ASHIM GHIMIRE

CPE301 – SPRING 2014

FINAL PROJECT:

PID CONTROL USING DC MOTOR

**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)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

-Atmega328p

-Motor driver (TB6612FNG)

-AVRISP mkII

- AtmegaXX8 Target Board

-Pololu USB AVR programmer

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

///PID controller of DC MOTOR

#define F\_CPU 8000000UL

#define P\_GAIN 0.8

#define I\_GAIN 0.005

#define D\_GAIN 0.01

#define BAUD 9600

#define BRC ((F\_CPU/16/BAUD) - 1) //Asynchrnonous Normal Mode

//calculating Baud Rate to feed into UBBR

#define TX\_BUFFER\_SIZE 128

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdlib.h>

#include <stdio.h>

#include <inttypes.h>

double dutycycle = 0;

int set\_rpm = 0,previous\_error = 0;

int error, feedback\_rpm = 0 , output;

int D\_error = 0, I\_error = 0;

int cliflag = 0;

unsigned int pulses;

char buffer[999];

void usart\_init(unsigned int ubrr) //initialize usart

{

UBRR0H = (unsigned char)(ubrr>>8);

UBRR0L = (unsigned char)ubrr;

//Enable receiver and transmitter

UCSR0B = (1<<RXEN0)|(1<<TXEN0);

// Set frame format: 8data, 2stop bit

UCSR0C = (1<<USBS0)|(3<<UCSZ00);

}

void usart\_send (unsigned char ch) //send data to terminal

{

while (!(UCSR0A & (1 << UDRE0)));

UDR0 = ch;

}

void USART\_tx\_string(char \* data) //send a string to the terminal

{

while ((\*data != '\0'))

{

while (!(UCSR0A & (1<<UDRE0)));

UDR0 = \*data;

data++;

}

}

int main(void)

{

DDRB = 0xFF; //Set output to PORTB

PORTB = 0x02; //Constant 10 to PORTB1, PORTB0

set\_rpm = 100;

usart\_init(BRC);

\_delay\_ms(100);

snprintf(buffer,sizeof(buffer), "Displaying the motor speed in RPM\n\r");

//USART\_tx\_string("Displaying the motor speed in RPM\n\r"); //Display the String

USART\_tx\_string(buffer);

PORTD = (1<<PORTD3); // sets 1 to INT1 to trigger interrupt on falling edge

EIMSK = (1<<INT1); //enables the external interrupts

//sei();

DDRD = (1<<PORTD6); //send PWM

//since we're dealing with only OC0A in the register, we're only gonna look at 0A bit of timer

//use fast pwm

TCCR0A = (1<<COM0A1) | (1<<WGM00) | (1<<WGM01); //Clears OC0A on compare Match, set OC0A at BOTTOM

//TOP value is 0xff i.e. 255 for a 8 bit timer

TIMSK0 = (1<<TOIE0); // interrupt set when there is an overflow

OCR0A = (dutycycle/100.0)\*255; //fraction of 255 representing the amount of time the PWM is on

sei();

TCCR0B = (1<<CS00); //No prescalar. THIS ONE ACTUALLY STARTS THE TIMER.

previous\_error = set\_rpm - feedback\_rpm;

while (1)

{

// dutycycle += 10;

error = set\_rpm - feedback\_rpm;

I\_error += (error);

D\_error = (error - previous\_error);

output = (P\_GAIN \* error) + (I\_GAIN \* I\_error) + (D\_GAIN \* D\_error);

previous\_error = error;

snprintf(buffer,sizeof(buffer), "The current speed is: %d RPM \r\n\n", OCR0A);

//send the message through USART

USART\_tx\_string(buffer);

if(OCR0A + output > 255){

OCR0A= 255;

}

else if(OCR0A + output < 0){

OCR0A= 0;

}

else{

OCR0A += output;

}

if(dutycycle >100)

{

dutycycle = 0;

}

}

}

ISR (TIMER0\_OVF\_vect)

{

OCR0A =(dutycycle/100.0)\* output; //it recalculates the duty cycle making PWM dynamic

}

ISR (INT1\_vect)

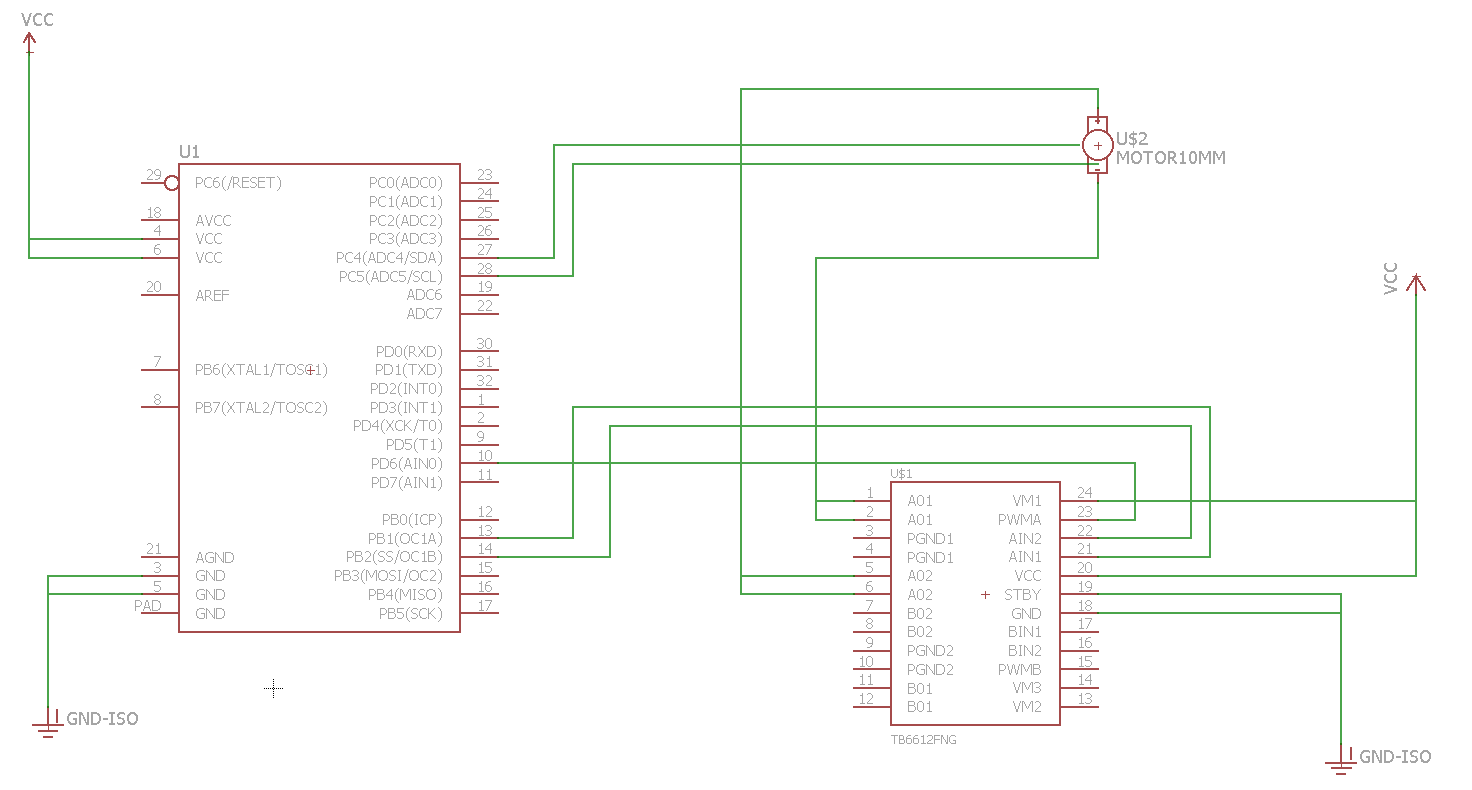
{

pulses++;

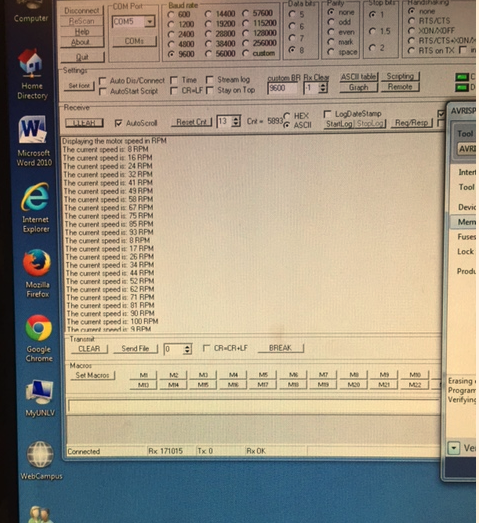
feedback\_rpm = (pulses/64)\*60;

}

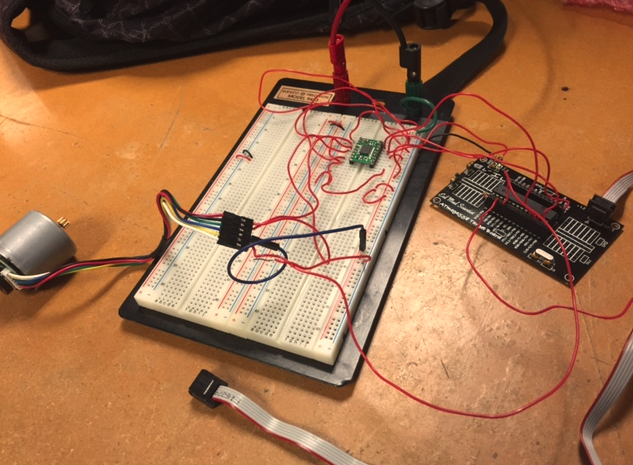
|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT OF EACH DEMO |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| https://www.dropbox.com/s/kg4x03t7lbd86k9/Video%20May%2014%2C%2011%2021%2047%20AM.mov?dl=0 | | | |
| 10. | GITHUB LINK OF THE DA |  |  |
| http:// https://github.com/Ashim-Ghimire/301\_GHIMIRE | | | |

**Student Academic Misconduct Policy**

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

“This assignment submission is my own, original work”.

NAME OF THE STUDENT