

D/A Converters

A/D converters allow a PIC to read analog data, such as information from a temperature sensor, light sensor, etc.

D/A converters allow a PIC to output analog data. With this, you can

- Output a sine wave to drive a speaker (rather than a square wave)
- Output other periodic waveforms, creating an electronic cello, violin, tuba, etc
- Control the speed of a DC motor
- Control the brightness of an LED, and
- Control the speed of an AC motor to name a few.

While the PIC18F4620 has a built-in A/D converter, it does not have a D/A. Likewise, you need to add an external D/A. That's why your lab kits include a MCP4921 chip

D/A with a PIC Microcontroller

The hardware to connect the MCP4921 to your PIC board is as follows:

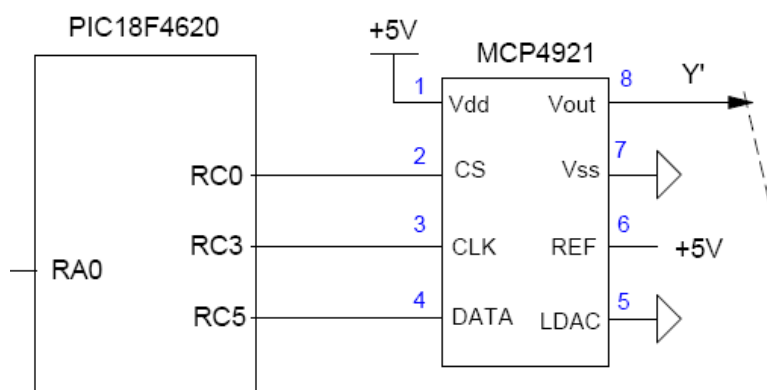


Figure 1: Connections from a PIC microcontroller to a MCP4921 D/A converter

To output a voltage to the MCP4921, 16-bits of data are sent

- First, Chip Select (CS) is pulled low, signifying the start of a message
- The first four bits are 0011 (binary 3). This sets the output to 0% to 100% of Ref.
- The next twelve bits are the number you want to convert
 - 0 (0x000) outputs 0V
 - 4095 (0xFFF) outputs 5V
- The clock line pulses high and low for each bit
- Once all 16 bits are sent, CS goes high again.

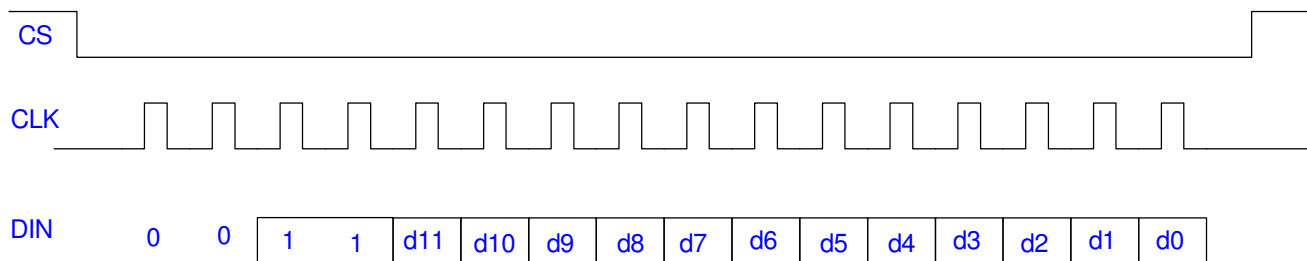


Figure 2: Timing for a MPC4921. CS going low starts the communications, high terminates it

A subroutine to drive this chip follows:

```
void D2A(unsigned int X)
{
    unsigned char i;

    TRISC0 = 0;
    TRISC3 = 0;
    TRISC5 = 0;

    // Add 0011 to the first four bits to set up the D/A

    X = X & 0x0FFF;
    X = X + 0x3000;

    RC0 = 1;
    RC3 = 1;

    // CS goes low to select the D/A chip
    RC0 = 0;

    // Send out 16 bits of data
    for (i=0; i<16; i++) {
        if (X & 0x8000) RC5 = 1; else RC5 = 0;
        RC3 = 0;
        X = X << 1;
        RC3 = 1;
    }

    // CS goes high to terminate the communicaitons
    RC0 = 1;
}
```

Electronic Sine Wave Generator

So far, we've been driving a speaker with a square wave when making an electronic piano. This results in a harsh buzzing sound. For a clean sound, output a sine wave. To do this, first, approximate a sine wave with N data points.

Assume for example we want the sine wave to go from -2V to +2V. Using an instrumentation amplifier

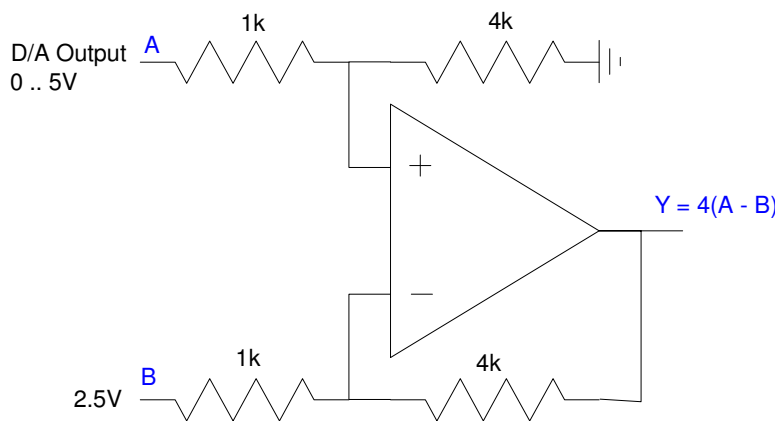


Figure 3: Instrumentation Amplifier to convert 0..5V to -10 .. +10V

you get:

Number to D/A	D/A Output	Circuit Output
0	0V	-10V
4,095	5V	+10V

Table 1:0-4095 map to -10V to +10V analog

The output voltage is related to the number you send to the D/A as

$$\text{Volts} = \left(\frac{D/A}{4095} \right) \cdot 20V - 10V$$

- To output -2V, you need to send the number 1638
- To output +2V, you need to send the number 2457

Let's use 16 points to approximate a sine wave. Use Matlab to generate a sine wave which goes from 1638 to 2457

```
-->t = [1:16]' / 16 * 2 * pi;
-->y = sin(t) * 409 + 2047;
-->y = round(y);
```

```
2204.
2336.
2425.
2456.
2425.
2336.
2204.
2047.
1890.
```

1758.
1669.
1638.
1669.
1758.
1890.
2047.

This looks like the following when you send these voltages to the D/A

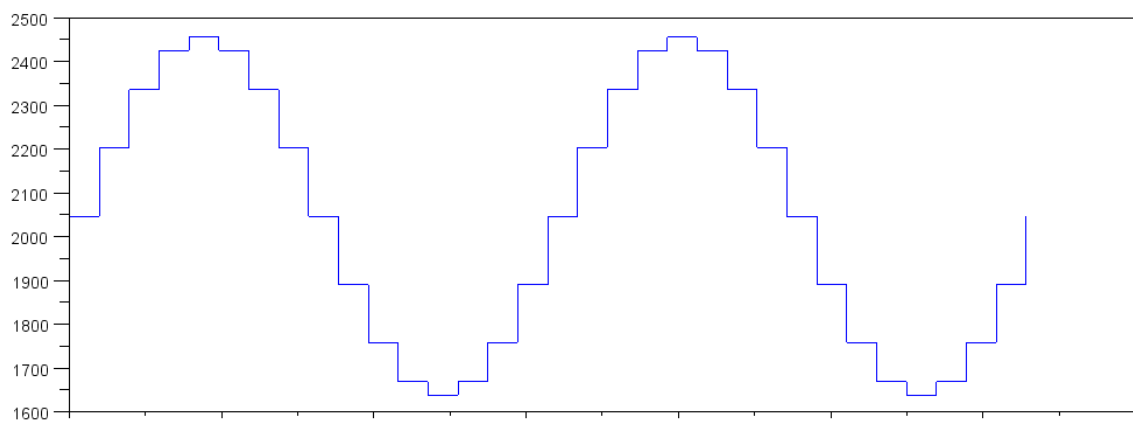


Figure 4: 16-Point approximation to a sine wave

It's not a perfect sine wave. To do better, more points would help - but more points means the PIC will have to output numbers to the D/A that much faster.

To play a 220Hz sine wave,

One cycle = $1/220$ second (4.54ms)

There are 16 D/A calls per cycle (we're representing a sine wave with 16 points)

Each D/A call = 286.1us

From before, when we wrote a wait routine, we found that counting to 620 with an integer takes about 1ms. 286.1us should require counting to 177

$$N = \left(\frac{286.1\mu s}{1ms} \right) 620 = 177$$

The following code should generate a 220Hz sine wave:

```
const unsigned int TABLE[16] = [2204, 2336, 2425, 2456, 2425, 2336, 2204, 2047, 1890,
1758, 1669, 1638, 1669, 1758, 1890, 2047];

// in the main routine
unsigned int i, j;

while(1) {
    i = (i + 1) % 16;
    D2A(TABLE[i]);
    for(j=0; j<177; j++);
}
```

Electronic Tuba

Note that if you change the look-up table, you change the signal sent to the speaker. The signal determines what type of instrument you're mimicking:

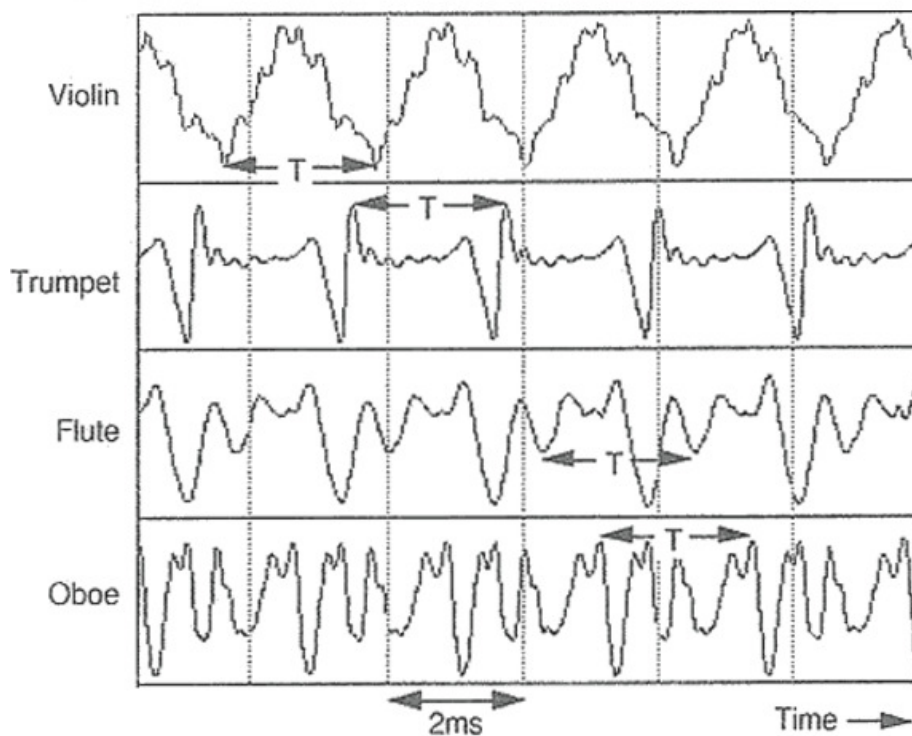


Figure 5: http://www.feilding.net/sfuad/musi3012-01/html/lectures/009_hearing_IV.htm

Analog Voltage Output:

- Control the speed of a DC servo motor (voltage = speed)

With a D/A, a PIC can output 0..5V with 4095 steps. With this, you can drive a DC motor at 0..20V with 4095 steps.

Hardware:

- To amplify the voltage 4x (0..5V becomes 0..20V), use a non-inverting amplifier.
- To amplify the current, use a power transistor (TIP112)

With negative feedback, V_+ equals V_- . For this to happen, the output must be 4x the input.

The op-amp's output is whatever it takes to make V_+ equal to V_- . With the 1.4V drop across the transistor's base to emitter junction, this results in the op-amp outputting 4 times the input, plus 1.4V (not that we really care - but it does).

The transistor provides a current gain of 1000. This means that to drive a load of 3A, the op-amp only needs to source 3mA - something an LM833 is capable of doing.

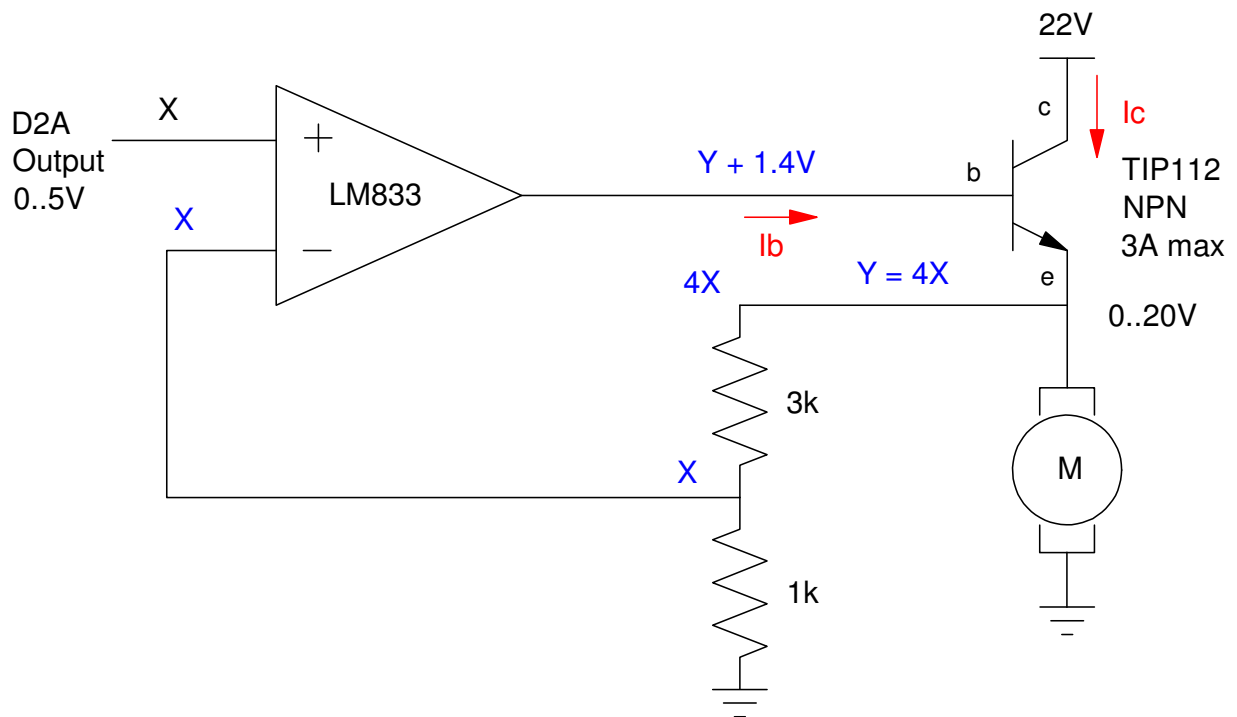


Figure 6: Schematic for driving a 20V motor at up to 3A

With this circuit, you can vary the voltage to the motor (and hence the speed) from 0V to 20V with 4095 steps using a PIC microcontroller.

Analog Current Output:

- Control the brightness of an LED (current = lumens)
- Control the torque of a DC servo motor (current = torque)

A slight variation on this circuit is a current amplifier. Again, an op-amp with negative feedback to force V_+ to equal V_- . To control current, place a resistor in series with the diode. Add a resistor to set the maximum current (100mA) when X is its maximum voltage (5V)

$$R = \frac{5V}{100mA} = 50\Omega$$

Add a transistor to amplify the current from the op-amp. Using a TIP112 (with a current gain of 1000) results in the op-amp having to output 0.1mA when driving 100mA to the load - something an LM833 is capable of doing.

Place the diode in series with the 50 Ohm resistor so that any current going through R also goes through the diode (almost: the base current splits of 1/1000th of the current to the 50 ohm resistor).

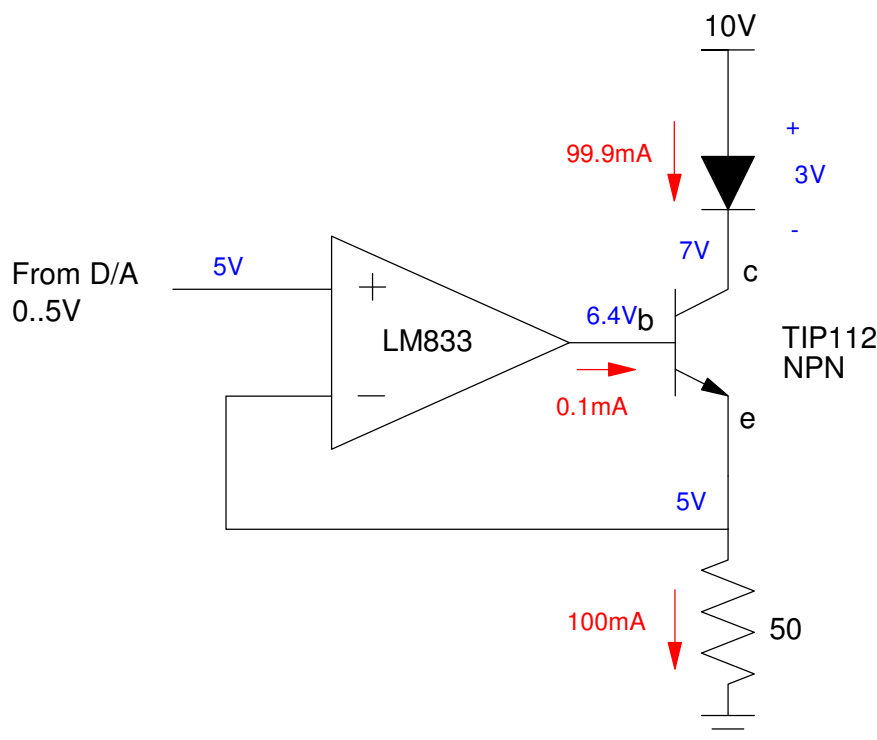


Figure 7: Schematic for driving 0-100mA from a 0-5V analog voltage

With this circuit, you can control the current to the LED from 0 to 100mA with 4095 steps.

Also note: if you replace the diode with a DC motor, you're controlling the current to the motor (torque) rather than voltage (speed).

Also also: Back in EE 206, you used current sources. That's what this circuit is: the current to the load is set by the voltage at X .

DC Servo Motor Control:

Suppose you want to control the speed of a DC servo motor. The first thing you need is a power amplifier. Two options are

- OPA2544 Dual Op-Amp (\$22 from Digikey)
- Advanced Motion Controls 30A8T (\$40 on ebay)

Option 1: OPA2544: This chip is a dual op-amp capable of +/- 35V and 4A.

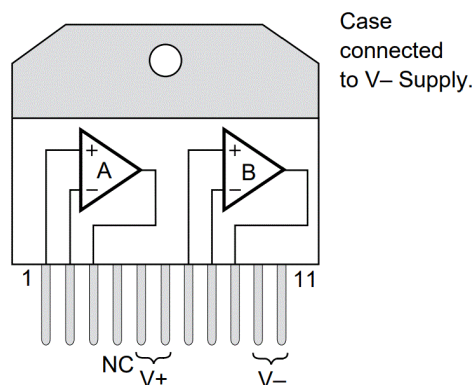


Figure 8: Pin-out for an OPA2544 power op-amp

To connect to a DC servo motor, use the following circuit

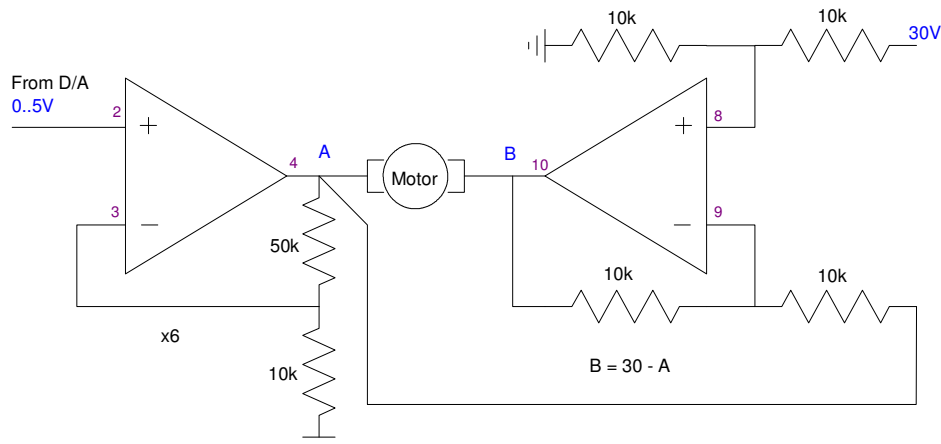


Figure 9: Schematic for driving a motor at +/-30V at 3A using the 0-5V from a D/A

As the D/A outputs 0..5V, the voltage V_{ab} goes from -30V to +30V driving the speed of the motor from full reverse to full forward.

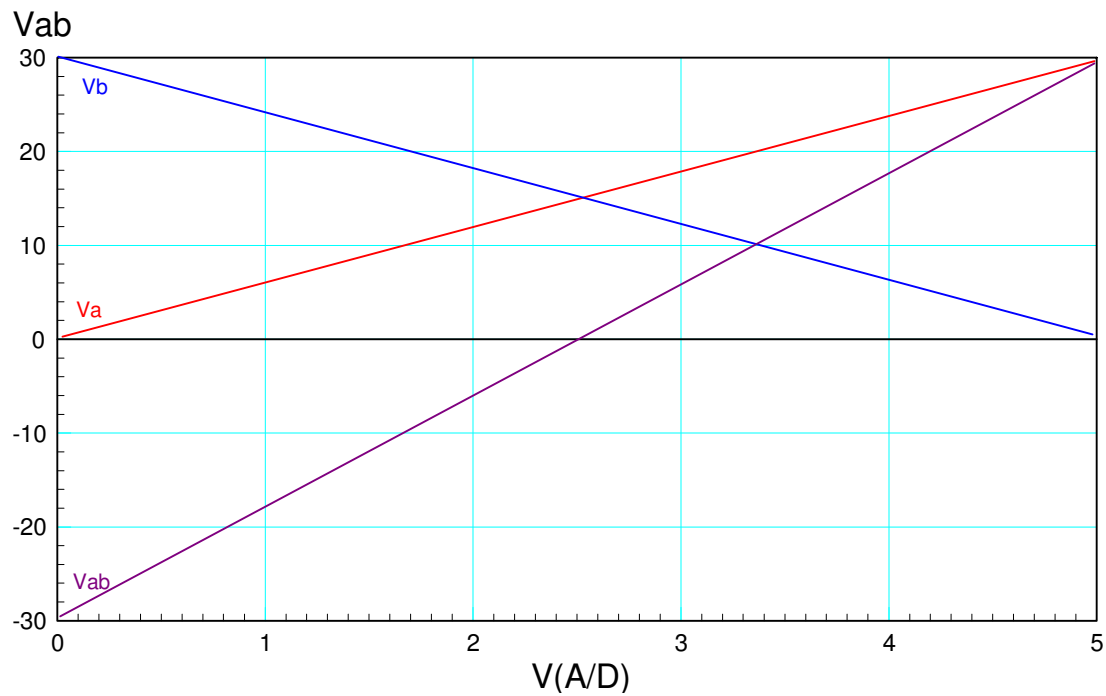


Figure 10: V_a and V_b are limited to 0-30V. V_{ab} goes -30V to +30V however.

Note the following:

You only need a +30V supply to drive the motor at -30V to +30V analog

- Neither V_a nor V_b is ground. They vary from 0 .. +30V such that the difference (V_{ab}) goes from -30V to +30V.
- Don't tie either V_a or V_b to ground. The op-amp won't like that.

Option #2: AMD 30A8V (better option)



Advanced Motion Controls 30A8V Brush Type PWM Servo Amplifier

★★★★★ Be the first to [write a review](#).

Condition: Used

Quantity: More than 10 available
79 sold / [See feedback](#)

Price: **US \$39.99**

[Buy It Now](#)

[Add to cart](#)

Figure 11: ebay: Advanced Motion Controls 30A8V. Typical price on ebay os \$25 to \$100 each

Another option is to use an Advanced Motion Controls 30A8V. This is really the same thing as an OPA2544, but better

- Voltage range: +20V to +80V
- Current Limit: Up to 20A
- Higher efficiency (data sheets say 90%)
- Protection circuitry (doesn't die as easily)

Just connect

- REF+ to the D/A output (0..5V)
- REF- to 2.5V
- Adjust the gain and offset so that you get +30V at+5V, -30V at -5V

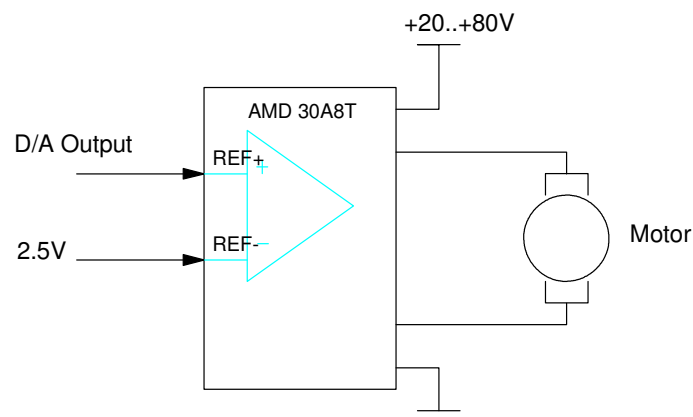


Figure 12: Hardware connections for an AMD 30A8T

Brushless DC Motor Controller (BLDC)

Similarly, if you want to drive a 3-phase AC synchronous motor, you need a variable frequency drive. These are available on ebay for about \$10 each

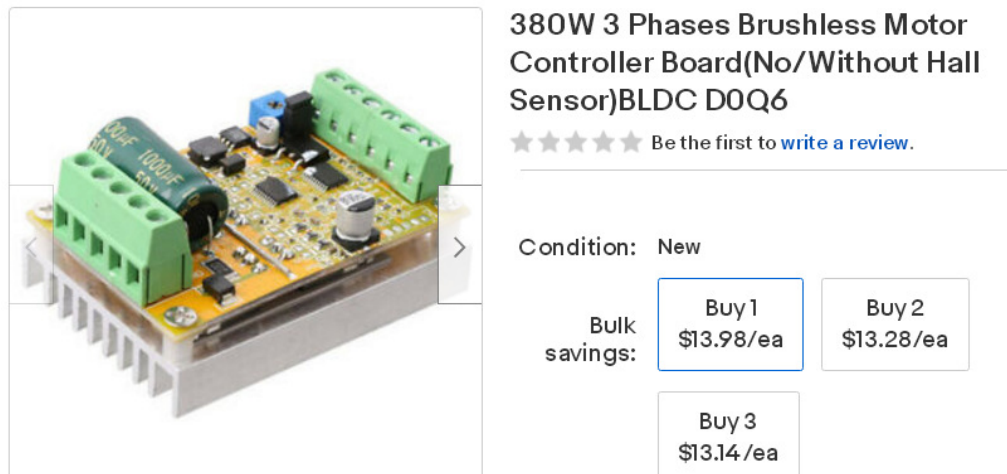


Figure 13: One of many ebay listings for a BLDC driver

These boards

- Take an analog input (0..5V), and
- Generate 3-phase AC waveforms
- With a frequency (i.e. speed) proportional to the input voltage

The hardware connections are as follows:

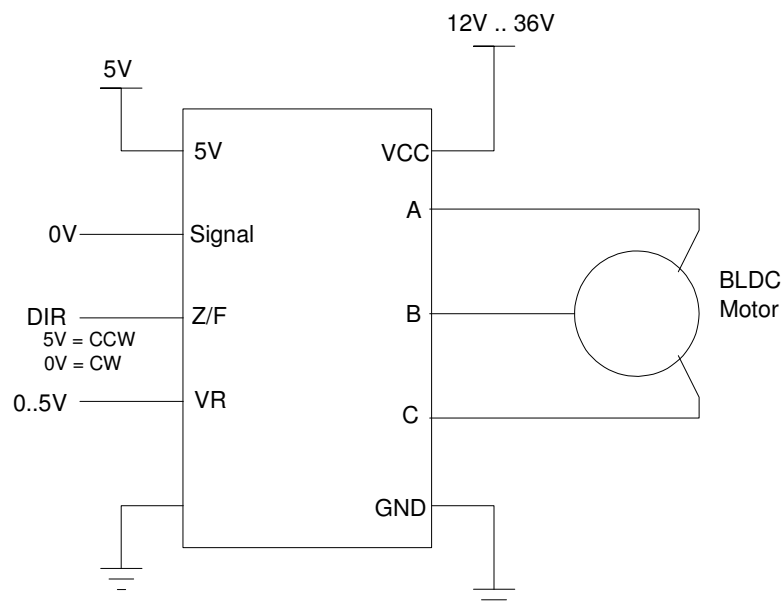


Figure 14: Hardware connections for driving a BLDC motor (i.e. a 3-phase AC motor)

If you look at the voltages on an oscilloscope, the signals at V_a , V_b , and V_c look like the following:

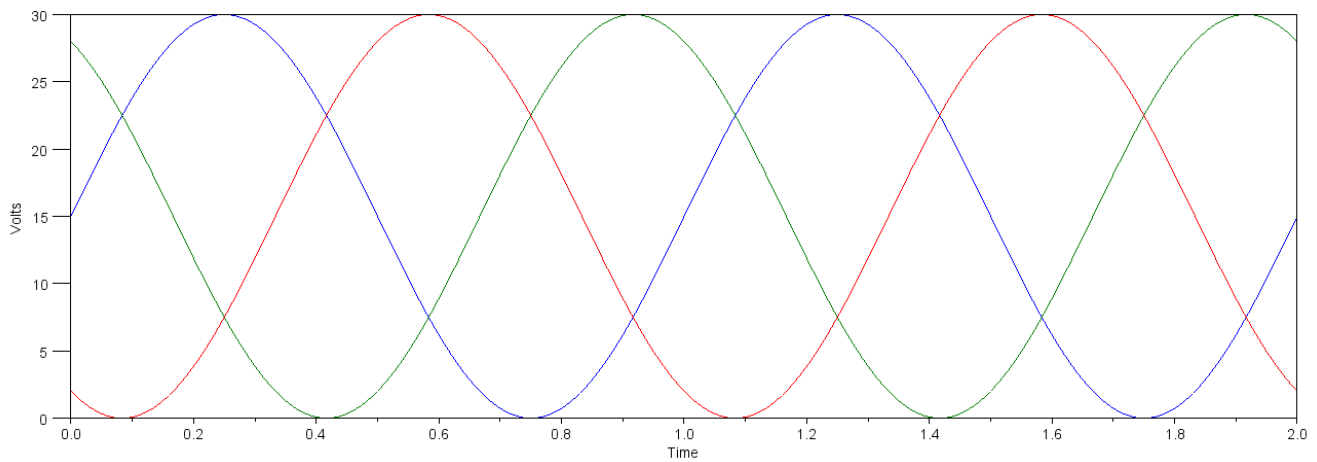


Figure 15: Voltages V_a , V_b , and V_c . The frequency is proportional to the input voltage at Z/F

The difference in voltages (V_{ab} , V_{bc} , V_{ca}) then are the desired 3-phase AC signals

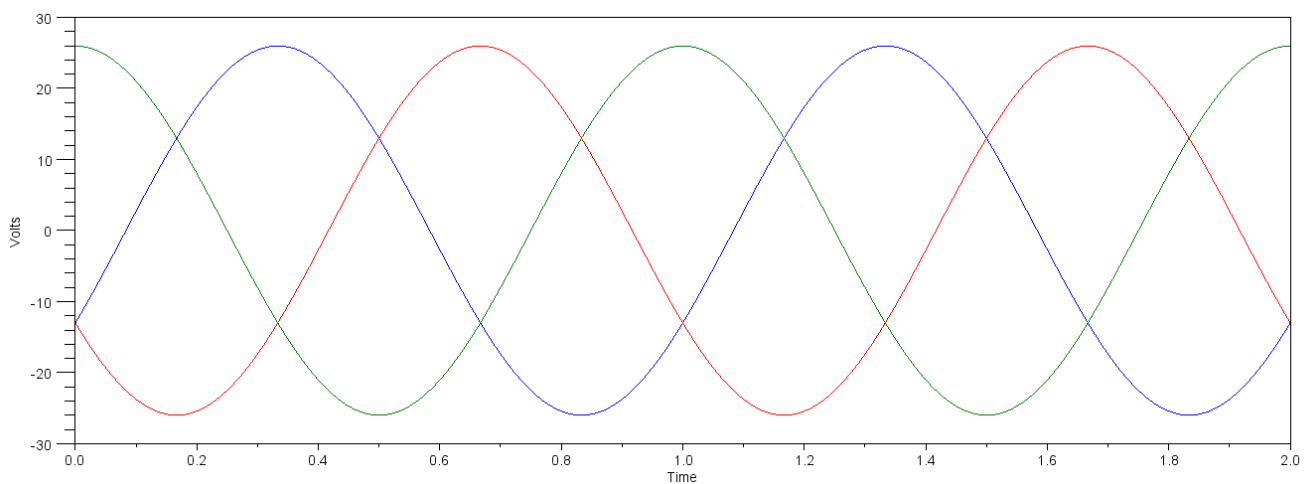


Figure 16: Voltages V_{ab} , V_{bc} , V_{ca} . The frequency is proportional to the input voltage at Z/F