Aleida Diaz-Roque

Elkaim

ECE 118: Intro to Mechatronics

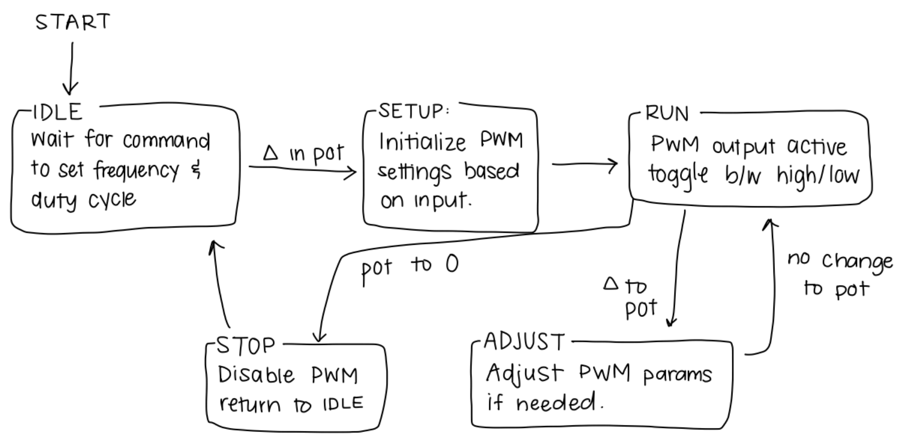
28 April 2018

Lab 3 - Motors

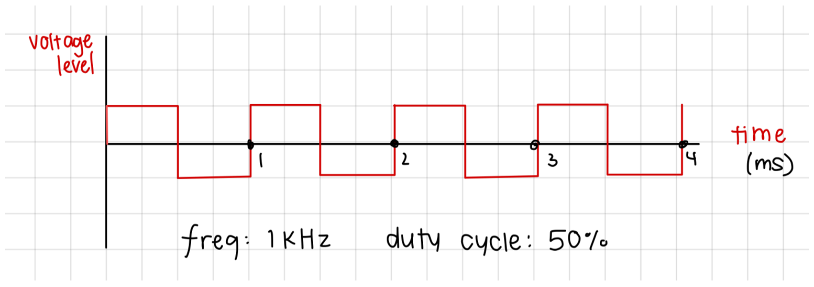
Part 0 - Preparation for Lab

The only thing you should stick into the 14-pin connectors on the boards are the ribbon cables or appropriate connector cables that are designed to fit these connectors.

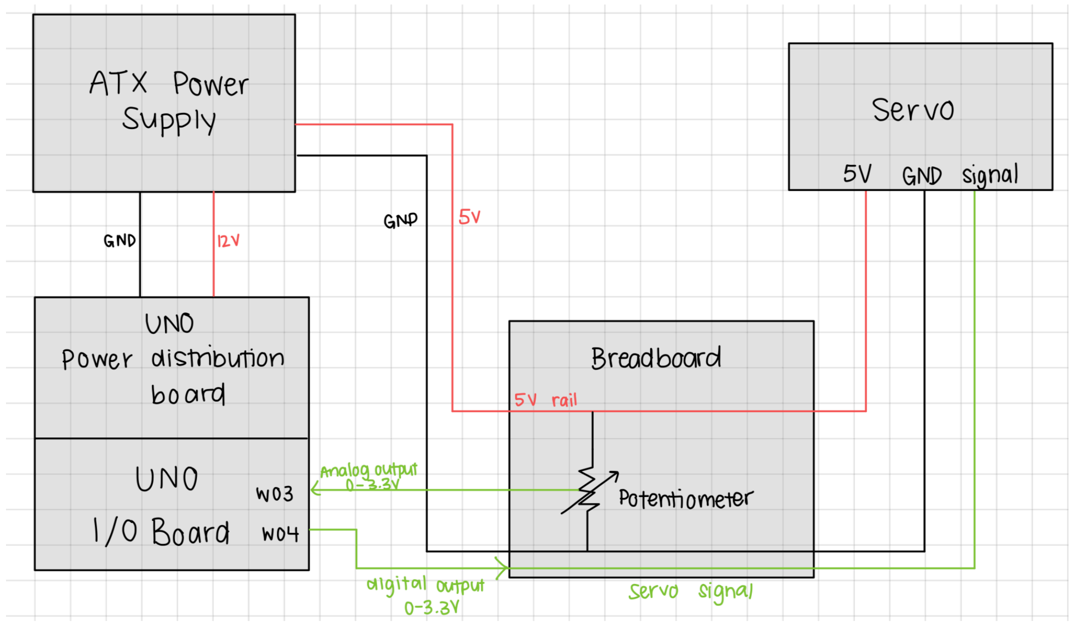
The following is an idea for a PWM driver.


START
IDLE
Wait for command to set frequency & duty cycle
Δ in pot
SETUP:
Initialize PWM settings based on input.
RUN
PWM output active toggle b/w high/low
pot to 0
Δ to pot
no change to pot
STOP
Disable PWM return to IDLE
ADJUST
Adjust PWM params if needed.

Below is an example plot of the expected output signal for a given frequency of 1 kHz and 50% duty cycle.



Part 1 - Driving and RC Servo



Connections:

Uno32 to RC Servo: PWM Output Pin to Servo Signal Input.

Potentiometer to Uno32: One terminal to 3.3V on Uno32 or the 5V rail from power supply.

Wiper (middle terminal) to an analog input on Uno32. Other terminal to Ground (GND).

Power Connections: 5V from Power Distribution Board to Servo Power Input.

Ground from Power Distribution Board to Servo Ground.

Choosing ports:

PWM Output Port is chosen because it is capable of generating PWM signals which are necessary to control the RC servo. The Analog Input Port is needed to read the varying voltage from the potentiometer which translates to position control of the servo.

Initialization code:

```
PWM_Init();
BOARD_Init();
LED_Init();
RC_Init();
```

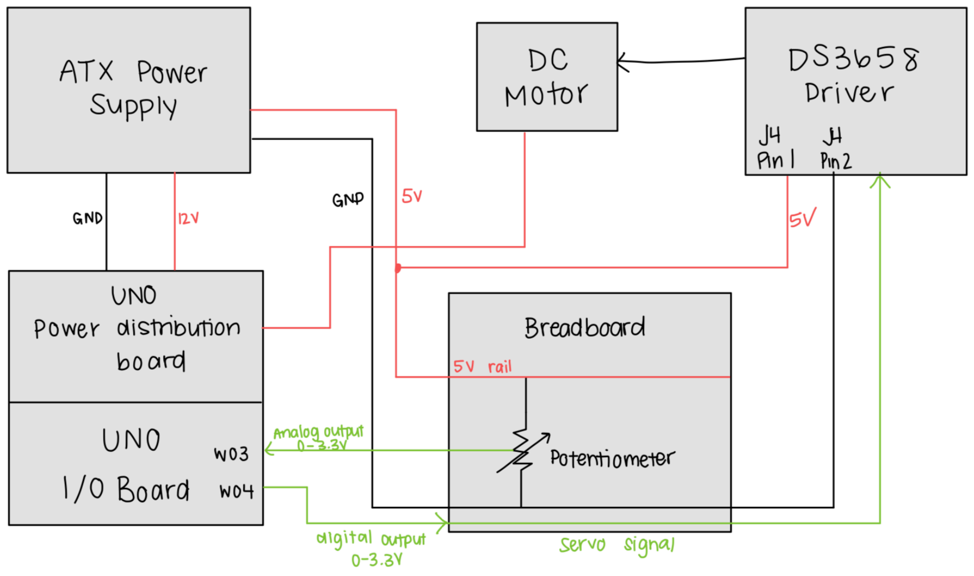
Type of signals:

PWM Signal: Uno32 to RC Servo (Digital Output).

Analog Signal: Potentiometer to Uno32 (Analog Input).

Power: 5V and Ground to RC Servo (Power).

Part 2 - Unidirectional Drive of an DC Motor



Connections:

<u>Uno32 to DS3658:</u> PWM output from a digital pin of the Uno32 to the input of the DS3658.

<u>DS3658 to DC Motor:</u> Connect the output from the DS3658 to one terminal of the DC motor, with the other terminal connected to ground or the motor's power supply as required.

<u>Potentiometer to Uno32:</u> One terminal to 3.3V on Uno32. Wiper (middle terminal) to an analog input on Uno32 (e.g., A0). Other terminal to Ground (GND).

<u>Power Connections:</u> Power the motor and the DS3658 from the power distribution board, ensuring correct voltage and current capabilities.

Choosing Ports:

PWM Output Port is chosen because it is capable of generating PWM signals which are necessary to control the motor speed via the DS3658. The Analog Input Port is necessary to read the varying voltage from the potentiometer which translates to speed control of the motor.

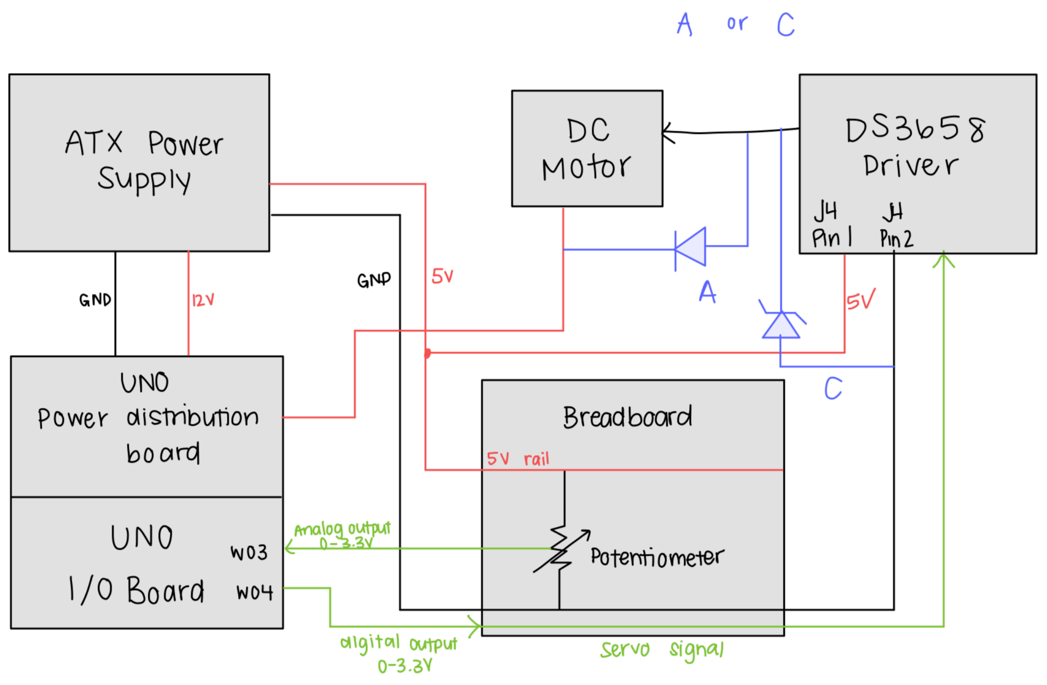
Initialization Code:

```
// Initialize the PWM library for the motor control pin
    PWM_Init(MOTOR_PWM_PIN);
    PWM_SetFrequency(MOTOR_PWM_PIN, 1000);  // Set a suitable
frequency for motor control

    // Initialize the AD library and add the potentiometer
channel
    AD_Init();
    AD_AddChannel(POT_PIN);

    BOARD_Init();
```

Part 3 - Snubbing the Inductive Kickback



Connections:

Uno32 to DS3658: PWM signal to control motor operation.

DS3658 to DC Motor: Connects the motor to the driver; snubbing diodes placed across the motor terminals.

Potentiometer to Uno32: Analog signal to determine motor speed.

Power Connections: Ensure DS3658 and motor are powered appropriately, with voltage levels suitable for the motor and within the specs of the DS3658.

Choosing ports:

PWM Output Port is suitable for generating control signals to the motor driver. The Analog Input Port is necessary for reading the voltage variation from the potentiometer which correlates to motor speed.

Initialization code:

```
    PWM_Init(MOTOR_PWM_PIN);  // Initialize PWM on specified
pin
    PWM_SetFrequency(MOTOR_PWM_PIN, 1000);  // Set PWM
frequency for motor control

    AD_Init();                // Initialize ADC module
    AD_AddChannel(POT_PIN);   // Add potentiometer pin to ADC
monitoring
    BOARD_Init();
```

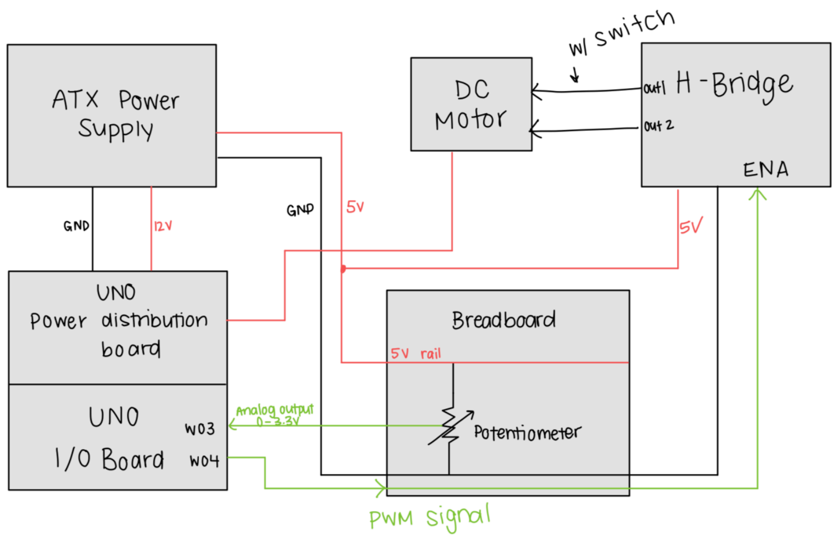
Type of signals:

PWM Signal (Digital Output): Controls the DS3658, thus regulating motor speed.

Power Signals (High-Current Output): Powers the motor, potentially modified by snubbing diodes to manage voltage spikes.

Analog Signal (Analog Input): Translates user input via a potentiometer into motor speed control.

Part 4 - Bidirectional control of a DC Motor



Connections:

<u>Uno32 to H-Bridge:</u> PWM output for speed control and digital outputs for direction control.

<u>H-Bridge to DC Motor:</u> Connects to the motor; controls which direction the motor shaft turns.

<u>Potentiometer to Uno32:</u> Analog input used to determine speed.

<u>Switch to Uno32:</u> Digital input used to set direction.

<u>Power Connections:</u> Appropriate power to the H-Bridge and motor.

Choosing Ports:

PWM Output Port is essential for adjusting motor speed via the H-bridge. Digital output ports for direction are needed for changing the polarity of voltage applied to the motor, thus changing its direction. The Analog Input Port reads voltage variation from the potentiometer which correlates to desired motor speed.

Initialization Code:

```
#include libraries
```

```
    PWM_Init(MOTOR_SPEED_PIN);
    PWM_SetFrequency(MOTOR_SPEED_PIN, 1000);  // Set
appropriate frequency

    // Initialize digital pins for direction control
    pinMode(MOTOR_DIR_PIN_1, OUTPUT);
    pinMode(MOTOR_DIR_PIN_2, OUTPUT);

    // Initialize the ADC for the potentiometer
    AD_Init();
    AD_AddChannel(POT_PIN);

    // Setup switch pin as input
    pinMode(SWITCH_PIN, INPUT_PULLUP);
```

Type of Signal:

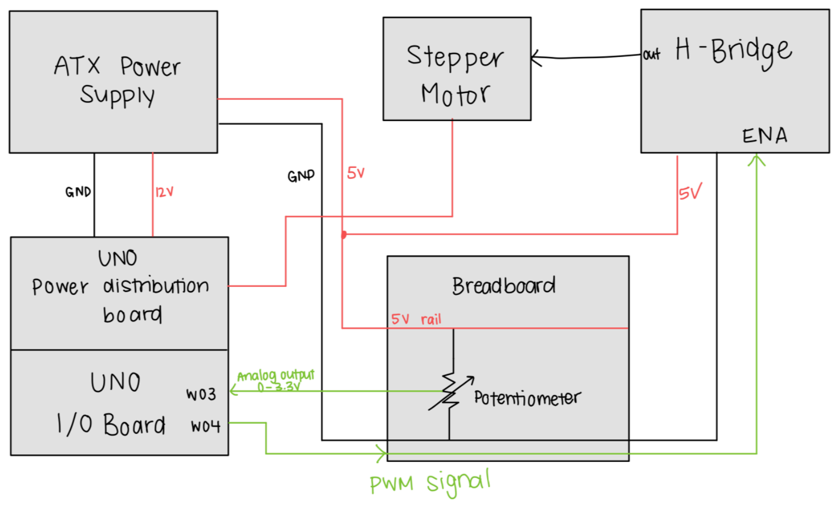
PWM Signal (Digital Output): Controls the motor speed via the H-bridge.

Directional Control Signals (Digital Outputs): Determines the direction of the motor by changing the polarity of the voltage applied to the motor's terminals.

Analog Signal (Analog Input): Translates user input via a potentiometer into motor speed.

Directional Switch Signal (Digital Input): Determines the direction of the motor's rotation based on the switch's position.

Part 5 - Control of a Stepper Motor



Connections:

<u>Uno32 to H-Bridge:</u> Digital outputs for controlling the H-Bridge, which in turn controls the current in the stepper motor coils.

<u>H-Bridge to Stepper Motor:</u> Connects to the motor; controls the activation of coils in sequence for motion.

<u>Power Connections:</u> Adequate power supplied to the H-Bridge and motor.

Choosing Ports:

Digital Output Ports are needed for sending precise control signals to the H-Bridge to drive the stepper motor according to the selected stepping mode.

Initialization code:

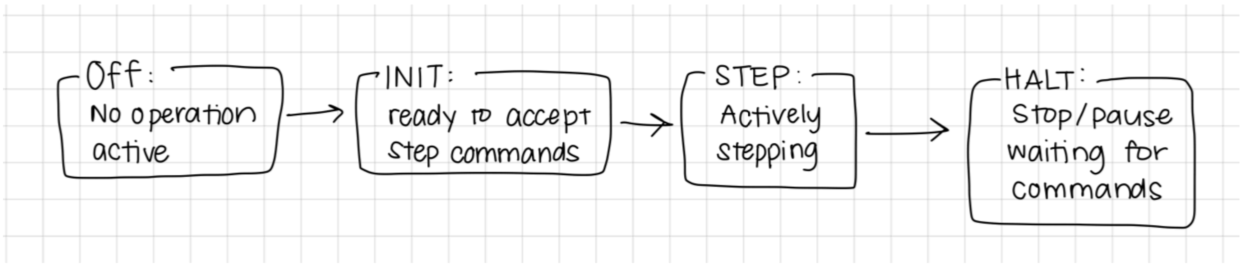
```
    pinMode(STEP_PIN_1, OUTPUT);
    pinMode(STEP_PIN_2, OUTPUT);
    pinMode(STEP_PIN_3, OUTPUT);
    pinMode(STEP_PIN_4, OUTPUT);

    // Initialize stepper control (if library requires
initialization)
```

```
    Stepper_Init(STEP_PIN_1, STEP_PIN_2, STEP_PIN_3,
STEP_PIN_4);
```

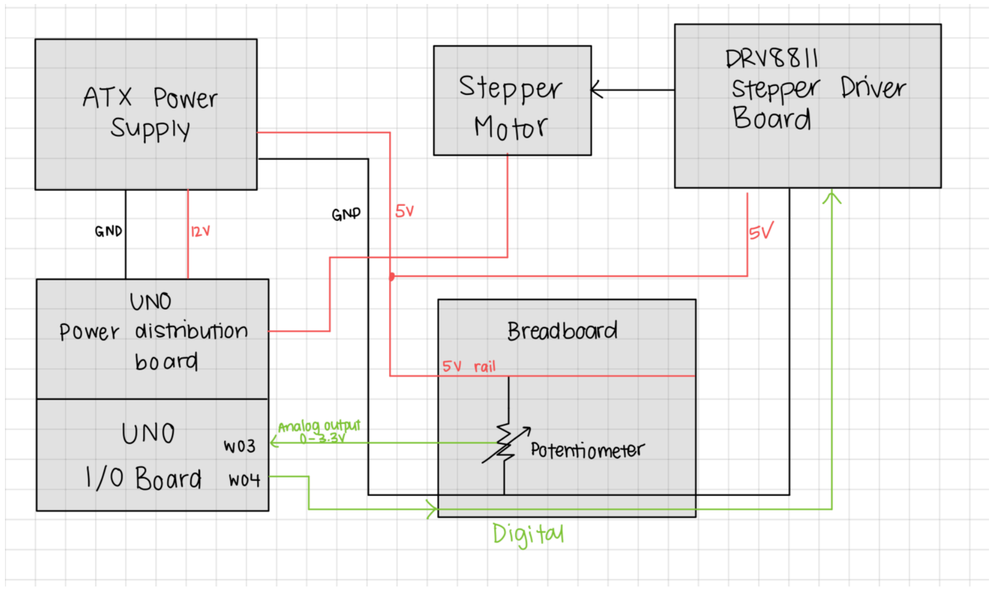
Type of Signal:

Digital Control Signals (Digital Outputs) control the H-Bridge inputs to alternate the direction of current through the stepper motor's coils, causing the motor to step.



Off: No operation active
INIT: ready to accept step commands
STEP: Actively stepping
HALT: Stop/pause waiting for commands

Part 6 - Stepper Motor using dedicated board



Connections:

<u>Uno32 to DRV8811 Driver Board:</u> Digital outputs from Uno32 for step commands, direction control, and possibly other signals like enable/disable.

DRV8811 to Stepper Motor: Connects to the motor to control its phases based on the driver's logic.

Power Connections: Ensure the driver and motor are adequately powered, following the specifications of both the motor and driver board.

Choosing Ports:

Digital Output Ports for Stepping and Direction are needed for sending step and direction commands to the driver board. Other control signals include enable, fault, and mode selection (for micro-stepping configurations).

Initialization code:

```
    pinMode(STEP_PIN, OUTPUT);   // Set stepping pin as output
    pinMode(DIR_PIN, OUTPUT);    // Set direction pin as output
    pinMode(ENABLE_PIN, OUTPUT); // Set enable pin as output

    digitalWrite(ENABLE_PIN, HIGH); // Enable the driver

    // Initialize the stepper driver if necessary
    Stepper_Init();
```

Type of Signal:

Step Signal (Digital Output): Triggers a step action in the stepper motor.

Direction Signal (Digital Output): Determines the direction of the stepper motion.

Enable Signal (Digital Output): Enables or disables the driver circuitry