Introduction

The objective of this lab is to write the necessary NIOS2 software and build the necessary hardware to implement a closed-loop controller to control the rotational speed of a DC motor that includes an integrated encoder. An encoder is a sensor that measures the rotational position or rotational speed of a spinning object; in this case, the internal motor shaft.

Your software should accept the following parameters, defined in the code: 1. **time**: time to run the controller 2. **T_s**: controller time step 3. **k_p**, **k_d**, **k_i**: PID constants 4. **setpoint**: the speed desired for the external motor shaft in RPM

Your controller must be deployed in a standard NIOS2 software project. The motor PWM generator and speed decoder must be deployed a hardware IP component.

Outside the board, you will need to wire up an H-bridge and connect it to the motor. Additionally you will need to use the bench power supply to power the H-bridge and motor.

The Motor

The motor is a 6V DC motor equipped with a 50:1 gear box and a built-in Hall effect encoder that produces 12 pulses per revolution of its internal shaft. This translates to 600 pulses per revolution of the external shaft, due to the 50:1 gear box. The encoder provides two outputs are 180 degrees out of phase from one another as a means to determine the motor rotation direction.

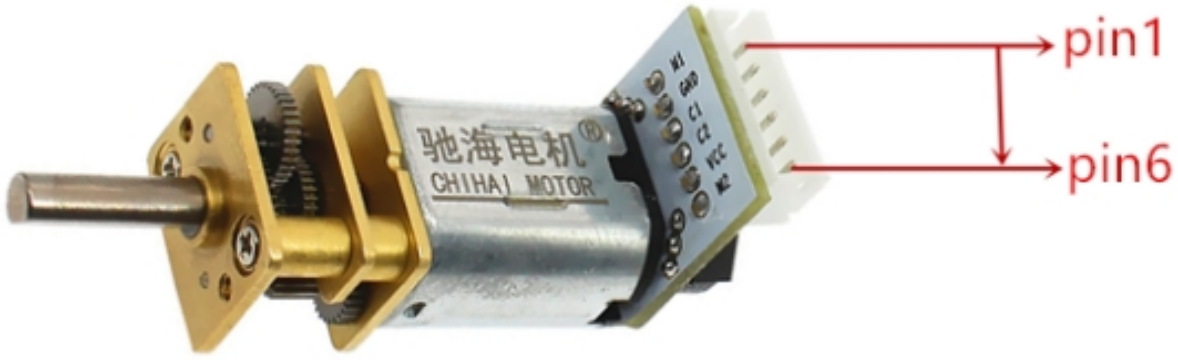

Figure 1: picture of motor

It has six pins: 1. M1 (white wire): motor input 1, positive voltage relative to M2 drives the motor forward, negative voltage drives the motor backward 2. GND (purple wire) 3. C1 (green wire): first encoder input, pulses 12 time per revolution (will be high when C2 is low for forward revolution) 4. C2 (yellow wire): second encoder input, pulses 12 times per revolution (will be high when C1 is low for backward

revolution) 5. VCC (black wire): connect to 6V 6. M2 (red wire): motor input 2, positive voltage relative to M1 drives the motor forward, negative voltage drives the motor backward

The H-Bridge

The H-bridge accepts two PWM signals on pins IA and IB, which each control the voltage delivered to its OA and OB pins. OA connects to motor's M1 pin, OB connect to the motor's M2 pin.

Pin definitions:

No.	Symbol	Function
1	OA	A road output pin
2	VCC	Supply Voltage
3	VCC	Supply Voltage
4	OB	B output pin
5	GND	Ground
6	IA	A road input pin
7	IB	B input pin
8	GND	Ground

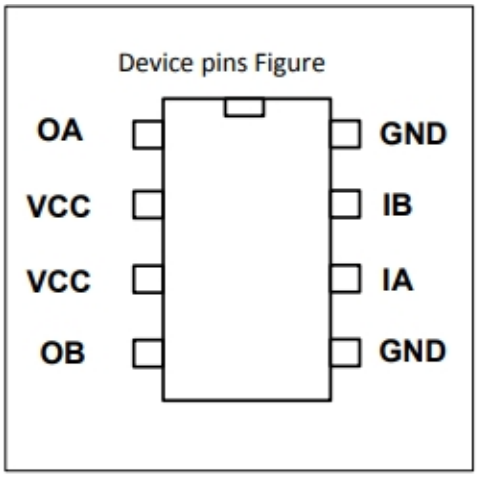

Figure 2: picture of hbridge

The H-bridge schematic is shown below. The IA pin controls switches S1 and S4, while the IB pin controls switches S2 and S3. The left and right side of the M block connect to the M1 and M2 pins of the motor.

This way, when switches S1 and S4 are on, the M1 and M2 pins on the motor will receive a positive voltage. When switches S2 and S3 are on, the M1 and M2 pins will receive a negative voltage.

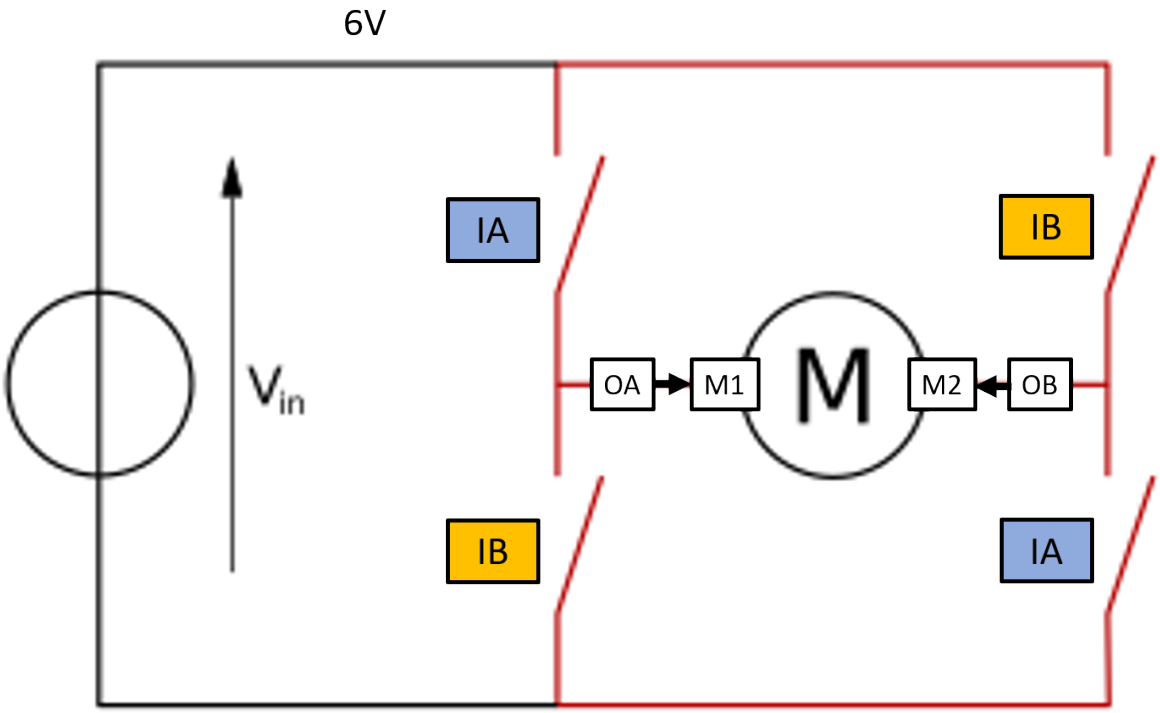

Figure 3: hbridge schematic

A PWM signal on IA will control the forward speed of the motor during which time the IB pin should be low. Likewise, a PWM signal on IB will control the backward speed of the motor during which time the IA pinshould be low..

The Encoder

The motor's encoder will produce a square wave, whose period (rising to rising edge or falling to falling edge) will give its rotational speed.

The speed in revolutions per minute (RPM) is computed as:

```
(1 / period) / GEARING / PULSES_PER_ROTATION * 60,
```

where GEARING = 50 and PULSES_PER_ROTATION = 12.

Re-calculate the period on every rising edge of either C1 or C2. The direction of rotation can be determined by comparing the values of C1 and C2 whenever there is a rising edge on either. If C1 is high while C2 is low then then motor is moving forward, otherwise it is rotating backward.

Pulse Width Modulation

Like in the previous lab, you must use a hardware generated PWM signal. For example, you can use a hardcoded PWM period of 2048 clock cycles, which is approximately 24 KHz.

The Motor IP Component

Create a custom IP for use with the motor.

The IP should export a 2-bit output, pwm_out, and a 2-bit input, encoder_in. In your top-level module, connect pwm_out to GPIO[1:0] and connect encoder_in to GPIO[3:2].

The IP should allow a single value to be written to it in the range of -100 to 100, which specifies the speed. The IP should allow a single value to be read from it, which will return the current speed of the motor in RPMs (which can be positive or negative depending on the rotation direction).

When accepting a new speed value, the IP will need to perform the following: 1. Store the sign of the speed in a "direction" register 2. Take the absolute value of the speed (e.g. `assign abs_speed = speed[7] ? ~speed+8'b1 : speed;`) 3. Rescale "abs_speed" to the PWM counter interval. For this, you can use fixed-point arithmetic and enforce a maximum value of 2047.

Use a 11-bit free-running counter to control the PWM signal. Use logic to route the PWM signal through one of the bits of pwm_out depending on the motor direction.

The IP should count the number of cycles between rising edges of both bits of encoder_in and compute the resultant RPMs. Use fixed-point arithmetic for this calculation. The direction can be determined by the value of encoder_in[1] on the rising ede of encoder_in[0].

The GPIO Connector on the DE2-115

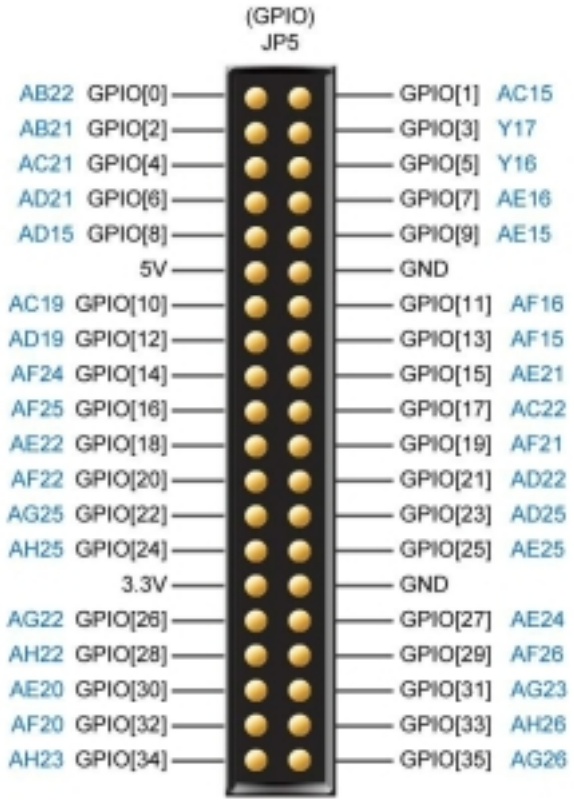

Figure 4: GPIO connector

Steps For Setting Up Your Project on the Bench:

1. Place the H-bridge on the breadboard
2. From the power supply, connect channel 1's positive terminal to the power supply positive row on the breadboard
3. From the power supply, connect channel 1's ground terminal to the ground row on the breadboard
4. From the DE2-115's expansion header, connect the pin on row 6 column 2 to the ground row on the breadboard
5. On the motor, connect VCC to the power supply positive row on the breadboard
6. On the H-bridge, connect pins 2 and 3 (middle pins, bottom row) to the power supply positive row on the breadboard
7. On the motor, connect GND to the ground row on the breadboard
8. On the H-bridge, connect pins 5 and 8 (outer pins, top row) to the ground row on the breadboard

9. From the DE2-115's expansion header, connect pins 0 and 1 to pins 6 and 7 (middle pins, top row) on the H-bridge
10. From the DE2-115's expansion header, connect pins 2 and 3 to the C1 and C2 on the motor
11. From the H-bridge, connect pins 1 and 4 (outer pins, bottom row) to the M1 and M2 pins on the motor

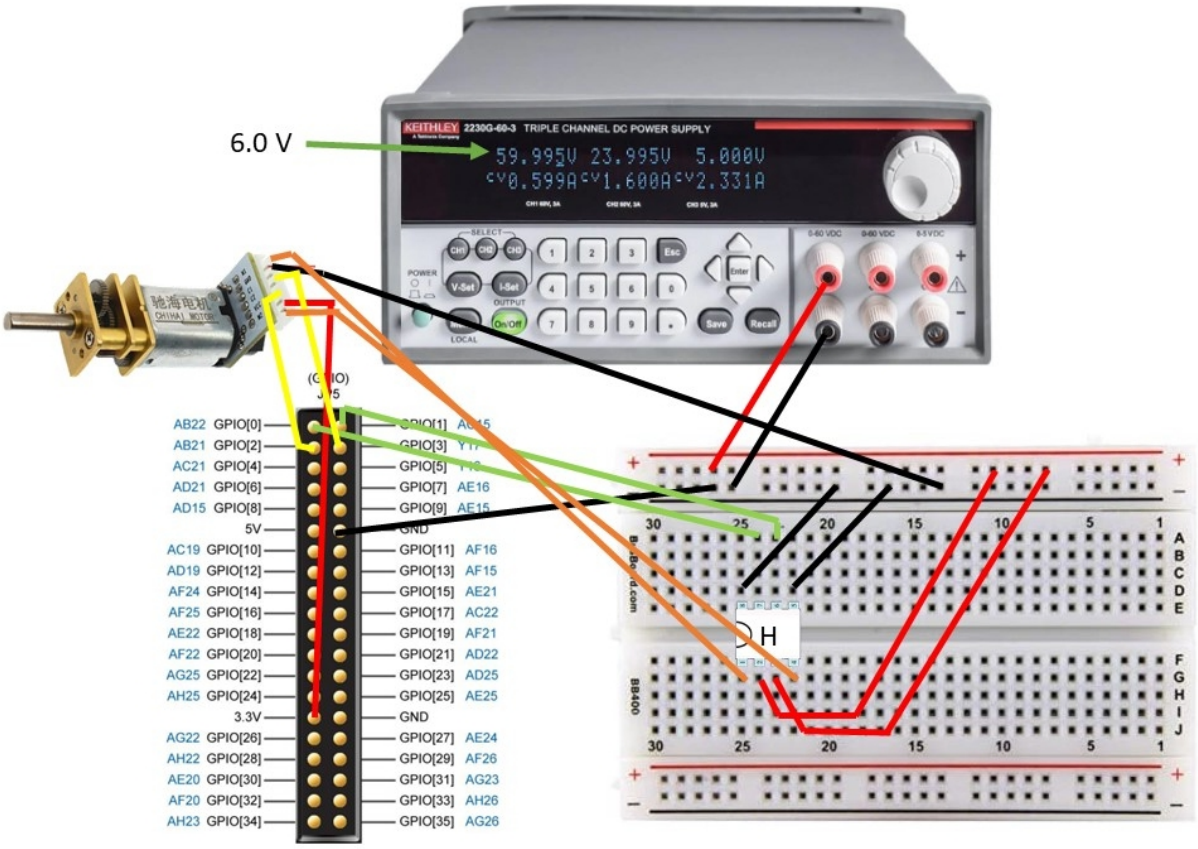

Figure 5: Overall schematic

Using the Power Supply

Each bench station is equipped with a Keithley 2230G-30 195-Watt DC power supply with 3 channels that can supply up to 6 Amps or 60 Volts per output. For this project, we need a 6 V supply, with the motor requiring no more than 100 mA of current.

To set up the power supply, follow these steps:

1. Turn the power supply on by pushing the green power button; note that this does not energize the output channels, as they will by OFF initially,

2. press the “CH1” button (assuming you want to use channel 1, although you can alternatively use channel 2),
3. press the V-Set button,
4. type “6.00” and press the “Enter” button, which is located in the center of the directional keys,
5. after approximately 3 seconds, the display will show “0.00 V” for the channel you just set, meaning that the voltage seen on that channel is currently zero. This is because the outputs are not turned on and there is no external voltage connected to the output channels,
6. when you are ready to activate the output channel, press the “(Output) On/Off” button, and then
7. watch the channel 1 voltage output, which should now show 6 Volts.

Requirements

Your Nios2 software should both control the motor and monitor its rotational speed and direction. The software should log the value of time, motor input, and motor speed and direction at **1/T_s** Hz sampling rate, and then dump this log to the terminal before the program ends.

The motor input should be represented as a duty cycle from -100 to 100 where positive values represent a duty cycle on the H-bridge’s IA pin with the IB pin set to 0 and negative values represent a duty cycle on the H-bridge’s IB pin with IA pin set to 0.

The motor speed should be in revolutions per minute (RPM), with positive values representing forward rotation and negative values representing backward rotation. You may use your own definition of “forward” and “backward”.

The program should run for **time** seconds, during which time it should cycle through the entire input signal.

The relationshop between the motor input and speed, given by the encoder, is shown below:

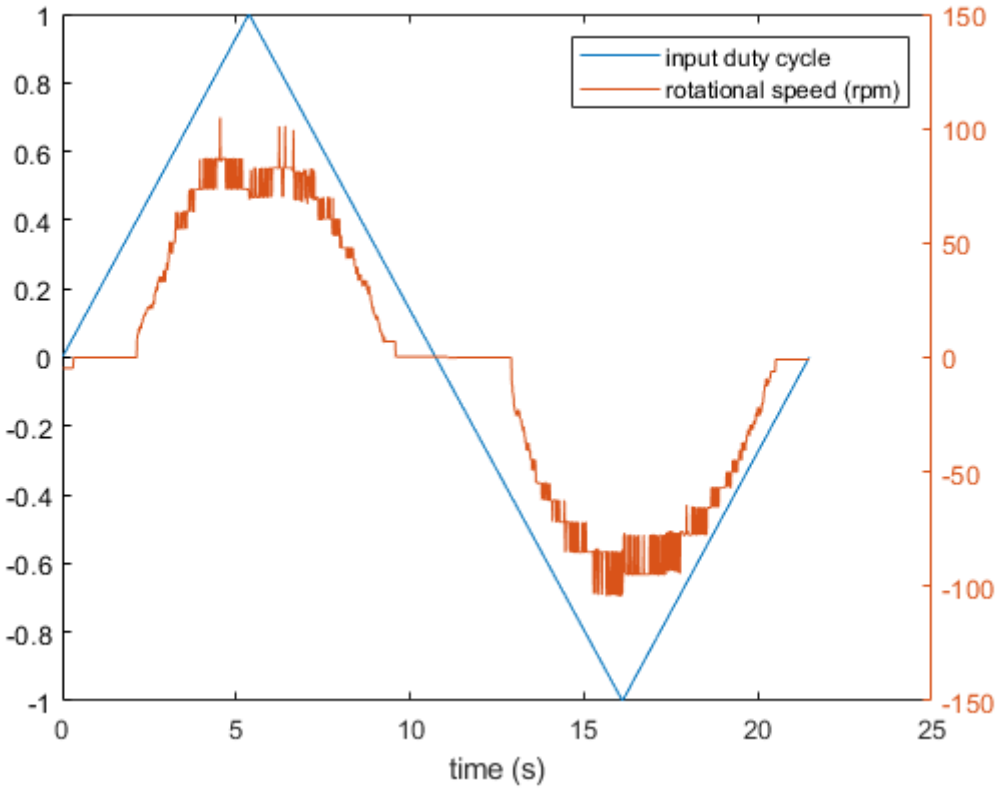

Figure 6: Overall schematic

To implement your controller, you will need to choose values of **k_p**, **k_d**, and **k_i** to force the motor speed to the setpoint. Note that values of 0 are valid for any of these parameters.

Controller Algorithm

Your controller must be a PID controller and should perform control at a period defined by **T_s**. The error derivative and integral calculations may also be performed at this same period.

```
reset and start cycle counter

while True:

    // determine if we should perform control
    cycle_counter = READ_CYCLE_COUNTER
    cycles_since_last_control = cycle_counter -
        prev_contol_action_cycles
```

```
    time_since_last_control = cycles_since_last_control / CPU_FREQ
    time_since_start = cycle_counter / CPU_FREQ

    // perform control
    if time_since_last_control >= T_s:
        current_motor_rpm = READ_MOTOR_RPM

        error_old = error
        error = current_motor_rpm - setpoint
        error_accum += error
        error_delta = error-error_old

        motor_input = k_p * error + k_d * error_delta + k_i *
            error_accum

        log time_since_start
        log motor_input
        log current_motor_rpm

        prev_contol_action_cycles = cycle_counter

        if time_since_start > time:
            break
```

Submitting Your Code

When you are ready to turn in your code for grading, each group should submit an archive of their project directory to Dropbox. To do this, enter the following commands:

```
rm -Rf <project_dir>/db <project_dir>/incremental_db <project_dir>/
    output_files
tar cvf lab4.tar <project_dir>
gzip lab4.tar
```

Rubric

- Bench setup = **20 points**
- Software interface to the DC motor = **20 points**
- Hardware interface for motor = **20 points**

- Controller design = **20 points**
- Correct operation = **10 points**
- Data logging = **10 points**

Total points possible: 100.

