**PID Speed Control of DC Motor**

**GOAL:**

* PID Speed Control of DC Motor with Encoder Feedback.
* Use USART interface for required speed and current speed as the input and display

**DELIVERABLES:**

This project ask the user for a desired speed and displays the current speed of the motor. Additionally once the push button is press the motor will do a slow stop. I was unable to get a clear signal from my encoders. The encoder kept sending high with no falling edge to replicate a pwm.

**Planning Process**

Initially I started by running a DC Motor using a PWM. This is done by using registers TCCR0A/B (PMW and pre-scale) and OCR0A (rising edge signal for motor). Once complete the next step was to use a push-button to bring the motor to a slow stop. Next using USART I was able to ask the user for a desired speed and set the motor to that speed. The motor speed range is 0-255 and needs to be changed to a RPM unit. I encountered problems with my Hall Effect sensors. My sensors sent a high signal without ever reaching a falling edge. I believe they were picking up some constant magnetic field from the coil inside the DC motor. If the encoders worked the plan was to store them in a counter. Using the encoder counter the current motor speed can be found and stored. By comparing the current speed of the motor with the desired speed get an error value. This value corresponds with the P component of the PID. Next the I term integrates the previous current and desired speed error in order to eliminate the residual error from continuous use. I was not planning to implement the D component as it is not needed for speed control. If we assume a gain of 1 then the new value for OCR0A would have been the sum of the P and I component added to the original value of OCR0A (current speed). The result is a new controlled speed for the DC motor. The terminal should then prompt the user for another speed.

**COMPONENTS:**

**L293D Motor Driver**

The L293D is a quadruple high-current half-H drivers. The L293 is designed to provide bi-direction drive currents. When enable is high the inputs 1 will be the pmw driving the DC motor

**DC Motor**

For this project I choose to use a 6V DC motor. The shaft of the motor is 25mm, chosen to match the settings for the encoders.

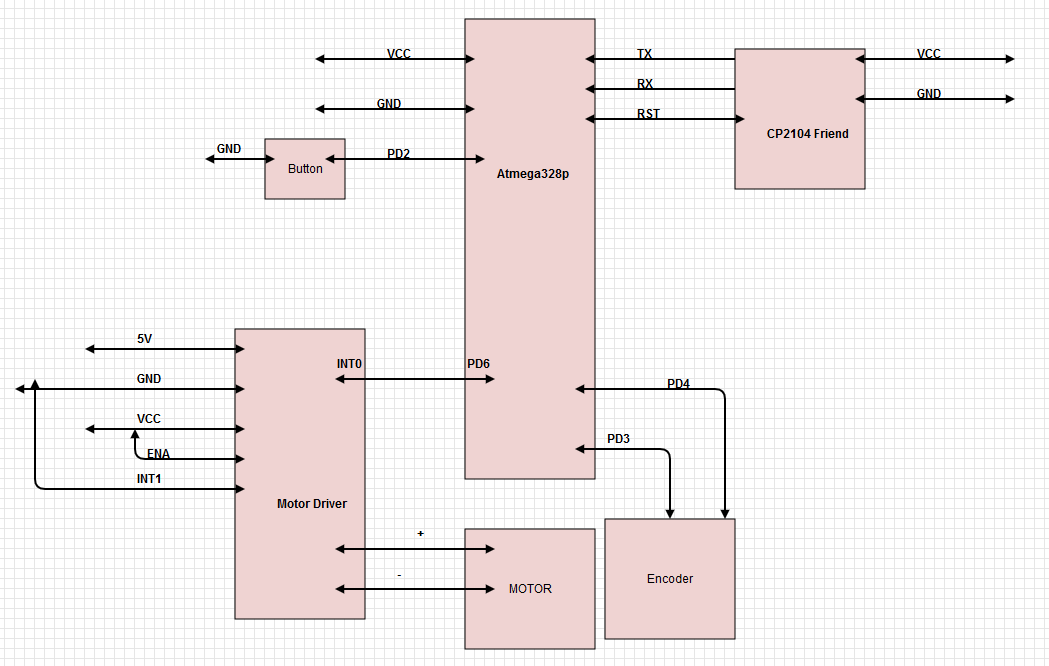
**Encoders**

Two a3144 hall effect sensors were attach to the 25mm motor. These sensors work by identifying a magnetic field and varies its output voltage proportionally. These are used for speed detection.

**ATmega328p**

The ATmega328p is an 8 bit microcontroller by Atmel. This microcontroller works by programming in either C or assembly. The code written will allow the users to determine what pins they want to set as input and output. This microcontroller consists of 32 general purpose registers. The ATmega328p consists of 6 PWMs, 3 timer/counters, up to 1kB EEPROM, 2kB SDRAM, I2C interface, SPI interface, 10-bit ADCs, and an 8MHz clock. The ATmega328p, however, is limited in the amount of pins can be used as input and output. Also, only one 16 bit timer exists on the chip and only one USART module is included.

**SCHEMATICS:**



**IMPLEMENTATION:**

* The DC motor interfaced with USART receives commands from the Terminal.
  + USART and the FTDI module was used to ensure correct data was being received from the IMU
* The encoder is used to measure the current speed of the DC Motor
* ATmega328p was used to link the current speed and desired speed change.
  + Here is where PID calculations are done. The Kp error is the difference between current and desired speed.
  + The difference is added to current speed to reach desired speed.

**SNAPSHOTS/SCREENSHOTS\*:**

https://www.youtube.com/watch?v=Rc9YwFjyqfc

**CODE:**

**Current working code:**

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdbool.h>

#define BAUDRATE 9600

#define BAUD\_PRESCALER (((F\_CPU/(BAUDRATE\*16UL)))-1)

#define F\_CPU 8000000UL

void USART\_INIT(void)

{

// This function sets up the microcontroller for serial communication

// The baud rate, registers, and prescaler value are all set.

UBRR0H = (uint8\_t)(BAUD\_PRESCALER>>8);

UBRR0L = (uint8\_t)(BAUD\_PRESCALER);

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

UCSR0C = (3<<UCSZ00);

}

void USART\_SEND(unsigned char data)

{

// This function sends a single character from the ATmega328P to the computer.

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

UDR0=data;

}

unsigned char USART\_RECEIVE(void)

{

while(!(UCSR0A & (1<<RXC0)));

return UDR0;

}

int main(void)

{

DDRD |= (1<<PD6) | (1<<PD1) ;

DDRD &= ~(1<<PD2);

PORTD |= (1<<PD2); // pull up resistor

DDRB=0xFF; // portB is output for pwm.

TCCR0A |= (1<<COM0A1)|(1<<WGM00)|(WGM01);  //set pwm fast

TCCR0B |= (1<<CS00); //Configure TCCR0 as explained in the article

unsigned char promptSpeed[35] = "This is the Current Speed: ## RPM\n";

unsigned char desiredSpeed[41] = "How Fast Would you like the motor to go? ";

unsigned char desiredSpeedNum[3];

USART\_INIT();

OCR0A=100; // Set OCR0 to 255 so that the duty cycle is initially 0 and the motor is not rotating

// Tell user the current speed

for(int i=0; i< 35; i++)

{

USART\_SEND(promptSpeed[i]);

\_delay\_ms(10);

}

USART\_SEND(10); // send line feed

// Ask user for speed

for(int i=0; i< 41; i++)

{

USART\_SEND(desiredSpeed[i]);

\_delay\_ms(10);

}

// get the desired speed from user

desiredSpeedNum[0] = USART\_RECEIVE();

USART\_SEND(desiredSpeedNum[0]);

desiredSpeedNum[1] = USART\_RECEIVE();

USART\_SEND(desiredSpeedNum[1]);

desiredSpeedNum[2] = USART\_RECEIVE();

USART\_SEND(desiredSpeedNum[2]);

USART\_SEND(10);

//set the desired speed

int DSpeed = ((desiredSpeedNum[0]-48)\*100) + ((desiredSpeedNum[1]-48)\*10) + ((desiredSpeedNum[2]-48));

OCR0A = DSpeed;

    while (1)

    {

if( !(PIND & (1<<PD2)) ) // if the button was pushed

{

while(OCR0A > 50)

{

OCR0A -= 10; // slowly decrease the speed of the motor

\_delay\_ms(1000);

} // when the motor speed < 50, then turn the motor off

OCR0A = 0;

}

    }

}

**REFERENCE:**

**ATmega328p Datasheet:** <http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf>

**Hall Sensor Datasheet:** <http://osepp.com/downloads/pdf/10.57.19.pdf>

**L293d Datasheet:** <http://www.ti.com/lit/ds/symlink/l293.pdf>