Drum Machine Hacks and Mods

04

Endless Encoder Input

Endless Encoder Input

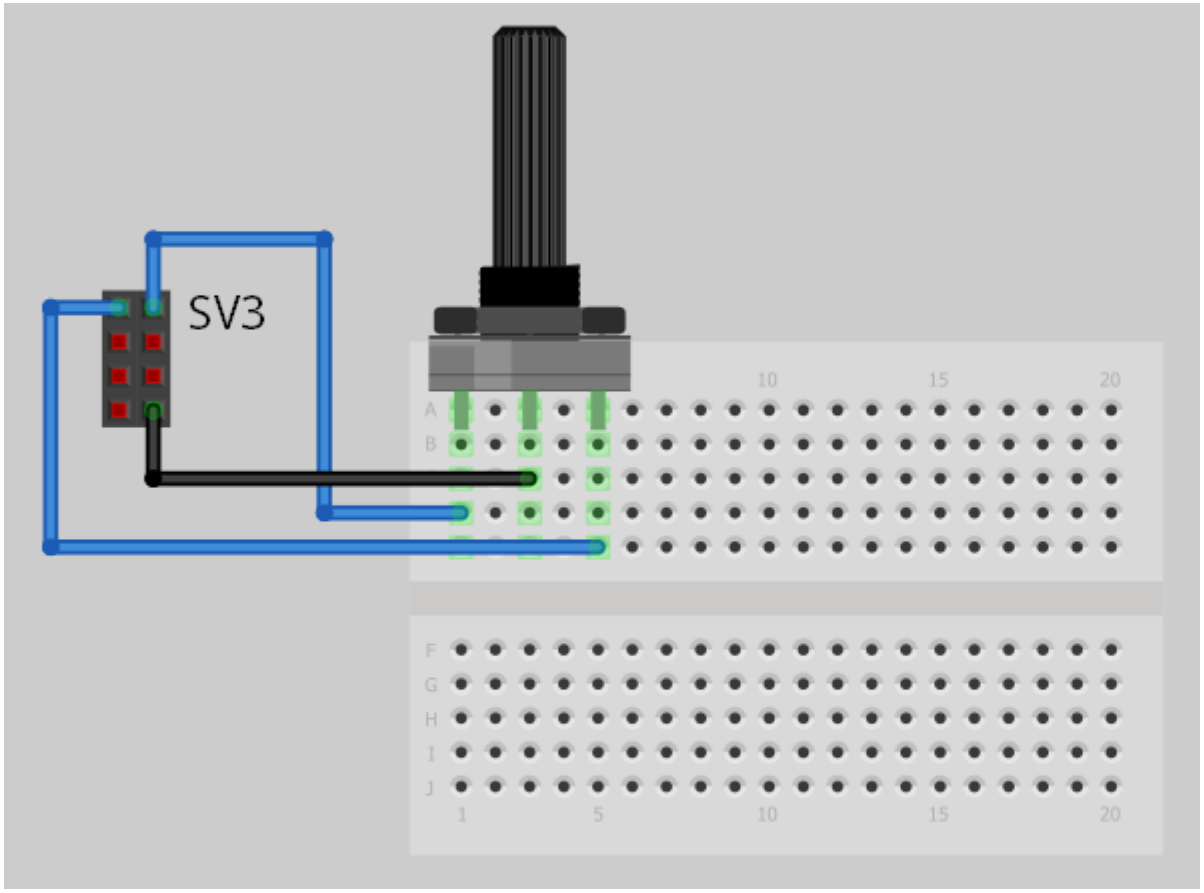
The great thing about an Encoder is that you can do quick value changes with this, and it works on any value on the interface. It actually sends up / down commands to the current selected value, just like pressing up and down on the interface buttons, but very quickly.

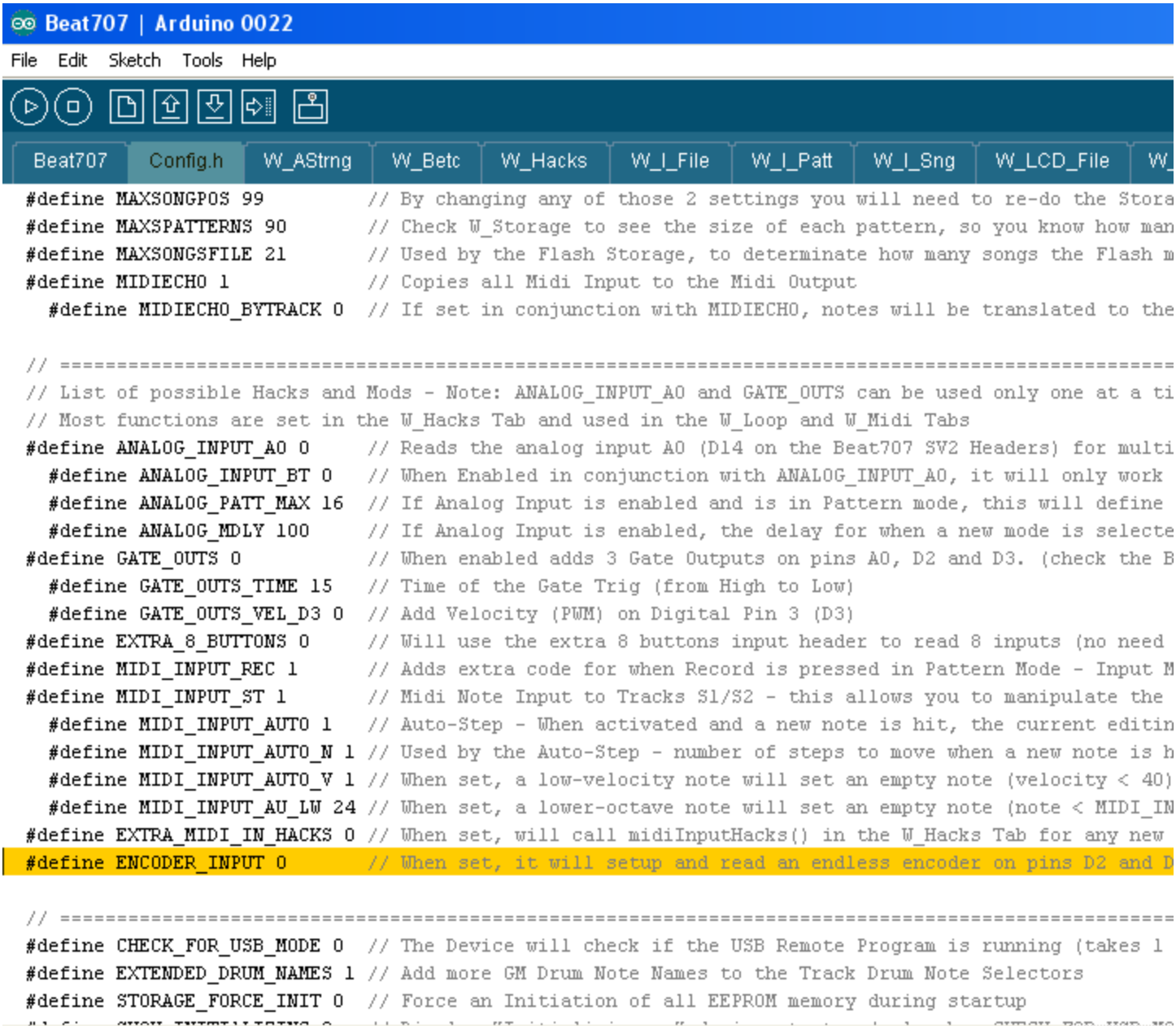
In order to use an Endless Encoder, all you need to do is active it in the Config.h Tab, and be sure you connect the device correctly to the SV3 header, by using pins D2, D3 and Ground.

The connections are very easy: A, B and C. B is Ground, A would go to D3 and C to D2. Simple as that. If the encoder is going the wrong direction, just swap D2 with D3 and vice-versa. (we are not using the push-click pins of the encoder, just A, B and C)

SparkFun.com has a very nice Encoder, the same we used on the YouTube Hack #4 video: http://www.sparkfun.com/products/9117

To enable the Encoder code, set ENCODER_INPUT to 1 in the Config.h Tab.




Beat707 | Arduino 0022
File Edit Sketch Tools Help
Beat707 Config.h W_AStrng W_Betc W_Hacks W_I_File W_I_Patt W_I_Sng W_LCD_File W_
#define MAXSONGPOS 99 // By changing any of those 2 settings you will need to re-do the Stora
#define MAXSPATTERNS 90 // Check W_Storage to see the size of each pattern, so you know how man
#define MAXSONGSFILE 21 // Used by the Flash Storage, to determinate how many songs the Flash m
#define MIDIECHO 1 // Copies all Midi Input to the Midi Output
#define MIDIECHO_BYTRACK 0 // If set in conjunction with MIDIECHO, notes will be translated to the
// ==========
// List of possible Hacks and Mods - Note: ANALOG_INPUT_A0 and GATE_OUTS can be used only one at a ti
// Most functions are set in the W_Hacks Tab and used in the W_Loop and W_Midi Tabs
#define ANALOG_INPUT_A0 0 // Reads the analog input A0 (D14 on the Beat707 SV2 Headers) for multi
#define ANALOG_INPUT_BT 0 // When Enabled in conjunction with ANALOG_INPUT_A0, it will only work
#define ANALOG_PATT_MAX 16 // If Analog Input is enabled and is in Pattern mode, this will define
#define ANALOG_MDLY 100 // If Analog Input is enabled, the delay for when a new mode is selecte
#define GATE_OUTS 0 // When enabled adds 3 Gate Outputs on pins A0, D2 and D3. (check the B
#define GATE_OUTS_TIME 15 // Time of the Gate Trig (from High to Low)
#define GATE_OUTS_VEL_D3 0 // Add Velocity (PWM) on Digital Pin 3 (D3)
#define EXTRA_8_BUTTONS 0 // Will use the extra 8 buttons input header to read 8 inputs (no need
#define MIDI_INPUT_REC 1 // Adds extra code for when Record is pressed in Pattern Mode - Input M
#define MIDI_INPUT_ST 1 // Midi Note Input to Tracks S1/S2 - this allows you to manipulate the
#define MIDI_INPUT_AUTO 1 // Auto-Step - When activated and a new note is hit, the current editin
#define MIDI_INPUT_AUTO_N 1 // Used by the Auto-Step - number of steps to move when a new note is h
#define MIDI_INPUT_AUTO_V 1 // When set, a low-velocity note will set an empty note (velocity < 40)
#define MIDI_INPUT_AU_LW 24 // When set, a lower-octave note will set an empty note (note < MIDI_IN
#define EXTRA_MIDI_IN_HACKS 0 // When set, will call midiInputHacks() in the W_Hacks Tab for any new
#define ENCODER_INPUT 0 // When set, it will setup and read an endless encoder on pins D2 and D
// ==========
#define CHECK_FOR_USB_MODE 0 // The Device will check if the USB Remote Program is running (takes 1
#define EXTENDED_DRUM_NAMES 1 // Add more GM Drum Note Names to the Track Drum Note Selectors
#define STORAGE_FORCE_INIT 0 // Force an Initiation of all EEPROM memory during startup