Rotary Encoders

The Potentiometer is popular as a user input for many types of functions. Volume control is quite popular, as is heat, speed, brightness, etc., but it has a few limitations.

To use it digitally, it requires an A2D mechanism, and not all microcontrollers may have them, or enough of them, or enough resolution to get the detail required for a particular application. And it may require some additional support circuitry to get a proper range, and mapping its response curve (typically linear, logarithmic, or 'audio') to a particular application can be tricky.

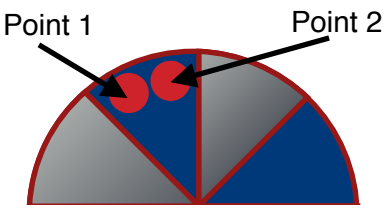
More importantly it is fixed in its mechanical operation. While that's fine in some applications, or can be solved mechanically in others (e.g. the joystick). There are times when you may need, say multiple revolutions, or maybe you want to restart at a set value, say with the volume on low after a power cycle instead of wherever the POT was left?

Further, it doesn't lend itself to allowing multiple control inputs. Imagine if a bluetooth speaker used a POT for volume control — then also wanted to allow the bluetooth connected device to change the volume? What happens when the POT is turned the next time? There must be another option.

Enter the ***rotary encoder***. The one in your kit looks like a potentiometer physically, but as you rotate it, you will discover that there are no end stops, it rotates in either direction continually. There are also more connections. This one has 5 connections, though one of these is for an internal button, activated by pressing in on the knob. Then power and ground and finally two other output signals.

It is by monitoring the output of these two pins, the microcontroller can determine when a rotation occurs and in what direction. Lets look at how.

Imagine that inside the rotary encoder there are two contact points placed next to each other facing a small wheel attached to the shaft. (*see diagram*). The wheel is a conductor and connected to the +5VDC line. But along the wheel alternating sections are covered with an insulator (say a thin plastic film). These sections are symmetric and just barely wider than the two contact point.



Blue == insulator
Silver == conductor

When the shaft is turned, the wheel rotates. As it does the two points will transition from +5V to 0 and back, however, depending on which way it turns, one pin or the other will be the first to transition to +5V, then the other, then the first pin will move back to 0, and so on.

By keeping track of the order that the points transition, one can determine the direction of rotation. This is called ***quadrature encoding,*** since there are 4 possible states for the 4 pins.

The simple rotary encoder we are using offers fairly low resolution and possible rotation speed. It also has 'detents', or stops about every 1/20th the way around, which is useful for a volume or similar control, so vibration doesn't change the setting.

However there are many other types of encoders, which use a similar concept but different systems to detect the points, i.e. laser/led, magnetic, etc. These may operate at much higher resolutions and speeds. They could, for example, monitor systems on your car, like wheel or crankshaft speed and position.

Combining a rotary encoder with an electric motor, allows you to drive a motor to a very specific position. Typically called Servomotors, or just servos for short, these come in a couple of varieties, some simple hobby-style units that actually use a potentiometer and endstop, and more precise ones which use a rotary encoder. The most common everyday application for servos are in inkjet printers, for controlling the printheads, paper feeders, etc.

As usual, will will be using an existing software library to handle the specifics. Monitoring the change of these two pins requires either very frequent checking of the two outputs, or the use of something called ***interrupts***.

Interrupts are special hardware signals which do just what their name implies, interrupt the processor's linear flow of instructions and jump, or ***vector*** as it is sometimes called, to a new set of instructions. These hardware level interrupts can be associated with certain of the GPIO pins, as well as to a few internal mechanisms, like timers. The pinout diagram shows which pins on the nano can be configured to trigger an interrupt. (hint, D2 and D3)

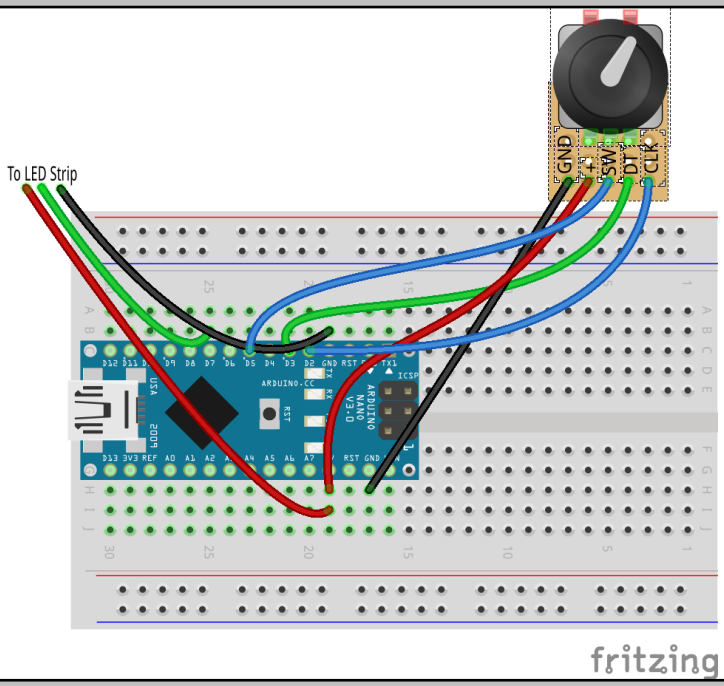
Check out the links below for some more details on rotary encoders and quadrature encoding.

Basic Wikipedia description

Video of homemade rotary encoder

PJRC explanation of the rotary encoder

Experiment: Round & Round

Components	Wiring Diagram
✓Nano ✓rotary encoder ✓LED strip	

Connection Instructions

Connect the LED strip as you have previously. You can place the rotary encoder in the breadboard or use M->F dupont jumpers to connect it. GND and + should be obvious. Connect SW to D5 for this example, and then connect the two output pins, labeled (oddly) DT and CLK on our board to D3 and D2 respectively.

Sketch(es)	roundnround.ino

Analysis Questions

What happens to the output if you swap the connections to D2 and D3?
What happens if you change to D11 and D12, altering the #defines in the code appropriately.

Programming Tasks

The sample program outputs some values over the serial link, and lights the LED strip a solid blue.
Modify it such that rotating the knob clockwise increases the # of blue LEDs, and counter clockwise rotation decreases the number. Be sure that over-rotation in either direction doesn't light more LEDs than you have (or less than 0!)
Pressing the button should toggle the LED strip on and off, your decision if you wan to start back at 0 or retain the previous value.
Optionally detect a double-push of the button, such that a double push rotates through a few colors (e.g. RED, BLUE, GREEN, YELLOW)