```
// Scan Makin 32 note organ pedalboard with Arduino Leonardo board
// Using MIDIUSB library v.1.0.4 by Gary Grewal
// John Harvey May 2019

// This version attemptes to debounce seriously corroded pedal contacts that resist all attempts to clean up
properly
// Help debounce by slowing scan right down to minimum musically possible (can't play pedals as fast as manu
anyway)
// Read pedal contact states sequentally into two KeyState arrays time separated by half loop period and onl
send MIDI noteOn if both agree
// Loop period is 25ms

//https://www.arduino.cc/en/Reference/MIDIUSB
//https://www.arduino.cc/en/Tutorial/MidiDevice
//https://github.com/arduino-libraries/MIDIUSB
//https://github.com/arduino/tutorials/blob/master/ArduinoZeroMidi/ArduinoZeroMidi.ino

#include  <MIDIUSB.h>
//#include  <pitchToFrequency.h>
//#include  <pitchToNote.h>
//#include  <frequencyToNote.h>

// Arduino input/output pins to interface to Makin pedalboard scanner PCB
const byte CK = 0;           // Clock output
const byte PS = 1;           // Parallel/serial select output
const byte SR = 2;           // Serial data in from 32-bit shift register (daisy chain of 4014s)
const byte ScopeSync = 7;    // Trigger oscilloscope at start of loop

// Create arrays to remember key states, true = key on, false = key off
boolean  KeyState[32];
boolean  KeyState1[32];
boolean  KeyState2[32];
boolean  PreviousKeyState;

void setup() {
  pinMode (CK, OUTPUT);
  digitalWrite(CK, LOW);
  pinMode (PS, OUTPUT);
  digitalWrite(PS, LOW);
  pinMode (SR, INPUT_PULLUP);
  pinMode (ScopeSync, OUTPUT);
  digitalWrite(ScopeSync, LOW);

  // Initialize digital pin LED_BUILTIN as an output and set LED to off
  pinMode(LED_BUILTIN, OUTPUT); // For bottom C visual indication
  digitalWrite(LED_BUILTIN, LOW);

  for (byte note = 0; note <= 31; note ++) {
    KeyState[note] = false;  // All notes off
    KeyState1[note] = false;  // All notes off
    KeyState2[note] = false;  // All notes off
  }
}

void loop() {

  digitalWrite(ScopeSync, HIGH);
  delayMicroseconds(100);
  digitalWrite(ScopeSync, LOW);

  // Scan pedals into two time-separated arrays
  // Get state of all notes (parallel JAM into shift registers)
  digitalWrite(PS, HIGH); // Enable parallel data entry
  delayMicroseconds(100);
  digitalWrite(CK, HIGH); // Latches data on rising edge of clock
  delayMicroseconds(100);
  digitalWrite(CK, LOW);
  delayMicroseconds(100);
  digitalWrite(PS, LOW);  // Restore serial data entry
  delayMicroseconds(100);
```

```
  for (byte note = 0; note <= 31; note ++) {
    KeyState1[note] = digitalRead(SR);       // First note (bottom C) is already available without shifting,
read first then shift
    if (note < 31) {                         // 32nd shift not needed
      digitalWrite(CK, HIGH);                // Shifts data on rising edge of clock
       delayMicroseconds(100);
       digitalWrite(CK, LOW);
       delayMicroseconds(100);
    }
  }

  delay(5);

  digitalWrite(PS, HIGH);
   delayMicroseconds(100);
  digitalWrite(CK, HIGH);
   delayMicroseconds(100);
  digitalWrite(CK, LOW);
   delayMicroseconds(100);
  digitalWrite(PS, LOW);
   delayMicroseconds(100);

  for (byte note = 0; note <= 31; note ++) {
    KeyState2[note] = digitalRead(SR);
    if (note < 31) {
      digitalWrite(CK, HIGH);
      delayMicroseconds(100);
      digitalWrite(CK, LOW);
      delayMicroseconds(100);
    }
  }

  delay(1);

  // Now process collected key data, comparing time-separated data for each key
  for (byte note = 0; note <= 31; note ++) {

    PreviousKeyState = KeyState[note];

    if (PreviousKeyState == false && KeyState1[note] == true && KeyState2[note] == true) {    // Note has jus
gone on
      KeyState[note] = true;
      midiEventPacket_t noteOn = {0x09, 0x92, byte(note + 0x24), 0x40}; // note + 0x24 produces an int in
compiler so use byte() to reconvert to byte value otherwise compiler outputs a narrowing conversion warning
       MidiUSB.sendMIDI(noteOn);
      MidiUSB.flush();
      if (note == 0) {
         digitalWrite(LED_BUILTIN, HIGH);    // Turn Arduino onboard LED on (Bottom C visual indication)
      }
    }

    if (PreviousKeyState == true && KeyState1[note] == false && KeyState2[note] == false) {    // Note has ju
gone off
      KeyState[note] = false;
      midiEventPacket_t noteOn = {0x08, 0x82, byte(note + 0x24), 0x40};
      MidiUSB.sendMIDI(noteOn);
      MidiUSB.flush();
      if (note == 0) {
         digitalWrite(LED_BUILTIN, LOW);    // Turn Arduino onboard LED off (Bottom C visual indication)
      }
    }

  }
  delay(4);
}
```