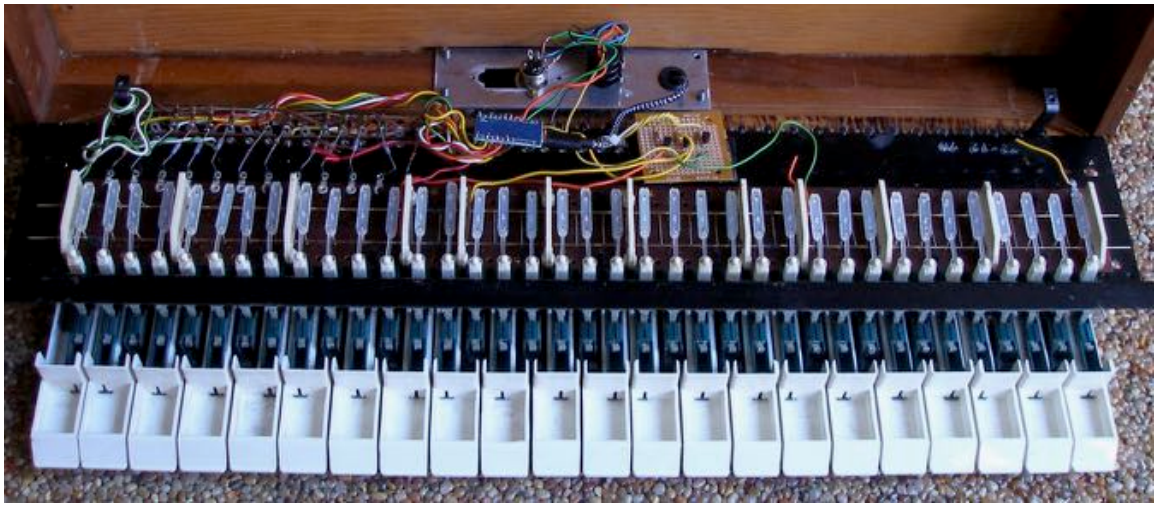
3-Octave Keyboard

3-Octave Music Keyboard

John Talbert, Sept. 2017

The Keyboard

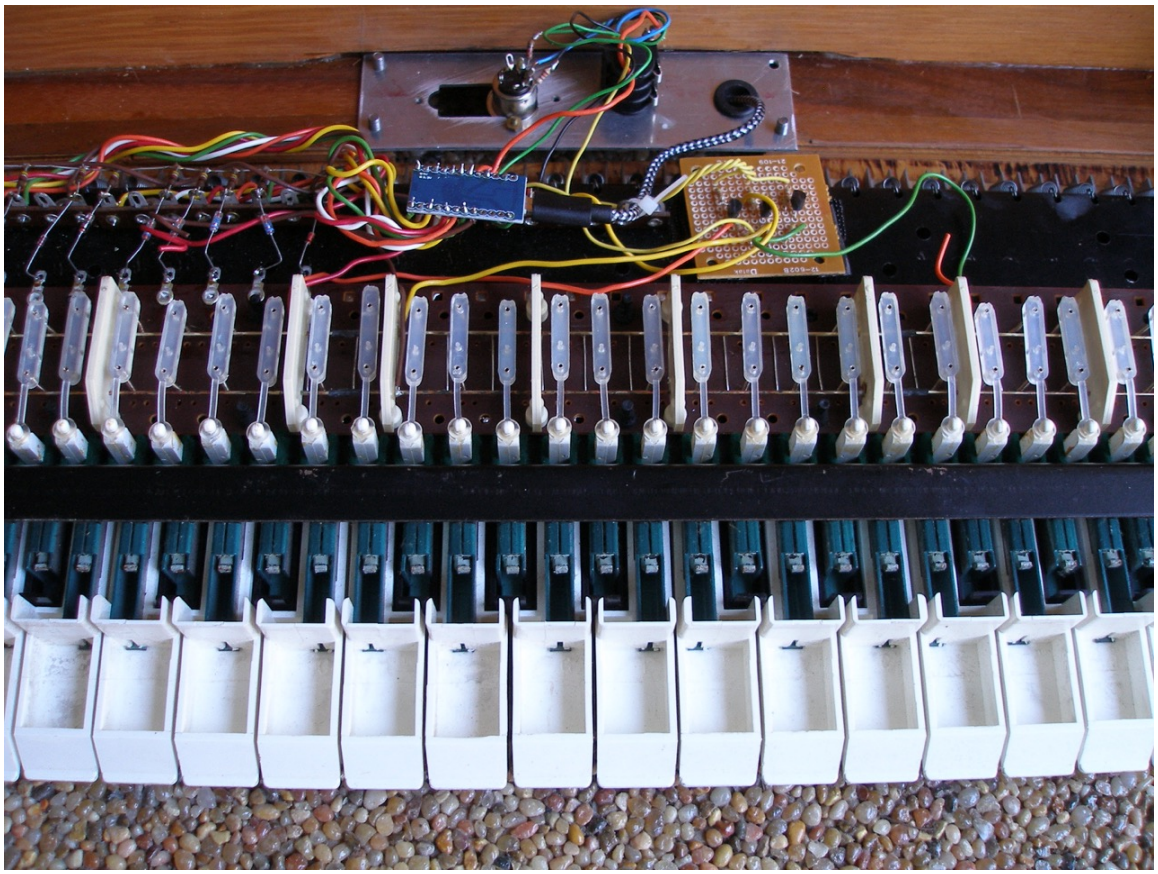
This is a 37-note music keyboard with electronic switches. At this time, the original instrument used with this keyboard is unknown.



Each key is connected to a flexible spring that is pulled and slightly stretched across a metal bar when the key is depressed. The metal bar covers the entire length of the keyboard but is electrically segmented into 4 separate “busses” each of which covers one octave or 12 keys. One

of the 4 is a very short and only covers the top “C” key. The present circuitry actually ignores this one top key and makes it non-functional.

The keyboard employs two of these key spring-bar circuits. I have worked out circuit diagrams for each of them. One sets up simple switches between the key springs and the octave buss bars. The other adds to this switch setup 12 resistor-diode circuits, one for each note of the octave.



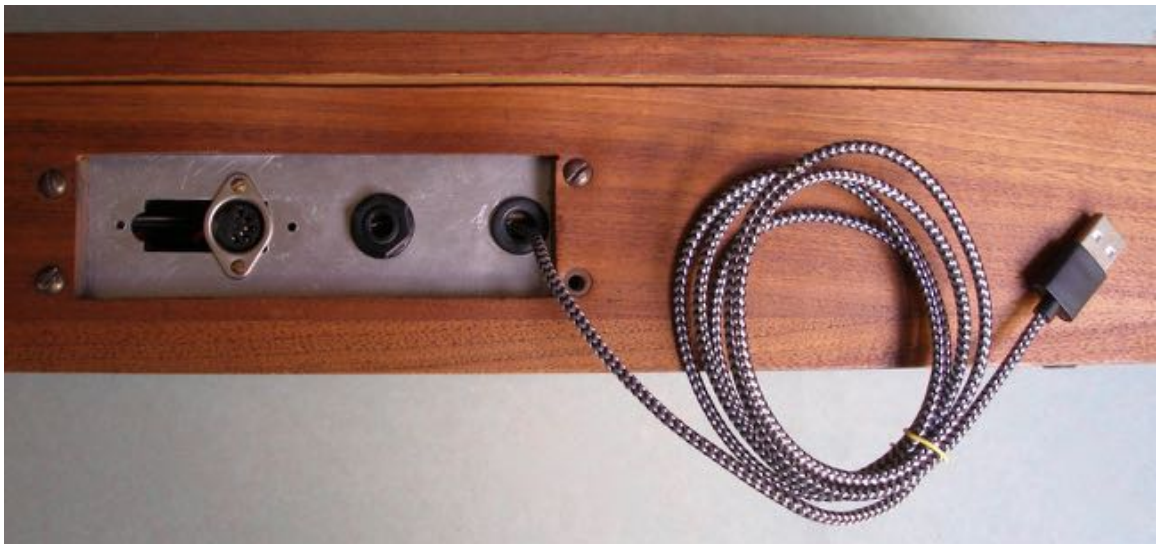
Duophonic

The keyboard is basically duophonic, able to correctly sense only two notes at a time. Three or more keys pressed will result in those notes appearing in all three octaves, not just the notes being pressed. This happens due to the limitations of a three bus switch setup.

It is best used as a monophonic keyboard that can accurately read legato style melodies where two notes might momentarily sound together in movement from one note to the next

The Circuit

A programmable micro controller has been installed inside the keyboard. It now has 3 useful outputs that are completely programmable. One is a USB serial cable that can transmit data on the state of the keys for use in programs like MAX/MSP. Another is a MIDI output that can transmit MIDI NoteOn and NoteOff commands initiated by key action. The third output is a phone jack that can output an audio signal in the form of a square wave.



The Microcontroller is an Arduino Pro Micro, sold by Sparkfun Electronics. Twelve of its pins have been configured as inputs to receive high or low voltage signals from the 12 resistor-diode circuits, one for each note of the octave. Three other pins have been configured as outputs to enable or disable each of the 3 octave busses. (The 4th, 'C' key buss has not been connected since we ran out of pins on the micro.)

To read the state of a particular key, you must first bring the octave buss for that key LOW by sending a HIGH voltage to the output pin connected to a common emitter transistor circuit that controls that buss.

The Programs

The Arduino Pro Micro (5v, 16MHz) microprocessor installed in the keyboard is fully programmable through its USB connection. I have written 3 programs that make use of each of the keyboard outputs – the serial USB line, the audio phone plug output, and the MIDI output.

In the basic program, all 36 keys are scanned, one octave at a time, and their on/off states are stored in an array. A second array stores the last scanned key states. Comparing the two array values reveals which keys have changed, having been pressed or released, at which point some action can be performed on an output.

All switches have “bounce” problems where the key state is unstable for several milliseconds due to the switch making and losing contact, or bouncing, when changing state. The usual way to deal with this in software is to delay any action when a state change is first detected. The initial change is noted in another array, but no action is taken until after a second scan confirms that the state change is still in effect.

Key bounce can be seen from the Arduino serial monitor as multiple changes happening on a single key press or release, instead of just one. A millisecond delay command in the program loop can be adjusted high enough that those multiple changes on a single key change disappear. It also

helps to occasionally clean off carbon deposits on the buss bars.

Serial Program

One program was written to output serial data over the USB cable. The data is in the form of key-number/key-state pairs. The key state equals 0 for pressed keys and 1 for released keys. The key-number is separated from the key-state with a space character, and the pairs are separated by carriage returns. The serial data can be read by the serial object in a program such as Max/MSP to affect sounds or videos or other interactions all programmed within the Max/MSP program.

MIDI Program

A second program was written to output MIDI NoteOns and MIDI NoteOffs (actually MIDI NoteOn with a velocity = 0). Note that the Arduino serial commands must use the Serial1() and not Serial() commands in order to work with the Pro Micro. This program enables polyphonic performance of any synthesizer with MIDI input.

This program sets up a straightforward, obvious use of the keyboard as a traditional music performance device. However, the MIDI out on the keyboard is not limited to just NoteOn and Off commands or even single note playback per

key. Any MIDI command can be used in the Arduino programming or any combination or timing of notes can be programmed, all triggered by key action in any way you like. Programmers are encouraged to go wild and create a unique MIDI performance device using this program as merely a starting point in your programming explorations.

PulseWave Audio Output

A third program uses the Arduino tone() command to create a monophonic music keyboard with a squarewave output on the phone jack. The output originates from a single pin on the Arduino (A3) oscillating from 0 to 5 volts. An array sets up values for key frequency, to be fed to the tone() command. The values in the array can be set for any desired tuning, or even to improve upon the somewhat off attempt at equal temperament tuning.

To facilitate more pleasing fast legato playing, the tone is programmed to change frequency but not abruptly turn off unless a key scan senses that no key is pressed.

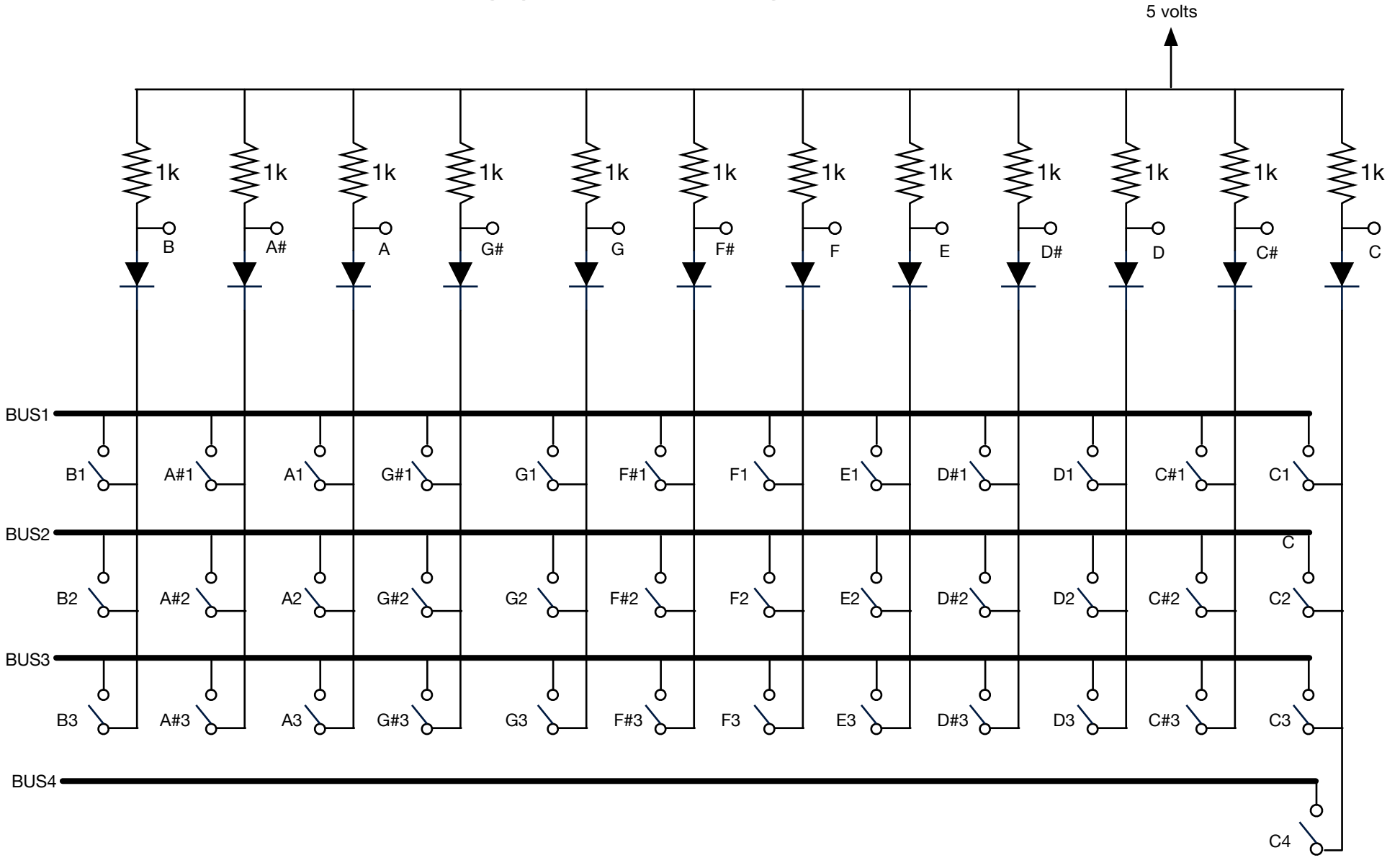
Hardware

The keyboard's USB cable is standard with a micro USB connection at the Microprocessor board. A cable tie secures it at the Arduino input.

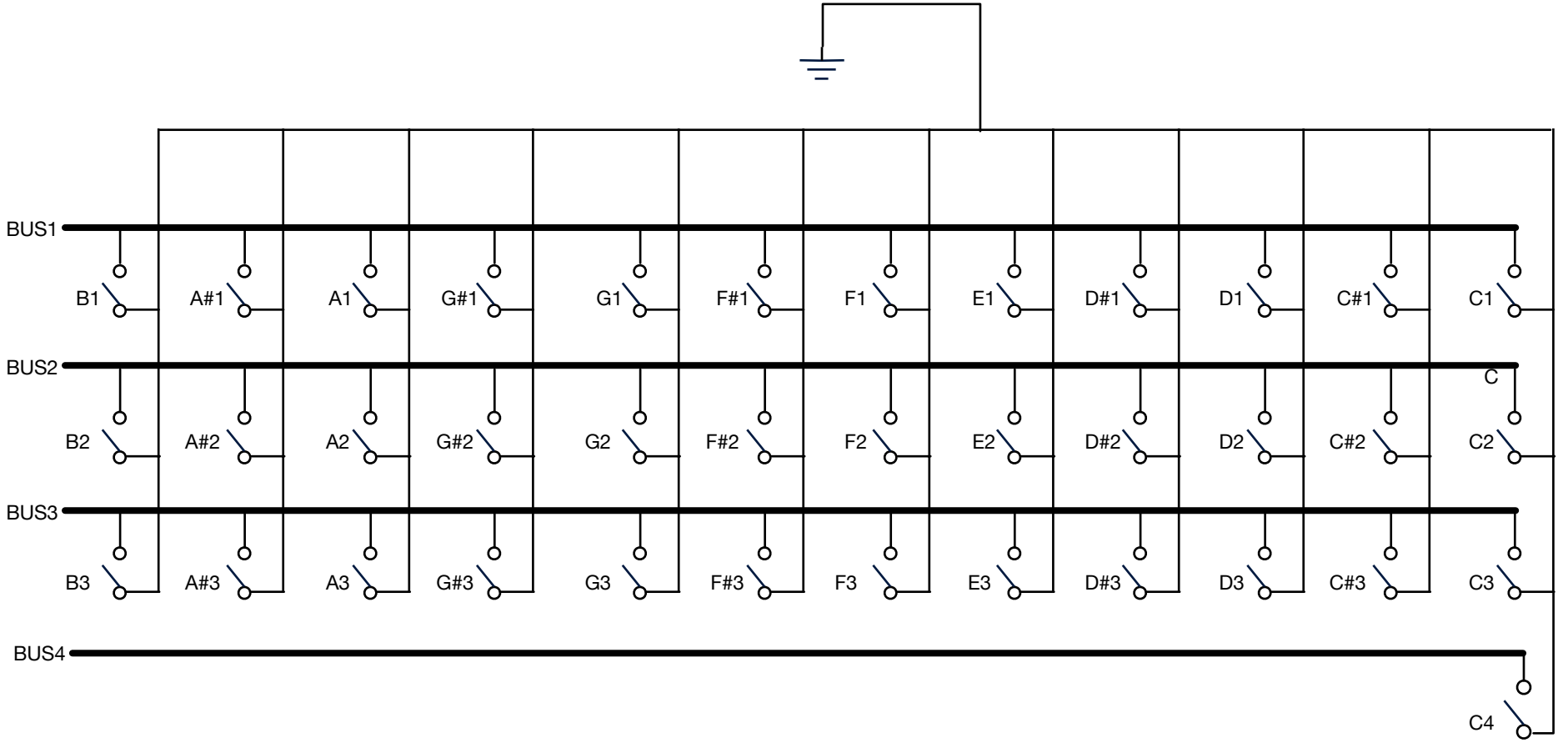
Programs can be built and loaded into the keyboard' microprocessor using an Arduino application provided free for that purpose. The program loading happens through a USB connection to any computer running the application.

Any program loaded into the Arduino will remain there and start up whenever power is applied to the board. Power to the keyboard can come from another computer through the USB cable, or you can use any USB power plug or USB charger to provide the power.

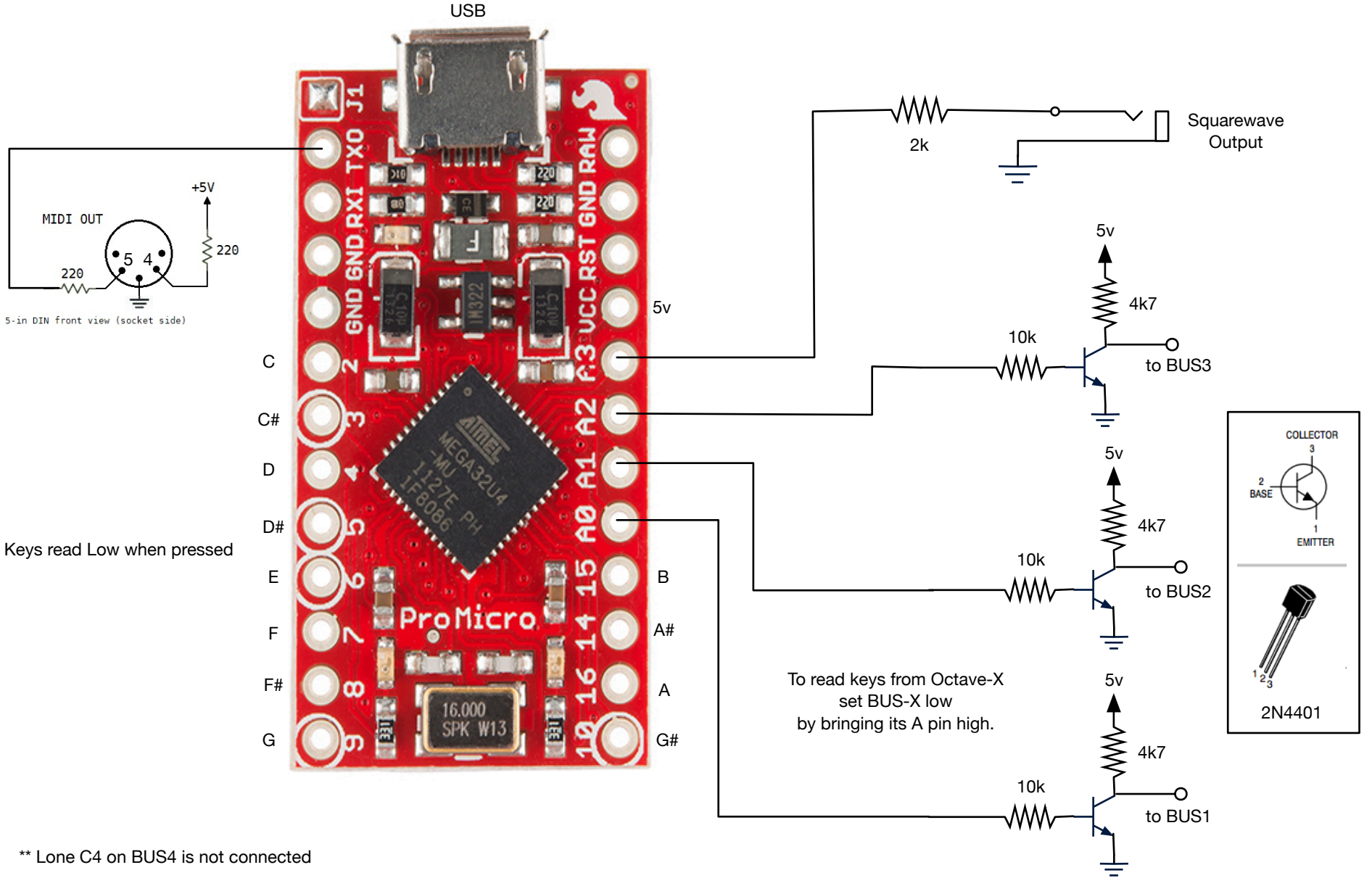
3-OCTAVE KEYBOARD



3-OCTAVE KEYBOARD
Second Bus



3-OCTAVE KEYBOARD
Arduino Pro Micro



```
/*
 *                     3-Octave Music Keyboard
 *                           Using Serial
 *
 * Only 12 keys at a time are enabled and readable from 12 digital input pins.
 * Each of the 3 octaves are enabled by writing a High to A0, A1, or A2 pins.
 * Keys read LOW when pressed.
*/
//Array to store Current read key values
byte keys[36] = {1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                 1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                 1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1 };

//Array to store Last Saved key values
byte keys_saved[36] = {1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                       1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                       1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1 };

//Array to store tests for those keys that have changed, for key bounce delay.
byte change[36] = {0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
                   0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
                   0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0 };

int octave = 0;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void setup() {

  Serial.begin(57600);

  pinMode(2, INPUT);  //12 inputs from an octave of the keyboard
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(14, INPUT);
  pinMode(15, INPUT);
  pinMode(16, INPUT);
  pinMode(A0, OUTPUT);  //Used to enable each of 3 octaves
  pinMode(A1, OUTPUT);
```

```
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT); //Used for audio output

  digitalWrite(A0, LOW);  //Disable octaves with a LOW
  digitalWrite(A1, LOW);
  digitalWrite(A2, LOW);

}  //end of setup

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loop() {

// key debounce scheme:
//when a key state change is detected, the change is noted but no action is
//taken until the change is found to be still there on the next scan of keys

  keyRD();  //read all 36 keys and store the key state in keys[]

  for (int i = 0; i < 36; i++){  //check each key for changes

          if (keys[i] != keys_saved[i]){  //if key state has changed

//(keys != saved) & (change is TRUE) =>
//Send Key# and State, Reset Change, Load Saved

                  if (change[i]){
                    Serial.print(i);
                    Serial.print(" ");
                    Serial.print(keys[i]);
                    Serial.println();
                    keys_saved[i] = keys[i];//load key_saved with new state
                    change[i] = 0;
                  }  //end of if change

//(keys != saved) & (change is FALSE) =>
//First time around, Do nothing but mark change=TRUE

                  else { change[i] = 1; } //end of else change

          }  //end of if keys

//(keys == saved) =>
//no change, or keys bounced away from a change => set change=FALSE
```

```
          else { change[i] = 0; }     //end of if


  } //end of for

  delay(5);  //debouncing delay

} //end of loop

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void keyRD(){ //Function to read all 36 keys and enter state into keys array

  for (int x = 0; x < 3; x++){

    if (x == 0) {digitalWrite(A0, HIGH); }  //Enable one of 3 octaves
    else if (x ==1) {digitalWrite(A1, HIGH); }
    else {digitalWrite(A2, HIGH); }

        octave = x * 12;
        keys[octave + 0] = digitalRead(2);
        keys[octave + 1] = digitalRead(3);
        keys[octave + 2] = digitalRead(4);
        keys[octave + 3] = digitalRead(5);
        keys[octave + 4] = digitalRead(6);
        keys[octave + 5] = digitalRead(7);
        keys[octave + 6] = digitalRead(8);
        keys[octave + 7] = digitalRead(9);
        keys[octave + 8] = digitalRead(10);
        keys[octave + 9] = digitalRead(16);
        keys[octave + 10] = digitalRead(14);
        keys[octave + 11] = digitalRead(15);

        digitalWrite(A0, LOW);
        digitalWrite(A1, LOW);
        digitalWrite(A2, LOW);

   } //end of for



}  //end of keyRD() function
```

```
/*
 *                      3-Octave Music Keyboard
 *                            Using MIDI
 *
 * Only 12 keys at a time are enabled and readable from 12 digital input pins.
 * Each of the 3 octaves are enabled by writing a High to A0, A1, or A2 pins.
 * Keys read LOW when pressed.
*/
//Array to store Current read key values
byte keys[36] = {1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                 1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                 1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1 };

//Array to store Last Saved key values
byte keys_saved[36] = {1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                       1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                       1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1 };
//Array to store tests for those keys that have changed, for key bounce delay.
byte change[36] = {0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
                   0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
                   0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0 };

int octave = 0;
int note = 48;  //MIDI note an octave below middle C
int velocity = 0;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void setup() {

  Serial1.begin(31250);  //MIDI Baud Rate

  pinMode(2, INPUT);  //12 inputs from an octave of the keyboard
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(14, INPUT);
  pinMode(15, INPUT);
  pinMode(16, INPUT);
  pinMode(A0, OUTPUT);  //Used to enable each of 3 octaves
```

```
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT); //Used for audio output

  digitalWrite(A0, LOW);  //Disable octaves with a LOW
  digitalWrite(A1, LOW);
  digitalWrite(A2, LOW);

}  //end of setup

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~

void loop() {

// key debounce scheme:  when a key state change is detected, the change is noted
but no action
// is taken until the change is found to be still there on the next scan of keys

  keyRD();  //read all 36 keys and store the key state in keys[]

  for (int i = 0; i < 36; i++){  //check each key for changes

          if (keys[i] != keys_saved[i]){  //if key state has changed

//(keys != saved) & (change is TRUE) => Send MIDI noteOn, Reset Change, Load Saved
                  if (change[i]){

                    note = i + 48;
                    if (keys[i] == 0) {velocity=64; }
                    else {velocity=0; }  //noteOff

                    noteOn(0x90, note, velocity);

                    keys_saved[i] = keys[i];  //load key_saved with new key state
                    change[i] = 0;

                  }  //end of if change

//(keys != saved) & (change is FALSE) =>  First time around, Do nothing but mark
change=TRUE
                  else { change[i] = 1; } //end of else change

          }  //end of if keys
```

```
//(keys == saved) => no change, or keys bounced away from a change => set
change=FALSE
          else {change[i] = 0; }     //end of if


  } //end of for

  delay(5);  //debouncing delay

} //end of loop

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~

void keyRD(){       //Function to read all 36 keys and enter state into keys array

  for (int x = 0; x < 3; x++){

    if (x == 0) {digitalWrite(A0, HIGH); }  //Enable one of 3 octaves
    else if (x == 1) {digitalWrite(A1, HIGH); }
    else {digitalWrite(A2, HIGH); }

        octave = x * 12;
        keys[octave + 0] = digitalRead(2);
        keys[octave + 1] = digitalRead(3);
        keys[octave + 2] = digitalRead(4);
        keys[octave + 3] = digitalRead(5);
        keys[octave + 4] = digitalRead(6);
        keys[octave + 5] = digitalRead(7);
        keys[octave + 6] = digitalRead(8);
        keys[octave + 7] = digitalRead(9);
        keys[octave + 8] = digitalRead(10);
        keys[octave + 9] = digitalRead(16);
        keys[octave + 10] = digitalRead(14);
        keys[octave + 11] = digitalRead(15);

        digitalWrite(A0, LOW);
        digitalWrite(A1, LOW);
        digitalWrite(A2, LOW);

   } //end of for
}  //end of keyRD() function

//  plays a MIDI note.  Doesn't check to see that
//  cmd is greater than 127, or that data values are  less than 127
```

```
//  noteOFF is a noteOn with velocity = 0.

void noteOn(int cmd, int pitch, int velocity) {
  Serial1.write(cmd);
  Serial1.write(pitch);
  Serial1.write(velocity);
}
```

```
/*
 *                    3-Octave Music Keyboard
 *                         Using Tone()
 *
 * Only 12 keys at a time are enabled and readable from 12 digital input pins.
 * Each of the 3 octaves are enabled by writing a High to A0, A1, or A2 pins.
 * Keys read LOW when pressed.
*/
//Array to store Current read key values
byte keys[36] = {1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                 1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                 1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1 };

//Array to store Last Saved key values
byte keys_saved[36] = {1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                       1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1,
                       1, 1, 1, 1,  1, 1, 1, 1,  1, 1, 1, 1 };

//Array to store tests for those keys that have changed, for key bounce delay.
byte change[36] = {0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
                   0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0,
                   0, 0, 0, 0,  0, 0, 0, 0,  0, 0, 0, 0 };

//Array to store frequency values for the Tone() function
int note[36] = {131, 139, 147, 156,  165, 175, 185, 196,  208, 220, 233, 247,
                262, 277, 294, 311,  330, 349, 370, 392,  415, 440, 466, 494,
                523, 554, 587, 622,  659, 698, 740, 784,  831, 880, 932, 988
};

int octave = 0;
int nokey = 1;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void setup() {

  pinMode(2, INPUT);  //12 inputs from an octave of the keyboard
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
```

```
  pinMode(14, INPUT);
  pinMode(15, INPUT);
  pinMode(16, INPUT);
  pinMode(A0, OUTPUT);  //Used to enable each of 3 octaves
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT); //Used for audio output

  digitalWrite(A0, LOW);  //Disable octaves with a LOW
  digitalWrite(A1, LOW);
  digitalWrite(A2, LOW);

}  //end of setup

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loop() {

// key debounce scheme:  when a key state change is detected,
// the change is noted but no action is
// taken until the change is found to be still there on the next scan of keys

  keyRD();  //read all 36 keys and store the key state in keys[]
  nokey=1;

  for (int i = 0; i < 36; i++){  //check each key for changes

          if (keys[i] == 0) { nokey=0; } //Note if any keys are pressed.

          if (keys[i] != keys_saved[i]){  //if key state has changed

//(keys != saved) & (change is TRUE) =>
//Send a tone() command, Reset Change, Load Saved

                  if (change[i]){

                    if (keys[i] == 0) { tone(A3, note[i]); }

                    keys_saved[i] = keys[i]; //load key_saved with new state
                    change[i] = 0;

                  }  //end of if change

//(keys != saved) & (change is FALSE) =>
//First time around, Do nothing but mark change=TRUE
```

```
                else { change[i] = 1; } //end of else change

          }  //end of if keys

//(keys == saved) =>
//no change, or keys bounced away from a change => set change=FALSE

          else {change[i] = 0; }     //end of if


  } //end of for

  if (nokey==1) { noTone(A3); }  //If no keys were pressed, turn off tone
  delay(5);  //debouncing delay

} //end of loop

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void keyRD(){ //Function to read all 36 keys and enter state into keys array

  for (int x = 0; x < 3; x++){

    if (x == 0) {digitalWrite(A0, HIGH); }  //Enable one of 3 octaves
    else if (x == 1) {digitalWrite(A1, HIGH); }
    else {digitalWrite(A2, HIGH); }

        octave = x * 12;
        keys[octave + 0] = digitalRead(2);
        keys[octave + 1] = digitalRead(3);
        keys[octave + 2] = digitalRead(4);
        keys[octave + 3] = digitalRead(5);
        keys[octave + 4] = digitalRead(6);
        keys[octave + 5] = digitalRead(7);
        keys[octave + 6] = digitalRead(8);
        keys[octave + 7] = digitalRead(9);
        keys[octave + 8] = digitalRead(10);
        keys[octave + 9] = digitalRead(16);
        keys[octave + 10] = digitalRead(14);
        keys[octave + 11] = digitalRead(15);

        digitalWrite(A0, LOW);
        digitalWrite(A1, LOW);
        digitalWrite(A2, LOW);
```

```
   } //end of for
}  //end of keyRD() function
```