

TIMSK2 |= (1<<OCIE2A);

//dc motor and capture timer

TCCR1A |= (1<<COM1A1) | (1<<WGM11) | (1<<WGM10); //mode 7 fast PWM

TCCR1B |= (1<<ICES1) | (1<<WGM12) | (1<<CS10); //prescale of 1

TIMSK1 |= (1<<ICIE1) | (1<<TOIE1); //interrupt set

}

int main(void)

{

DDRB |= (1<<1); //set PORTB outputs

PORTB |= (1<<0); //set pull up resistor

//setup functions

timer\_setup();

adc\_init();

USART\_init();

sei();

//initialize array

for(int i = 0; i < IMEM-1; i++){

mean\_I\_err[i] = (*int32\_t*)0.0;

}

while (1)

{

rps = 15625.0 / tick\_tot; //calc revolutions per second

rpm = rps\*60; //calc revolutions per minute

CURSPEED = rpm; //set RPM to curretn speed

*snprintf*(out, sizeof(out), "%.2f ", (float)rpm); //print speed

USART\_tx\_string(out);

USART\_tx\_string(" \r\n");

}

}

//PID algorithm

ISR(TIMER2\_COMPA\_vect){

err\_old = err\_new; //save new error

err\_new = SETSPEED - CURSPEED; //calc speed error

P\_err = err\_new; //save new to last

mean\_I\_err[IMEM-1] = (*int32\_t*)err\_new;

mean\_I\_error = 0;

//shift left and add all

for(int i=0; i<IMEM-1; i++){

mean\_I\_err[i] = (*int32\_t*)mean\_I\_err[i+1];

mean\_I\_error += (*int32\_t*)mean\_I\_err[i];

}

I\_err = (*int32\_t*)mean\_I\_error;

D\_err = (*int32\_t*)err\_new - err\_old;

//calc PID

PID = (*int32\_t*)Pvalue \* P\_err + Ivalue \* I\_err \* .1 + Dvalue \* D\_err / .1;

//PID out of bounds check

if(PID > duty\_max)

PID = duty\_max;

else if(PID < 0)

PID = 0;

duty = (*uint16\_t*)PID;

OCR1A = duty \* 7.5; //OCR1A scale calculation

}

//PID CAPTURE ISR

ISR(TIMER1\_CAPT\_vect) {

//rising edge 1

if(count==0 && edge==0){

tick1 = ICR1; //save tick time stamp

cap1 = (*uint32\_t*)(tick1) + ((*uint32\_t*)TIMER1ovf\*0x0400L);

rev\_count++; //increment revolution counter

count++; //increment count

edge = 1; //toggle edge value

TCCR1B ^= (1 << ICES1); //toggle capture select

}

//falling edge 1

else if(count==1 && edge==1){

tick1 = ICR1; //save tick time stamp

cap2 = (*uint32\_t*)(tick1) + ((*uint32\_t*)TIMER1ovf\*0x0400L);

rev\_count++; //increment revolution counter

count++; //increment count

edge = 0; //toggle edge value

TCCR1B ^= (1 << ICES1); //toggle capture select

}

//rising edge 2

else if(count==2 && edge==0){

tick1 = ICR1; //save tick time stamp

cap3 = (*uint32\_t*)(tick1) + ((*uint32\_t*)TIMER1ovf\*0x0400L);

rev\_count++; //increment revolution counter

count++; //increment count

edge = 1; //toggle edge value

TCCR1B ^= (1 << ICES1); //toggle capture select

}

//falling edge 2

else if(count==3 && edge==1){

tick1 = ICR1; //save tick time stamp

cap4 = (*uint32\_t*)(tick1) + ((*uint32\_t*)TIMER1ovf\*0x0400L);

rev\_count++; //increment revolution counter

count = 0; //reset count

edge = 0; //toggle edge value

TCCR1B ^= (1 << ICES1); //toggle capture select

finish = 1; //set final position value

}

//calculate avg / restart

if(finish == 1){

avg1 = cap3 - cap1; //avg1 calculation

avg2 = cap4 - cap2; //avg2 caclulation

tick\_tot = (avg1 + avg2) / 2; //total ticks

TCNT1 = 0; //reset timer

TIMER1ovf = 0; //reset overflow

finish = 0; //reset final position value

}

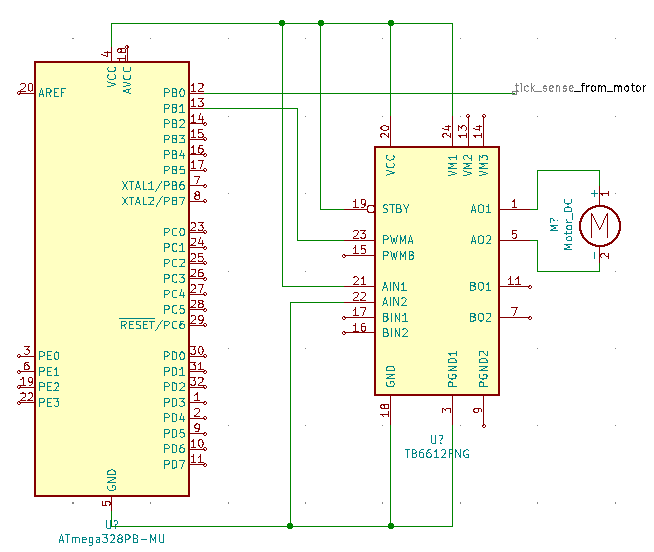
}

//TIMER1 OVF ISR

ISR(TIMER1\_OVF\_vect) {

TIMER1ovf++; //increment overflow count

}

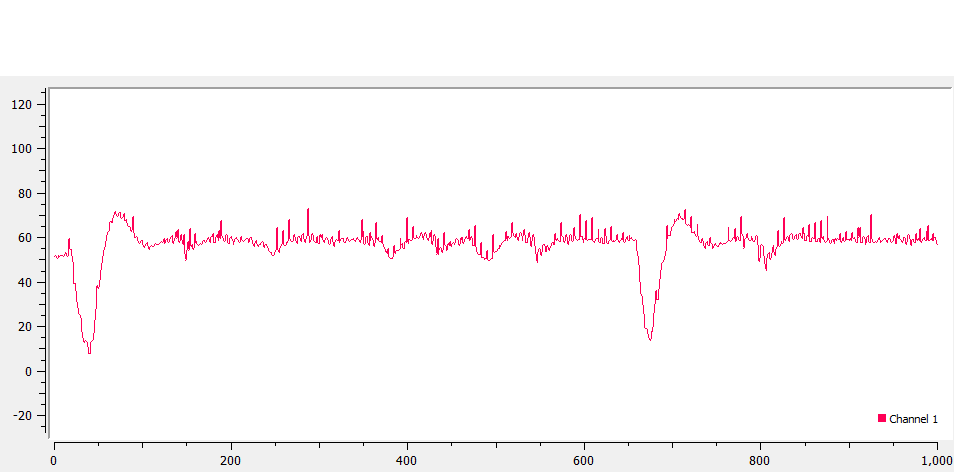


*Midterm II Circuit - Task 4*

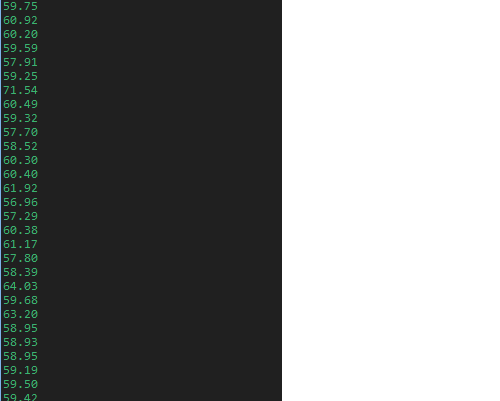
*A picture containing electronics, adapter

Description automatically generated*

*Midterm II Circuit – Task 4*



*TASK #4 SerialPlot Output – showing PID maintain 60 rpm*

**

*TASK #4 Terminal Output showing PID control maintain 60 rpm*

**Github Link:**

[submission\_da/MIDTERM\_2 at main · brianwolak/submission\_da (github.com)](https://github.com/brianwolak/submission_da/tree/main/MIDTERM_2)

**Video Links:**

**Task1:** <https://youtu.be/uggxgGmAdAE>

**Task2:** <https://youtu.be/BOGM-Dba--4>

**Task3:** <https://youtu.be/gCH52aHjdlk>

**Task4:** <https://youtu.be/MDKxZyZVY60>