PID control of DC motor with encoder

Phillip Sortomme

Goal:
Use PID method to drive the speed of a DC motor with encoder connected to an Atmega328p microcontroller. The required speed is set using user input via USART and both required and current speeds are displayed to PC terminal.

Deliverables:
The end goal was to control the speed of the DC motor using the PID method with the desired speed being transmitted via USART by the user (keyboard interface). A soft stop was to be implemented by a specific user command (s<time_in_seconds>). Due to personal time constraints the total project was not completed, however the method for transmitting the instruction was finished

I. Components

A. Component 1

The primary component used was the atmega328p microcontroller. All other components were connected through the microcontroller

B. Component 2

For the motor with encoder I used Actobotics 195 RPM planetary gear motor with encoder. This motor uses 3-12V to power it with 12V being optimal. The motor driver used in connection with this was the L293D half-H driver. This driver was used due to its ability to allow motor control voltages between 4.5V and 36V. so the 12V necessary for the motor used was no issue.

C. Component 3

For the microcontroller to PC USART communication I used the FTDI breakout board. This was chosen because the mini usb was an easy hookup for the design, and also because the board could supply the 5V needed to power the microcontroller.

II. SCHEMATICS

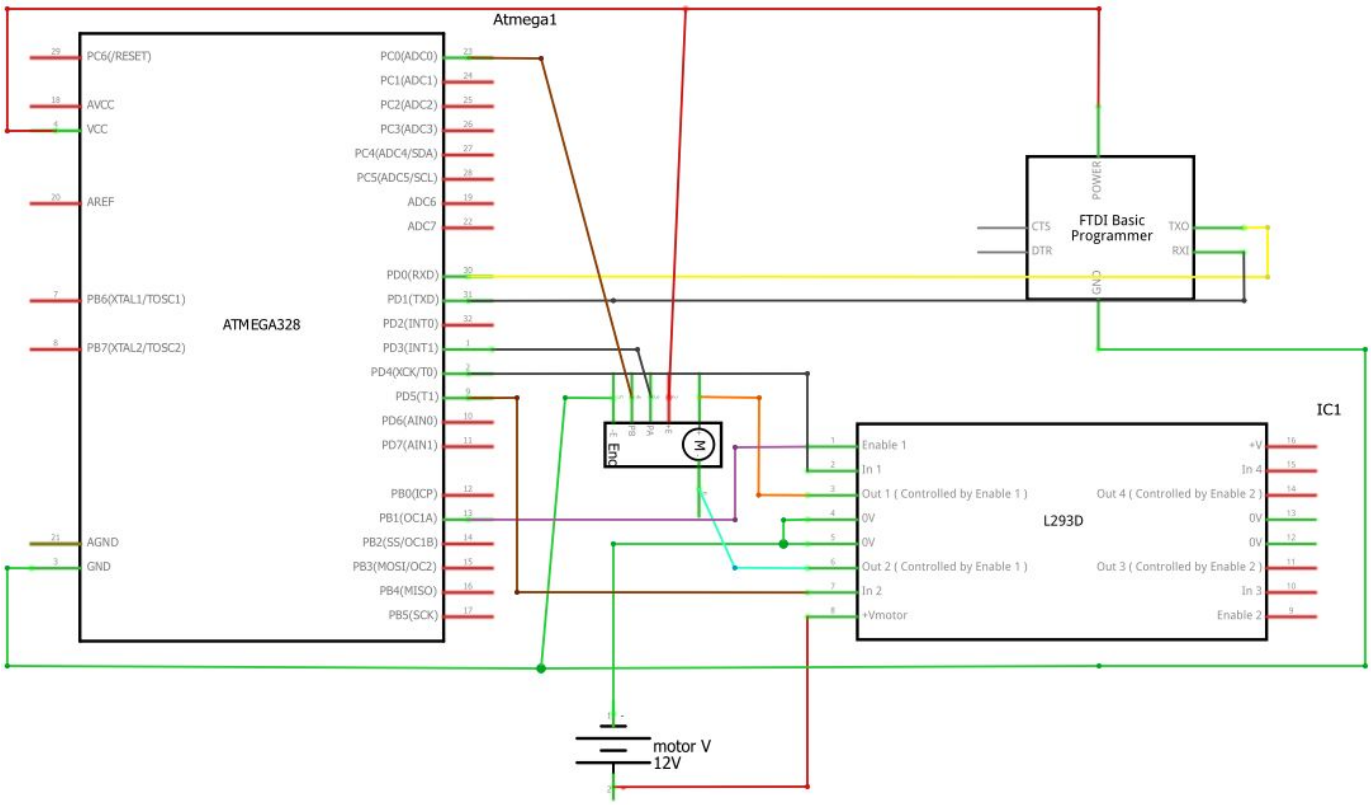

Figure 1: circuit design for motor with encoder

III. IMPLEMENTATION

Using the provided code bits from lectures I hoped to implement the motor control strictly with user input through data received from keyboard by USART communication. I spent a great deal of time working on this aspect of the project, and while I was finally able to finish this portion of the design I was left with little time to get the PID DC motor control design worked out. The goal was to receive required speed values from the user through interrupts then convert that value to the equivalent output for the motor. To get the USART to work properly I needed to use interrupts. This created the problem of how to ensure the entire data was given before it could be used. I found to be most efficiently accomplished by using a circular buffer (which I found the design for [1]), and by checking if a return or newline is given. Because the inputs needed to run the motor were very specific an error would occur if the input was not what was expected

IV. SNAPSHOTS AND

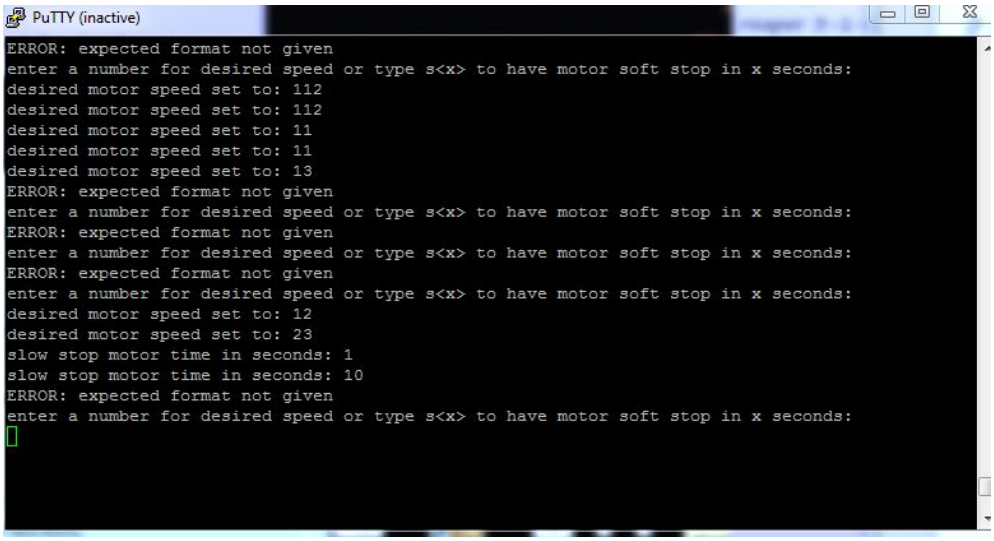

Figure 2: sample output of user input and feedback

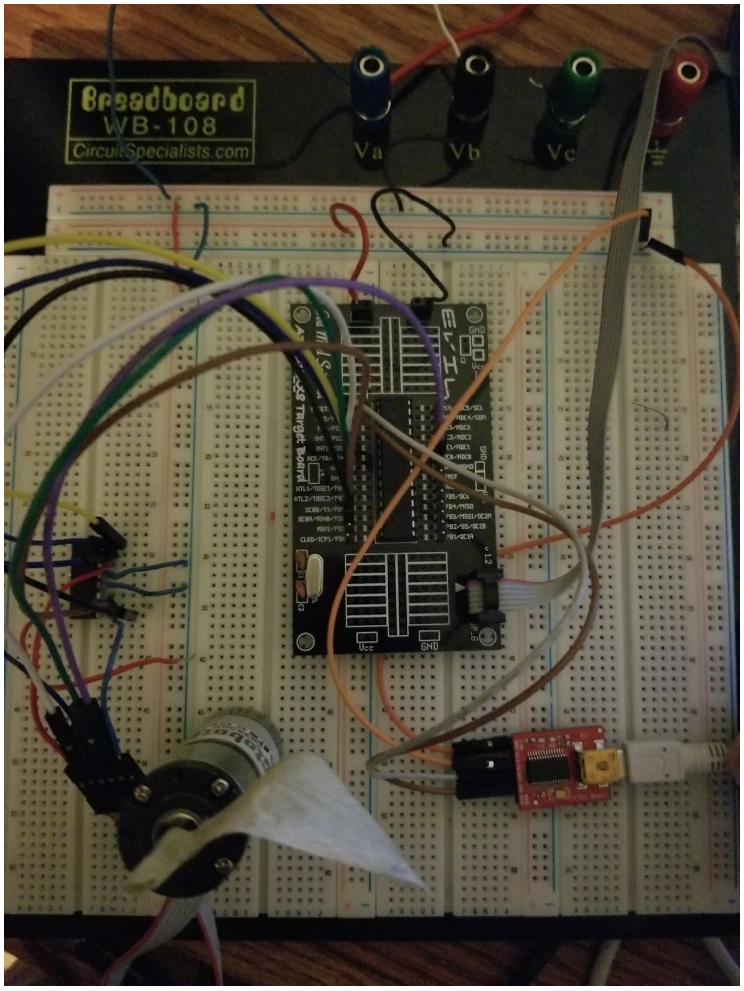

Figure 3: complete circuit implementation

V. Conclusion

Having spent the majority of the time on the user interface to the microcontroller via USART, I believe this could be an effective method of direct control on projects that require user input. The implications of this method of control are even greater if used with a wireless transmitter rather than direct connection through USB.

VI. References

[1] D. Camera, The Lightweight USB Framework for AVRs, (2018), GitHub repository, https://github.com/abcminiuser/lufa/tree/master/LUFA/Drivers/Misc