Electronically Interlocking Radio Buttons (*improved!*)

by throbscottle

The term "radio buttons" comes from the design of old car radios, where there would be a number of push buttons pre-tuned to different channels, and mechanically interlocked so that only one can be pushed in at a time.

I wanted to find a way of making radio buttons without having to buy some actual interlocking switches, because I want to be able to select alternative preset values in another project which already has a rotary switch, so I wanted a different style to avoid mistakes.

Tactile switches are plentiful and cheap, and I have a load dismantled from various things, so they seemed the natural choice to use. A hex D-type flip flop, the 74HC174, performs the interlock function nicely with the help of some diodes. Possibly some other chip could do a better job but the '174 is very cheap, and the diodes were free (board pulls)

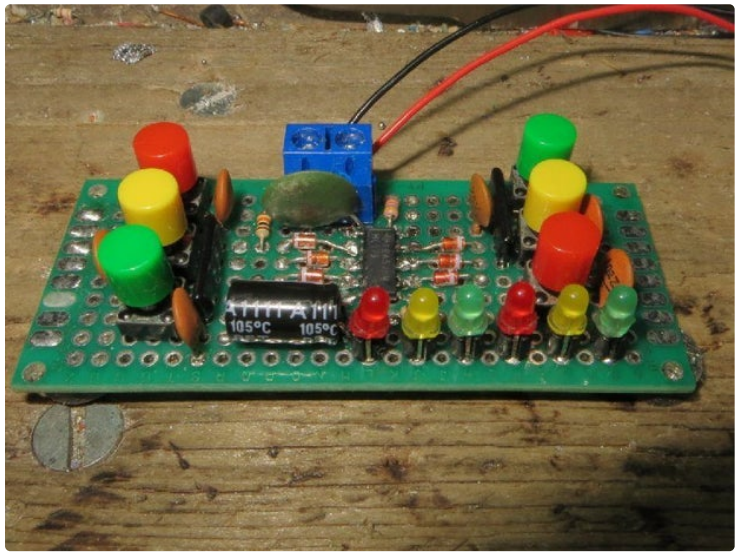
Some resistors are also needed, and capacitors to de-bounce the switches (in the first version) and provide power-on-reset. I have since found that by increasing the clock delay capacitor, the switch debounce capacitors are not needed.

The simulation "interlock.circ" runs in Logisim, which you can download here: http://www.cburch.com/logisim/ (Sadly no longer under development).

I have produced 2 improved versions of the circuit, in the first, just the debounce capacitors are removed. In the second, a transistor is added to enable one of the buttons to be activated at switch on time, giving a default setting.

Supplies:

- 1x 74HC174
- 6x tactile switches or other type of momentary switch
- 7x 10k resistors. These can be SIL or DIL packaged with a common terminal. I used 2 packages containing 4 resistors each.
- 6x 100n capacitors - exact value is not important.
- 1x 47k resistor
- 1x 100n capacitor, minimum value. Use anything up to 1u.
- Output devices, eg small mosfets, or LEDs
- Materials for assembling circuit



Step 1: Construction

Assemble using your preferred method. I used double sided perforated board. It would be easier to do with a through hole DIL packaged chip, but I often get SOIC devices because they're usually much cheaper.

So with a DIL device, you don't have to do anything special, just plug it in and wire it up.

For an SOIC, you need to do a little trick. Bend alternate legs up a little so they don't touch the board. The remaining pins will be at the correct spacing to match the pads on the board. Here's a guide to how I bent mine (UP means bent up, DOWN means leave alone)

- UP: 1, 3, 5, 7, 10, 12, 14, 16
- DOWN: 2, 4, 6, 8, 9, 11, 13, 15

This way 4 of the diodes can be connected to pads and only 2 need to be connected to raised legs. Part of me suspects this would be better the other way around, however.

Lay the diodes out to either side of the chip and solder them in place.

Fit the pull-down resistors for each of the D inputs. I used 2 SIL packs of 4 resistors each,

Fit the pull-down resistor for the clock input. If using SIL packages, connect one of the spare resistors instead of a separate one

Fit the switches next to the resistors.

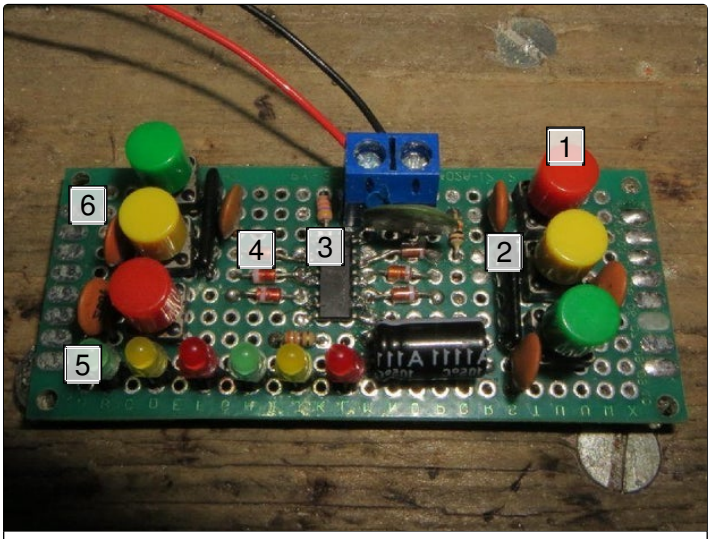
Fit the de-bouncing capacitors for the switches as close to them as will fit.

Fit your output devices. I used LEDs for testing and demonstration, but you could fit some other device of your

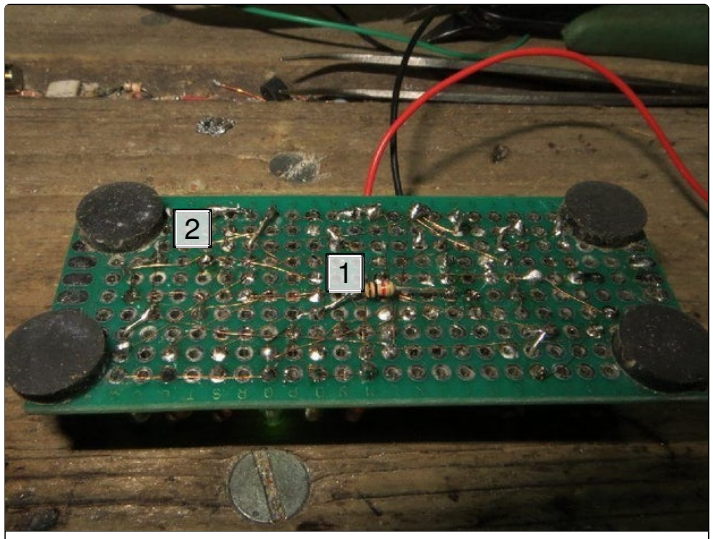
choosing to get multiple poles on each output, for example.

- If you fit LEDs they only need 1 current limiting resistor in the common connection, as only 1 LED is lit at a time!
- If you use MOSFETs or other devices, pay attention to the orientation of the device. Unlike a real switch, the signal still has a relationship to the 0v connection of this circuit so the output transistor must be referenced to it.

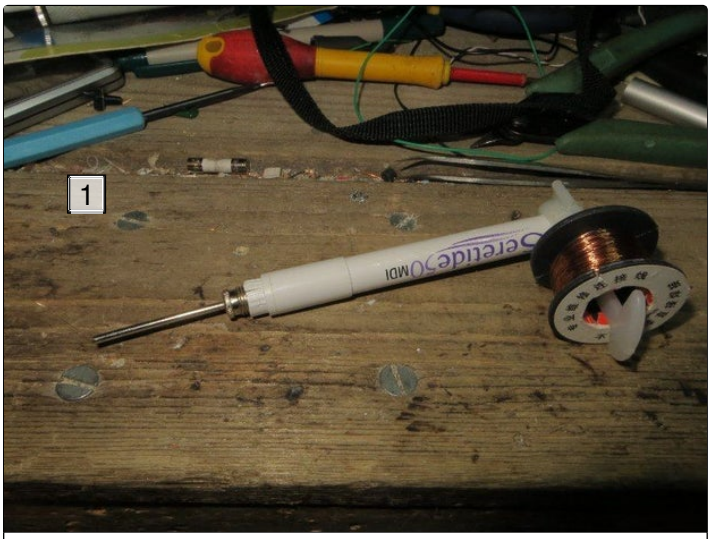
Wire everything together according to the schematic. I used 0.1mm magnet wire for this, you may prefer something a bit less fine.



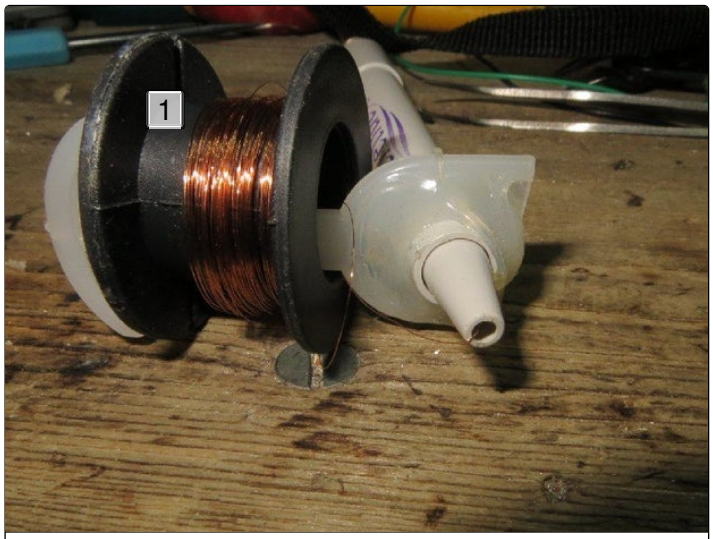
1. Tactile switches fitted with coloured tops
2. SIL package of 4 resistors
3. 74HC174
4. Diodes
5. LEDs for demonstration and testing
6. Debounce capacitors



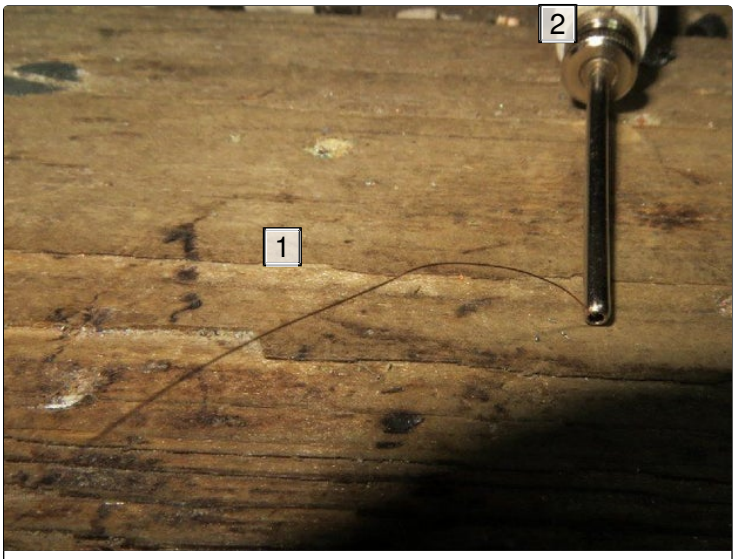
1. 1k resistor helps delay clock slightly. It's on this side because I was experimenting
2. Messy wiring using 0.1mm magnet wire



1. Home made wiring pen. It has some crumpled mylar film inside to grip the wire.



1. Magnet wire goes in one end of the pen. The little spool holder is just glued on



1. 0.1mm wire
2. Ball-inflating nozzle that came with a pump

Step 2: How It Works

I've provided 4 versions of the schematic: the original with switch debouncing capacitors, with and without output mosfets, and a further two versions where the clock delay capacitor has been increased, so that debouncing the switches has become unnecessary, finally with the addition of a transistor which will virtually "press" one of the buttons when the power is turned on.

The circuit uses simple D type flip-flops with a common clock, conveniently you get 6 of these in the 74HC174 chip.

The clock and each of the D inputs of the chip is pulled to ground via a resistor, so the default input is always 0. The diodes are connected as a "wired OR" circuit. You could use a 6 input OR gate, then you wouldn't need the pull down on the clock input, but where's the fun in that?

When the circuit is first switched on, the CLR pin is pulled low via a capacitor to reset the chip. When the capacitor charges, the reset is disabled. I chose 47k and 100nF to give a time constant approximately 5x that of the combined debounce caps and pull down resistors used for the switches.

When you press a button, it puts a logic 1 on the D input it's connected to and via a diode triggers the clock at the same time. This "clocks in" the 1, making the Q output go high.

When the button is released, the logic 1 is stored in the flip-flop, so the Q output remains high.

When you press a different button, the same effect takes place on the flip-flop it is connected to, but because the clocks are commoned, the one which has a 1 on it's output already now clocks in a 0, so it's Q output goes low.

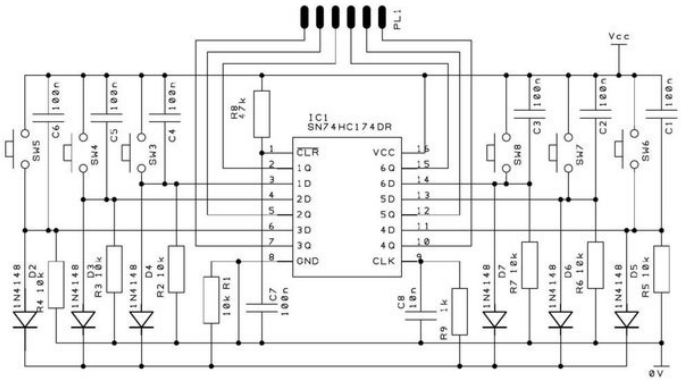
Because the switches suffer from contact bounce, when you press and release one you don't get a neat 0 then 1 then 0, you get a stream of random 1's and 0's, making the circuit unpredictable. You can find a decent switch debouncing circuit here: http://www.labbookpages.co.uk/electronics/debounce.html

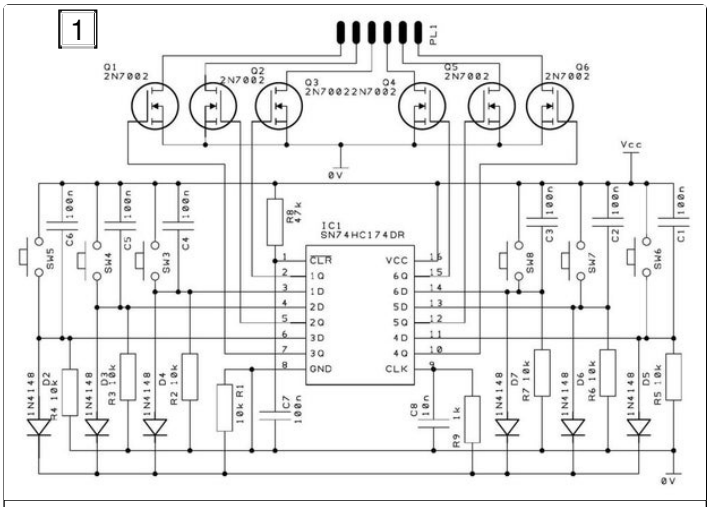
I eventually found that with a sufficiently large clock delay capacitor, debouncing individual switches is un-necessary.

The Q output of any flip-flop goes high when it's button is pressed, and the not-Q output goes low. You can use this to control a N or P MOSFET, referenced to the low or high power rail, respectively. With the load connected to the drain of any transistor, it's source would typically be connected to 0v or the power rail, depending on polarity, however it will act as a switch referenced to some other point, as long as it still has headroom to turn on and off.

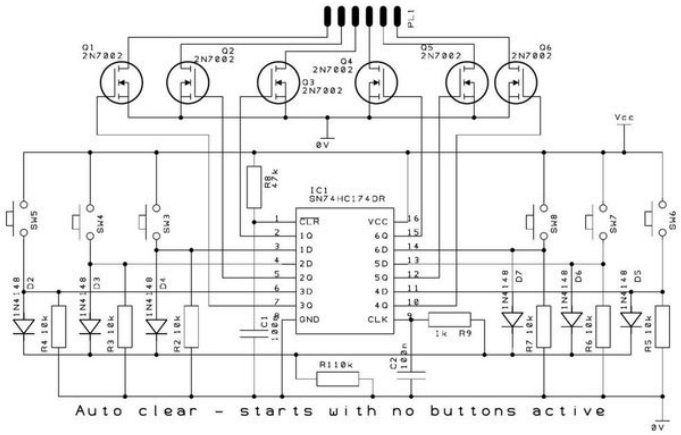
The final schematic shows a PNP transistor which is connected to one of the D inputs. The idea is that when power is applied, the capacitor at the base of the transistor charges until it reaches the point where the transistor conducts. Because there is no feedback, the collector of the transistor changes state very quickly, generating a pulse which can set the D input high and trigger the clock. Because it is connected to the circuit via a capacitor, the D input returns to it's low state and is not noticeably affected in normal operation.

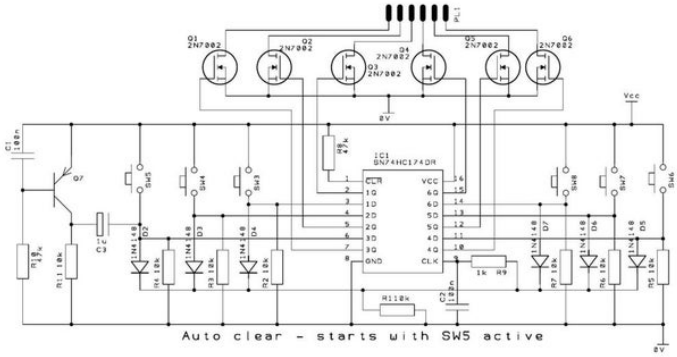
https://youtu.be/plM4teEuBV8





1. Alternative output





[redacted] **https://www.instructabl…** Download

[redacted] **https://www.instructabl…** Download

[redacted] **https://www.instructabl…** Download

[redacted] **https://www.instructabl…** Download

Step 3: Pros and Cons

After I built this circuit I wondered if it was worthwhile doing. The objective was to get radio button like functionality without the expense of the switches and mounting frame, however once the pull-down resistors and de-bouncing capacitors were added in, I found it a bit more complex than I would have liked.

Real interlocking switches don't forget which switch was pressed when the power is turned off, but with this circuit it will always return to it's default setting of "none", or a permanent default.

A simpler way to do the same thing would be to use a microcontroller, and I don't doubt someone is going to point this out in the comments.

The problem with using a micro is, you have to program it. Also you have to either have enough pins for all the inputs and outputs you need, or have a decoder to create them, which instantly adds another chip.

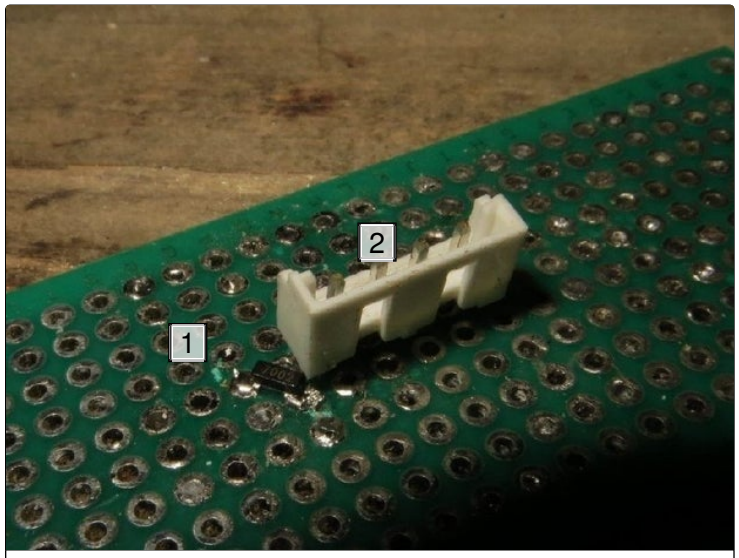
All the parts for this circuit are very cheap or free. A bank of 6 interlocking switches on eBay costs (at the time of writing) £3.77. Ok so that's not much, but my 74HC174 cost 9 pence and I already had all the other parts, which are cheap or free anyway.

The minimum amount of contacts you normally get with a mechanical interlocking switch is DPDT, but you can easily get more. If you want more "contacts" with this circuit, you have to add more output devices, typically mosfets.

One big advantage compared to standard interlocking switches is that you can use any type of momentary switches, positioned anywhere you like, or even drive the inputs from entirely different signal.

If you add a mosfet transistor to each of the outputs of this circuit, you get an SPCO output, excepts it's not even really that good, because you can only connect it 1 way. Connect it the other way and you get a really low powered diode instead.

On the other hand, you can add a lot of mosfets to an output before it gets overloaded, so you can have an arbitrarily large number of poles. By using P and N type pairs, you can also create a bi-directional outputs, but this also adds complexity. You can also use the not-Q outputs of the flip-flops, which gives you an alternative action. So there is potentially a lot of flexibility with this circuit, if you don't mind the extra complexity.



1. 2N7002 MOSFET
2. Header for connecting a fan