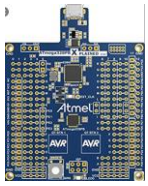
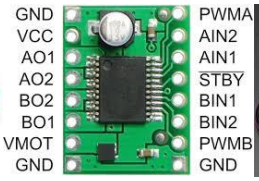
CPE301 – SPRING 2019

Midterm 2

Student Name: Meral Abu-Jaser
Student #: 5003137888
Student Email: abujaser@unlv.nevada.edu
Primary Github address: https://github.com/MeralAbuJaser/Submission_da.git
Directory: https://github.com/MeralAbuJaser/Submission_da/tree/master/Midterm%202

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

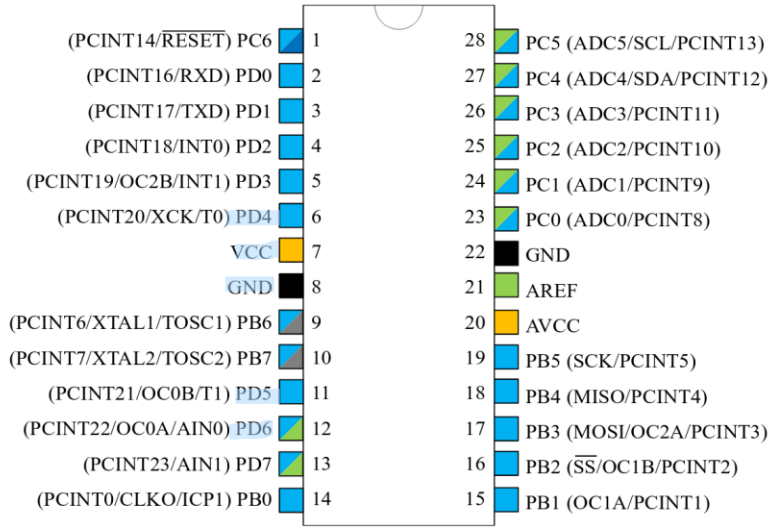



Atmega 328pb

Pin-out

Figure 5-1. 28-pin PDIP



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

```
/*
 * Midterm2_Task1.c
 *
 * Created: 5/9/2020 3:59:25 AM
 * Author : Meral
 */
#include <avr/io.h>
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void){
	int num;
	DDRD |= (1<< DDRD4) | (1<< DDRD5) | (1<< DDRD6);	//pins PD4, PD5, and PD6 are outputs
	OCR0A = 128;					//50% duty cycle
	TCCR0A |= (1<< COM0A1) | (1<<WGM01) | (1<< WGM00);	//enable non-inverting, Fast PWM
	TCCR0B |= (1<< CS00);				//pre-scaler 1

	while (1){
		//clockwise direction
		if(num == 1){
			PORTD|=(1<<DDD5);
			PORTD&=~(1<<DDD4);
			_delay_ms(3000);
			num = 0;	//set value to counterclockwise
		}
		//counterclockwise direction
		if(num == 0){
			PORTD&=~(1<<DDD5);
			PORTD|=(1<<DDD4);
			_delay_ms(3000);
			num = 1;	//set value to clockwise
		}
	}
}
```

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

```
/*
 * Midterm2_task2.c
 *
 * Created: 5/9/2020 4:03:22 PM
 * Author : Meral
 */
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
volatile int adc_value;
// Initialize ADC
void adc_init(void) {
	/**Setup and enable ADC**/
	ADMUX =  (0<<REFS1) | //Reference selection bits
	(1<<REFS0) | //AVcc - external cap at AREF (5)V
	(1<<ADLAR) | //ADC right adjust result
	(0<<MUX2)  | //Analog channel selection bits
	(0<<MUX1)  | //ADC4 (PC4 PIN27)
	(0<<MUX0);

	ADCSRA = (1<<ADEN)  | //ADC enable
	(0<<ADSC)  | //ADC start conversion
	(0<<ADATE) | //ADC auto trigger enable
	(0<<ADIF)  | //ADC interrupt flag
	(0<<ADIE)  | //ADC interrupt enable
	(1<<ADPS2) | //ADC Prescaler select bits
	(1<<ADPS1) | //128 AS PRESCALAR SELECION BIT
	(1<<ADPS0);    //Select channel
}
void read_adc(){
	ADCSRA |= (1 << ADSC);                      //enable conversion
	while(ADCSRA & (1<< ADSC));
	adc_value = ADCH;                           //value of the Potentiometer is stored in adc_value
}
int main(void){
	int num;
	adc_init();
	DDRD |= (1<< DDRD4) | (1<< DDRD5) | (1<< DDRD6);          //pins PD4, PD5, and PD6 are outputs
	OCR0A = 128;                                              //50% duty cycle
	TCCR0A |= (1<< COM0A1) | (1<<WGM01) | (1<< WGM00); //enable non-inverting, Fast PWM
	TCCR0B |= (1<< CS00);                                     //pre-scaler 1

	while (1){
		read_adc();
		OCR0A = adc_value;
		//clockwise direction
		if(num == 1){
			PORTD|=(1<<DDD5);
			PORTD&=~(1<<DDD4);
			_delay_ms(5000);
		}
		num = 0;       //set value to counterclockwise
		_delay_ms(1000);
		//counterclockwise direction
		if(num == 0){
			PORTD&=~(1<<DDD5);
			PORTD|=(1<<DDD4);
			_delay_ms(5000);
			num = 1;       //set value to clockwise
		}
	}
	return 0;
}
```

Task 3

```
/*
 * Midterm2_task3.c
 *
 * Created: 5/9/2020 4:43:06 PM
 * Author : Meral
 */
#define F_CPU 16000000UL
#define BAUD 9600
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>

volatile uint32_t Counter = 0;
volatile float potentialMeter = 0;//to store Potentiometer value
volatile double temp_rpm = 0;//to store rpm value
volatile uint32_t encoder = 0;//determine which encoder is used either 1x or 2x

volatile double RPM1X = 0;
volatile double RPM2X = 0;

double period1x = 0;//to store period of rise/fall
double period2x = 0;//to store period of rise/fall

void USART_init(){
	UBRR0H = UBRRH_VALUE;
	UBRR0L = UBRRL_VALUE;
	UCSR0C = _BV(UCSZ01) | _BV(UCSZ00);//8-bit data
	UCSR0B = _BV(RXEN0) | _BV(TXEN0);//Enable RX and TX
}
// Initialize ADC
void adc_init(void) {
	/**Setup and enable ADC**/
	ADMUX =  (0<<REFS1) | //Reference selection bits
	(1<<REFS0) | //AVcc - external cap at AREF (5)V
	(1<<ADLAR) | //ADC right adjust result
	(0<<MUX2)  | //Analog channel selection bits
	(0<<MUX1)  | //ADC4 (PC4 PIN27)
	(0<<MUX0);

	ADCSRA = (1<<ADEN)  | //ADC enable
	(0<<ADSC)  | //ADC start conversion
	(0<<ADATE) | //ADC auto trigger enable
	(0<<ADIF)  | //ADC interrupt flag
	(0<<ADIE)  | //ADC interrupt enable
	(1<<ADPS2) | //ADC Prescaler select bits
	(1<<ADPS1) | //128 AS PRESCALAR SELECION BIT
	(1<<ADPS0);    //Select channel
}
```

```
//Send data to the serial port
void USART_tx_string( char *data ){
	while ((*data != '\0')){
		//while the register is empty enter date
		while (!(UCSR0A & (1 <<UDRE0)));
			UDR0 = *data;
			data++; //increment data location forward
		}
}
void read_adc(){
	unsigned char i = 10;
	while(i--){
		ADCSRA |= (1<<ADSC);
		while(ADCSRA & (1<<ADSC));
		potentialMeter += ADC;
	}
	potentialMeter = potentialMeter/10;//average the adc values
}
void rpm1x(){
	unsigned char i = 10;
	while(i--){
		/*
		divide by 16000000, then we convert to seconds
		*/
		temp_rpm = (1/((period1x/16000000)*960))*60;
		RPM1X += temp_rpm;//sum up the values
	}
	RPM1X = RPM1X/10;//average the value of rpm by didving by the count of loops
}
void rpm2x(){
	unsigned char i= 10;
	while(i--){
		/*
		we divide by two to get the value of the rising and faling edge
		then the value is converted by diving it by 1 and multiplying by 60
		*/
		temp_rpm = (1/(((period2x/2.0)/16000000)*960))*60;
		RPM2X += temp_rpm;//sum up the values
	}
	RPM2X = RPM2X/10;//average the value of rpm by didving by the count of loops
}
ISR(TIMER1_CAPT_vect){	//Capture ISR
	volatile uint32_t duration = 0;
	if(encoder == 0){	//calculate the period for 2x
		encoder = 1;	//set encoder to be 1
		duration = ICR1;	//save duration
		period2x = (duration + (Counter * 65535) + period1x);//calculate total period
		TCNT1 = 0;	//reset timer1
		Counter = 0;	//reset overflow counter
		TCCR1B |= (1 << ICES1);
	}
	if(encoder == 1){	//calculate the period for 1x
		encoder = 0;	//reset the encoder
		duration = ICR1;	//save duration
		period1x = (duration + (Counter * 65535));//calculate total period using overflow timer and
ticks
		TCNT1 = 0;	//reset timer1
		Counter = 0;	//reset overflow counter
		TCCR1B |= (0<<ICES1);
	}
}
```

```
ISR(TIMER1_OVF_vect){
		Counter++;//overflow counter
}
void timer(){
	TCCR0A |= (1<<COM0A1)|(0<<COM0A0);//fast PWM, non-inverted
	TCCR0A |= (1<<WGM02)|(1<<WGM01)|(1<<WGM00);
	TCCR0B |= (1<<CS00);//no prescaling
	TCNT1 = 0;//initialize timer timer1 counter to 0
	TCCR1A = 0;//set the capture to start
	TCCR1B = (0<<ICNC1)|(1<<ICES1)|(1<<CS10);//start timer
	TCCR1C = 0;
	TIFR1 = (1<<ICF1)|(1<<TOV1);//interrupt enabled
	TIMSK1 = (1<<ICIE1)|(1<<TOIE1);
}
void readPotentialMeter(){
	if ((potentialMeter >= 62260) && (potentialMeter < 65535)){//PWM is 95% which is the limit
			OCR0A = 62260;//max PWM value is 95%
			_delay_ms(1000);
	}
	else if ((potentialMeter < 62257) && (potentialMeter >= 3250)){//PWM less then continue to read
ADC value
	OCR0A = potentialMeter;//save potentiometer value in registor
	_delay_ms(1000);
	}
	else{//if PWM is below 5% reset value
		OCR0A = 0;//reset compare registor
		_delay_ms(100);
	}
}
void display(){
	char value1x[20];//buffer to store 1x rpm
	char value2x[20];//buffer to store 2x rpm
	USART_tx_string("\n1x: ");
	sprintf(value1x, "%f", RPM1X);
	USART_tx_string(value1x);//display 1x value

	USART_tx_string("\n2x: ");
	sprintf(value2x, "%f", RPM2X);
	USART_tx_string(value2x);//display 2x value
	USART_tx_string("\n\n");
	_delay_ms(1000);
}
int main(void){
	OCR0A = 0;//set register to 0
	//setting the input and output pins according to the wiring
	DDRD |= (1<<6);
	DDRD |= (1<<5);
	DDRD |= (1<<4);
	DDRC &= ~(1<<0);
	PORTC |= (1<<0);
	DDRB &= ~(1 << DDB0);
	adc_init();//intialize adc
	USART_init();//initialize UART
	timer();//initialize PWM
	sei();//enable global interrupts
	while (1){
		PORTD |=(1<<DDD5);
		PORTD &=~(1<<DDD4);
		read_adc();
		readPotentialMeter();
		rpm1x();
		rpm2x();
		display();
	}
}
```

Task 4

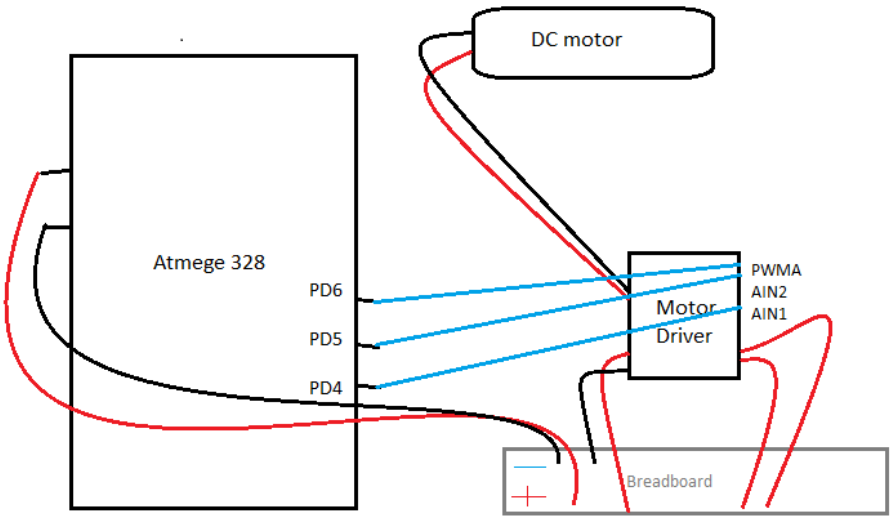
```
/*
 * Midterm2_task4.c
 *
 * Created: 5/9/2020 9:36:46 PM
 * Author : Meral
 */
#define BAUD 9600
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <util/setbaud.h>
volatile char size_of_outPut[20];
volatile uint8_t button = 0;              //ON/OFF switch button
volatile float rpm4x;                           //stores the speed
volatile uint32_t period4x = 0;           //stores the period of the rise/fall edge
volatile uint16_t Counter1= 0;            //overflow counter
volatile uint16_t Counter2 = 0;           //
volatile uint16_t overflows = 0;
volatile uint16_t capture = 0;
//Send data to the serial port
void USART_tx_string( char *data ){
       while ((*data != '\0')){
              //while the register is empty enter date
              while (!(UCSR0A & (1 <<UDRE0)));
                     UDR0 = *data;
                     data++; //increment data location forward
              }
}
void USART_init(){
       UBRR0H = UBRRH_VALUE;
       UBRR0L = UBRRL_VALUE;
       UCSR0C = _BV(UCSZ01) | _BV(UCSZ00);//8-bit data
       UCSR0B = _BV(RXEN0) | _BV(TXEN0);//Enable RX and TX
}
void display(){
       float speed[20];     //buffer to store values
       _delay_ms(500);
       snprintf(speed, sizeof(speed), "4x: %f ", rpm4x);
       USART_tx_string(speed);     //display value
       USART_tx_string("\n");
}
void calculate_speed(){
       //calculate the speed of the motor in rpm
       period4x = (capture + overflows*0x10000L);
       rpm4x = (960000000/(12*period4x))/80; //calculating and converting to rpm
}
void enableCapture() {
       //sets ports directions
       DDRB &= ~(1<<PINB0);
       TCCR1A = 0x00;                     //normal mode
       TCCR1B = 0x41;                     //enables capture on rising edge
       TCCR1C = 0x00;
       TIFR1 = (1<<ICF1)|(1<<TOV1);
       TIMSK1 = (1<<ICIE1)|(1<<TOIE1);//capture interrupt and overflow interrupt
}
```

```
void timer(){
	TCCR1A = 0x00;                    //normal mode timer
	TCCR1B = 0x41;                    //rising edge capture
	TCCR1C = 0x00;
	TIFR1 = (1<<ICF1)|(1<<TOV1);
	TIMSK1 = (1<<ICIE1)|(1<<TOIE1);   //capture interrupt and overflow interrupt
	TCCR3A = 0x00;
	TCCR3B = 0x45;                    //capture on falling edge enable
	TCCR3C = 0x00;
	TIFR3 = (1<<ICF3)|(1<<TOV3);
	TIMSK3 = (1<<ICIE3)|(1<<TOIE3);
}

float read_adc(){
	ADCSRA |= (1<<6);                       // Enables ADC conversion
	while (!(ADCSRA&(1<<4)));         // Waits until ADIF is set ADC is done
	ADCSRA |= (1<<4);                       // Clears flag
	return ADC;                                     // Returns value
}
void initialize(){
	DDRC &= ~(1<<PINC0);  //potentiometer input
	ICR1 = 0XFFFF;
	ADMUX = 0x40;        //right justify
	ADCSRA = 0x87;              //128 pre-scaler
	DDRC &= ~(1 << PINC1);
	PCICR = (1 << PCIE1);       //pin change interrupt enabled
	PCMSK1 = (1 << PCINT9);     //
}
int main(void){
	timer();
	USART_init();
	float duty = 0;      //store duty cycle
	initialize();
	sei();               //enable global interrupt

	_delay_ms(1000);
	USART_tx_string("Press S1-A1 button to start reading accurate values\n");
	USART_tx_string("use the potentiometer to change the values");
	_delay_ms(2500);

	while (1){
		read_adc();
		calculate_speed();
		duty = 100*(read_adc()/1023);            //calculated the potentiometer limit value
		OCR0A = (duty/100)*255;
		display();
	}
}
ISR(TIMER1_CAPT_vect) {
	capture = ICR1;                         //capture ICR3
	TCNT1 = 0;
	overflows = Counter1;
	Counter1 = 0;                           //reset counter
	TIFR1 |= (1<<ICF1);                     //clears flag
}
ISR(TIMER3_CAPT_vect) {
	capture = ICR3;                         //capture ICR3
	TCNT3 = 0;
	overflows = Counter2;
	Counter2 = 0;                     //reset counter
	TIFR3 |= (1<<ICF3);                     //clear flag
}
```

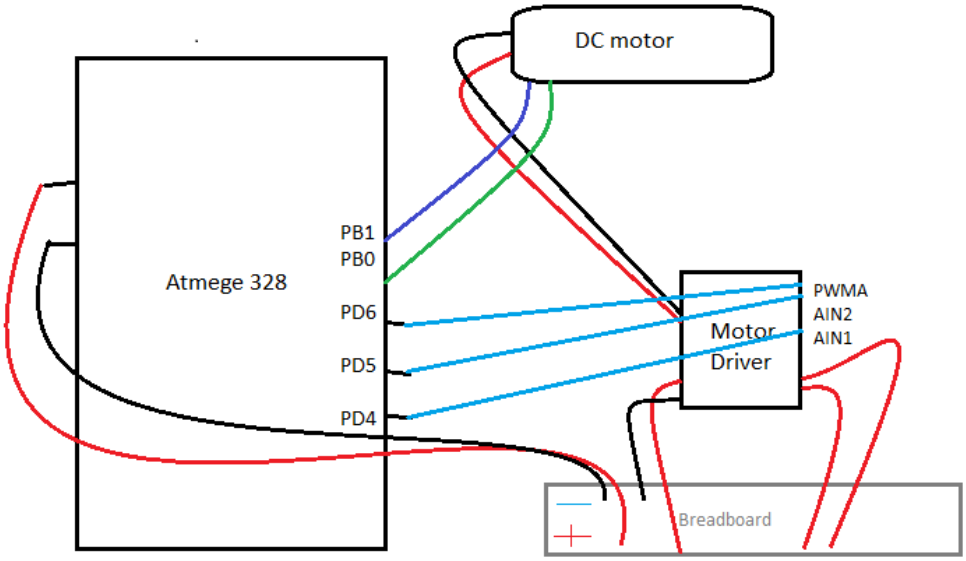
```
ISR(TIMER1_OVF_vect) {
	Counter1++;
}

ISR(TIMER3_OVF_vect) {
	Counter2++;
}

ISR(PCINT1_vect){
	if (PINC & (1<<PINC1)){
		button = ~button;
			//clockwise direction
			if (button){
				DDRD |= (1<<PIND6);
				PORTD|=(1<<DDD5);
				PORTD&=~(1<<DDD4);
				TCCR0A = 0x83;
				TCCR0B = 0x02;
			}
		//change direction of the motor to counterclockwise
		if(!button){
			DDRD |= (1<<PIND6);
			PORTD&=~(1<<DDD5);
			PORTD|=(1<<DDD4);
			TCCR0A = 0x83;
			TCCR0B = 0x02;
		}
	}
	PCIFR = (1<<PCIF1);   //clears flag
}
```

4. SCHEMATICS

task 1 and task 2



Task 3 and task 4



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

Task 1

```
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Midterm2_Task1.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\Meral\Docume
Done building target "Build" in project "Midterm2_Task1.cproj".
Done building project "Midterm2_Task1.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

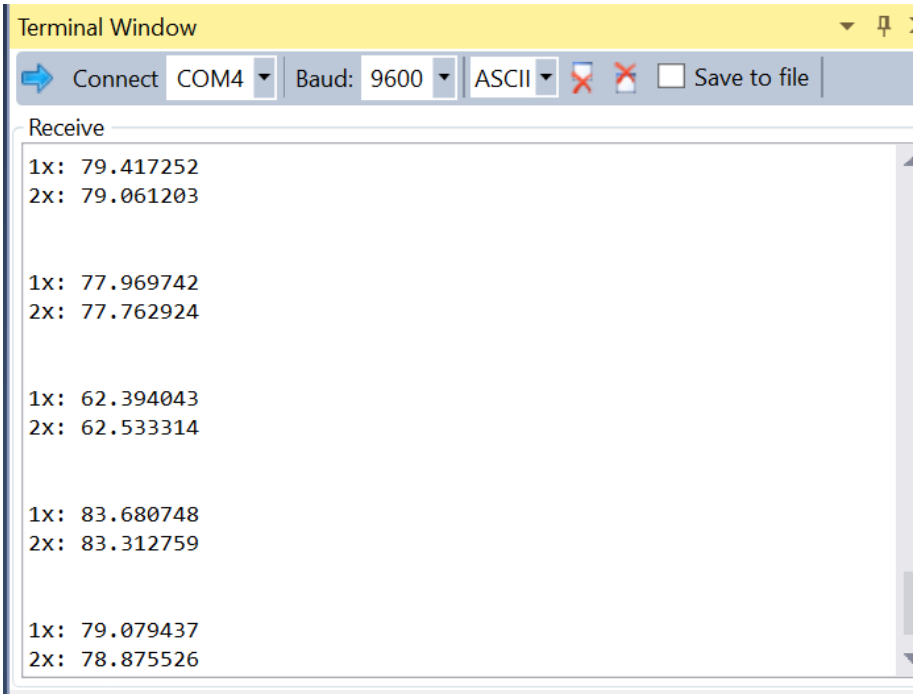
Task 2

```
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Midterm2_task2.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\Meral\Doc
Done building target "Build" in project "Midterm2_task2.cproj".
Done building project "Midterm2_task2.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Task 3

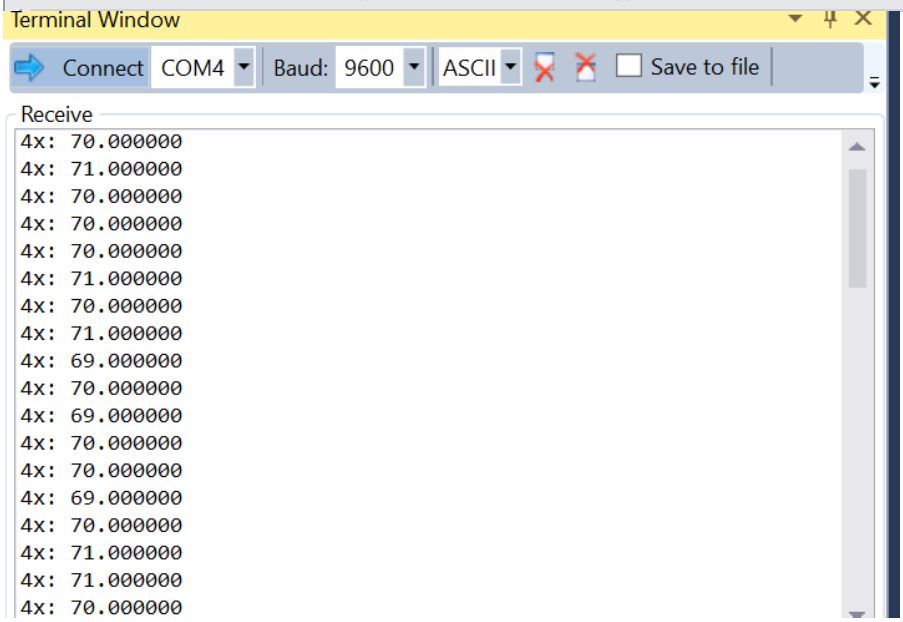
```
Done building target "CoreBuild" in project "Midterm2_task3.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\meral\Doc
Done building target "Build" in project "Midterm2_task3.cproj".
Done building project "Midterm2_task3.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Terminal window



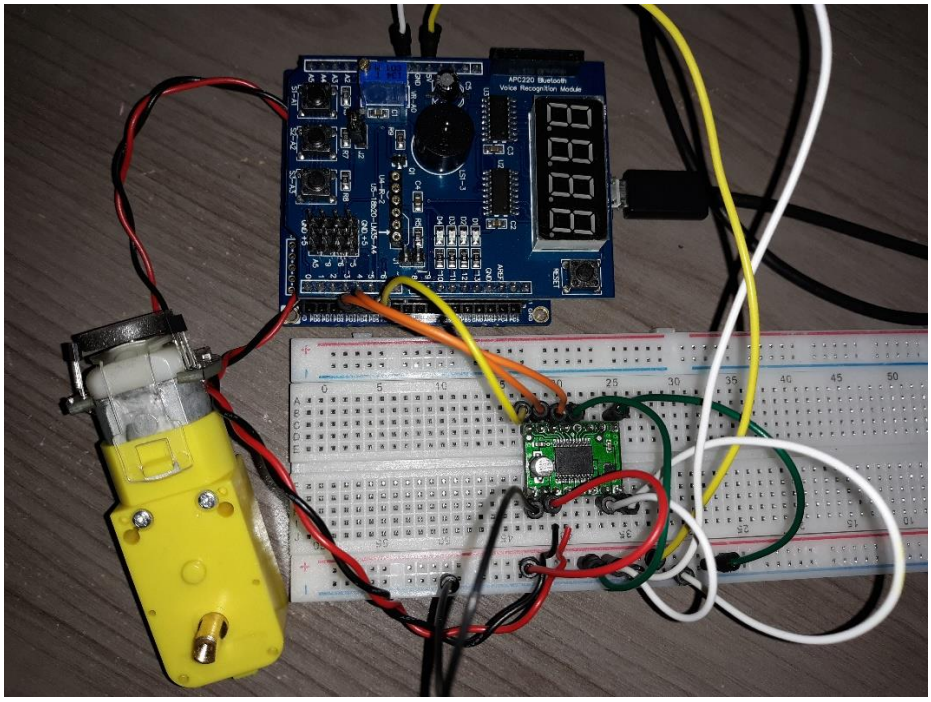
Task 4

```
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Midterm2_task4.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\meral\Document
Done building target "Build" in project "Midterm2_task4.cproj".
Done building project "Midterm2_task4.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

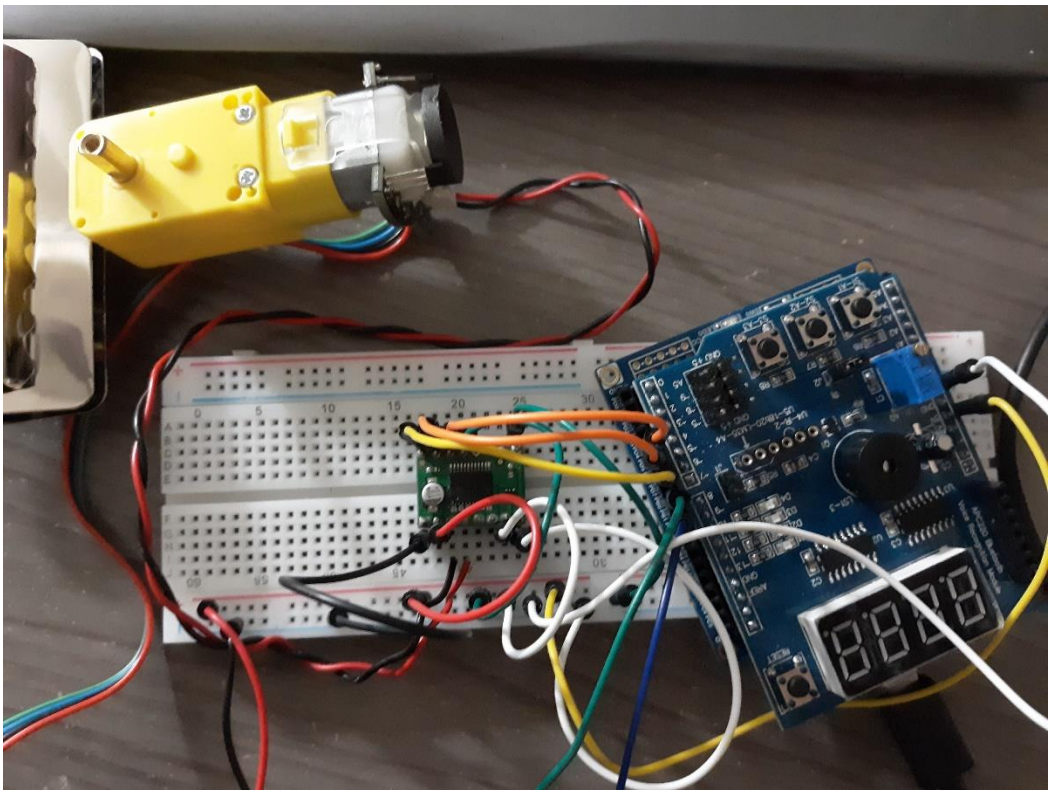


6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Task and task 2



Task 3 and task 4



7. **VIDEO LINKS OF EACH DEMO**

Task 1
https://www.youtube.com/watch?v=EtwA_S_jP0A
task 2
https://www.youtube.com/watch?v=4aROcVPRbes
task 3
https://www.youtube.com/watch?v=E_CkFYN_O8A
task 4
https://www.youtube.com/watch?v=VSFk8os6qvY

8. **GITHUB LINK OF THIS DA**

https://github.com/MeralAbuJaser/Submission_da/tree/master/Midterm%202

"*This assignment submission is my own, original work*".
Meral Abu-Jaser