Using a Digital Potentiometer with the Arduino

The Arduino can be used to control a digital potentiometer, using the I2C ("I-squared-C") protocol. There are a few things you should know about the I2C protocol to use the Arduino in this way. These brief notes are intended to help get you started.

AD5171 Digital Potentiometer:

A useful tutorial can be found here: [http://arduino.cc/en/Tutorial/DigitalPotentiometer](http://arduino.cc/en/Tutorial/DigitalPotentiometer). This tutorial makes use of the AD5171 digital potentiometer, which has 64 levels of resistance (i.e., the wiper can go to 64 different positions). The pin-out for the AD5171 is as follows (the figure is borrowed from the tutorial):

![AD5171 Pin Configuration](image)

Information regarding the various pins can be found on the AD5171 datasheet:

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>W</td>
<td>Wiper Terminal W. GND ≤ Vw ≤ VDD.</td>
</tr>
<tr>
<td>2</td>
<td>VDD</td>
<td>Positive Power Supply. Specified for operation from 2.7 V to 5.5 V. For OTP programming, VDD needs to be a minimum of 5.25 V and have a 100 mA driving capability.</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>Common Ground.</td>
</tr>
<tr>
<td>4</td>
<td>SCL</td>
<td>Serial Clock Input. Requires a pull-up resistor. If it is driven direct from a logic controller without the pull-up resistor, ensure that Vt min is 0.7 V × VDD.</td>
</tr>
<tr>
<td>5</td>
<td>SDA</td>
<td>Serial Data Input/Output. Requires a pull-up resistor. If it is driven direct from a logic controller without a pull-up resistor, ensure that Vt min is 0.7 V × VDD.</td>
</tr>
<tr>
<td>6</td>
<td>ADD0</td>
<td>IC Device Address Bit. Allows a maximum of two AD5171s to be addressed.</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>Resistor Terminal B. GND ≤ Vb ≤ VDD.</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>Resistor Terminal A. GND ≤ Va ≤ VDD.</td>
</tr>
</tbody>
</table>

Please look at the tutorial noted above to see how the AD5171 and the Arduino should be wired up together. Note that the ADD0 pin is tied to ground in the tutorial. This pin is used to “address” the pot. Tying it to ground corresponds to a “zero,” while tying it to 5V corresponds to a “one.” In order to “write” to the AD5171, the following bytes need to be sent (this screenshot is taken from the AD5171 datasheet):

<table>
<thead>
<tr>
<th>Slave Address Byte</th>
<th>Instruction Byte</th>
<th>Data Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>T</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>A</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>D5</td>
</tr>
<tr>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
<td>A</td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Arduino takes care of the specifics using ones of its libraries. Here is the code used in the tutorial:
// I2C Digital Potentiometer
// by Nicholas Zambetti <http://www.zambetti.com>
// and Shawn Bonkowski <http://people.interaction-ivrea.it/s.bonkowski/>

// Demonstrates use of the Wire library
// Controls AD5171 digital potentiometer via I2C/TWI
// Created 31 March 2006
// This example code is in the public domain.
// This example code is in the public domain.

#include <Wire.h>

void setup()
{
  Wire.begin(); // join i2c bus (address optional for master)
}

byte val = 0;

void loop()
{
  Wire.beginTransmission(44); // transmit to device #44 (0x2c)
    // device address is specified in datasheet
  Wire.write(byte(0x00)); // sends instruction byte
  Wire.write(val);       // sends potentiometer value byte
  Wire.endTransmission(); // stop transmitting

  val++; // increment value
  if(val == 64) // if reached 64th position (max)
  {
    val = 0; // start over from lowest value
  }
  delay(500);
}

We can understand various pieces of this code by looking at Table 7, above.

**Wire.beginTransmission(44):** This starts the transmission of data to the pot. The “44” corresponds to the “slave address byte” (i.e., the first byte) in Table 7. Sending this byte basically tells the AD5171 (as opposed to other devices that could be on the I2C line as well) to wake up and pay attention. The first byte in Table 7 is 010110(AD0)0. If you look in the Arduino documentation, you will see that the Arduino uses a 7-bit format for beginTransmission, dropping the last bit. Since we have tied AD0 to ground, it is a “0”. Thus, we use 0101100, which is 32+8+4=44. Note that we could have two AD5171 pots on the I2C line, one with AD0 at ground and one with it at 5V. In this case, we would use “44” to address one of them and “45” to address the other one.
**Wire.write(byte(0x00)):** This corresponds to the “instruction byte” (i.e., the second byte) that is sent to the digital pot. In Table 7 this is listed as TXXXXXXX. Here T=1 would be to permanently burn the resistance setting to the pot, and X means “don’t care.” Thus, in the sample code, 0x00 is sent (in hexadecimal).

**Wire.write(val):** Finally, the desired wiper setting is sent to the pot as a number between 0 and 63.

**DS1803 Digital Potentiometer:**

The DS1803 digital potentiometers are a bit different than the AD5171s. Here is the pin-out, taken from the datasheet:

```
H1 1 16 VCC
NC 2 15 NC
L1 3 14 H0
W1 4 13 L0
A2 5 12 W0
A1 6 11 NC
A0 7 10 SDA
GND 8 9 SCL
```

...and here is the protocol for writing to the chip, also taken from the datasheet:

**2–WIRE WRITE PROTOCOL Figure 5**

In this case, the A0, A1 and A2 pins can be used to specify the specific chip to be addressed, meaning that 8 different DS1803s could be used on the same I2C line by tying these pins to various combinations of ground and 5V. If we tie all three to ground, and strip off the last bit, we have 0101000, which is 32+8=40. Thus, transmission will be initiated with
“Wire.beginTransmission(40).” The command byte for writing to Pot-1 (which is the one that I had wired up) is 10101010. This is 170 in decimal or 0xaa in hexadecimal. Thus, we use “Wire.write(byte(0xaa))” to send the instruction byte. Finally, since the DS1803 is a 256-step pot, we can vary “val” from 0 to 255 (instead of “0 to 63,” as was the case for the AD5171).

Thus, the following code should work (I also shortened the delay):

```cpp
#include <Wire.h>

void setup()
{
  Wire.begin(); // join i2c bus (address optional for master)
}

byte val = 0;

void loop()
{
  Wire.beginTransmission(40); // transmit to device #40
     // device address is specified in datasheet
  Wire.write(byte(0xaa)); // sends instruction byte
  Wire.write(val); // sends potentiometer value byte
  Wire.endTransmission(); // stop transmitting

  val++; // increment value
  if(val == 256) // if reached 256th position (max)
  {
    val = 0; // start over from lowest value
  }
  delay(5);
}
```