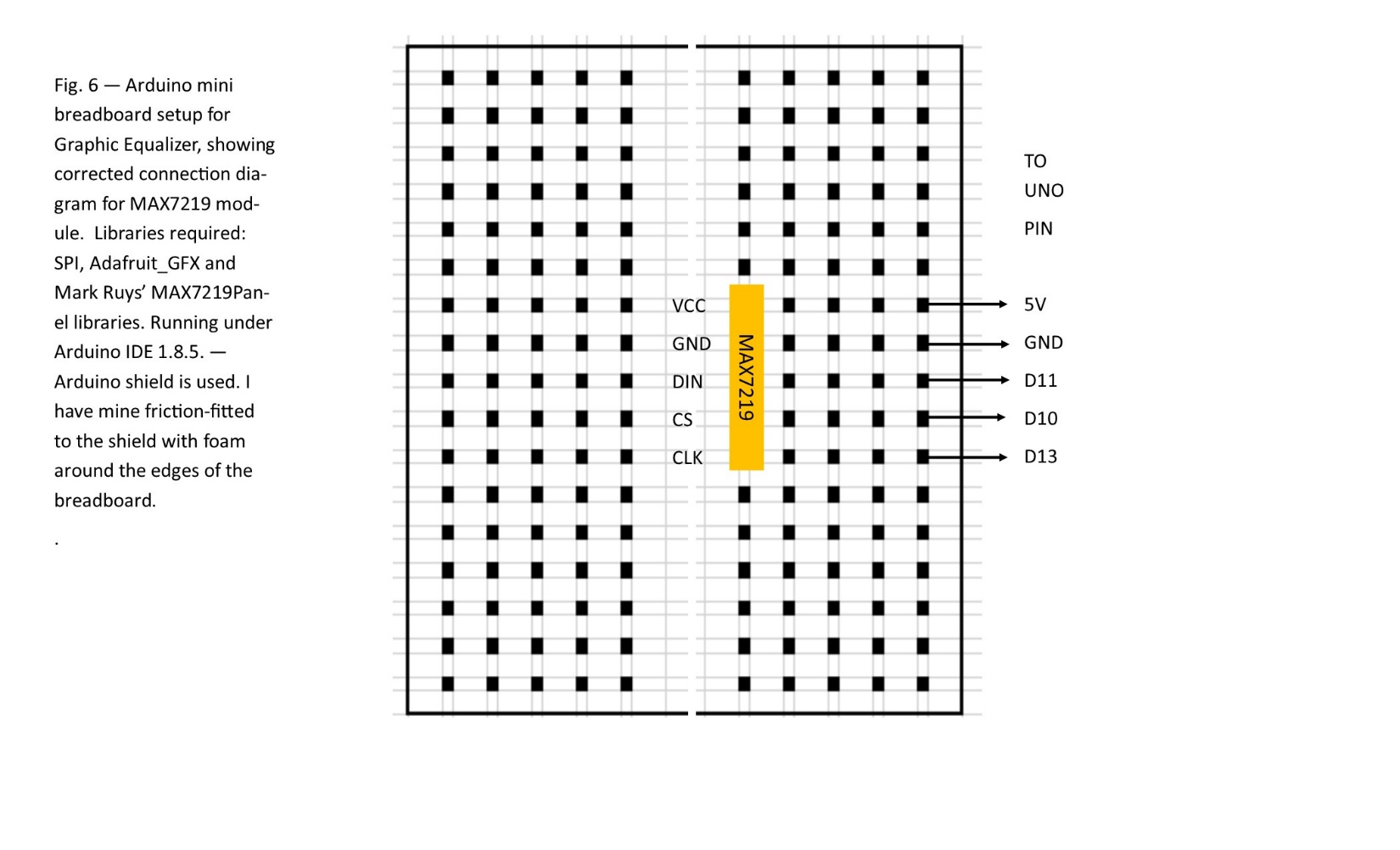
**Project 6 – 3-16 Band Graphic Equalizer with MAX7219**

Sketch simulates a graphic equalizer using a 32x8 LED array controlled by MAX7219s. This particular one has a variable **wid**, which allows the user to choose various number of bands for the equalizer using a lookup table. The equalizer ranges from 3 bands when **wid** is set to its maximum value of 11, to 16 bands when it is set to its minimum value of 2. It can also display 4,5,6,7,8 or 11 bands, depending on the value of **wid**.



Code for Graphic Equalizer sketch

/\*

\* 7219 Equalizer by Bill Jenkins

\* Rev. 08/10/2018

\* Hardware - 8x32 LED array with MAX7219

\* This sketch simulated the output of a graphic equalizer

\* The number of bands is set by adjusting the value of the wid parameter - see below

\*/

#include <SPI.h>

#include <Adafruit\_GFX.h>

#include <Max72xxPanel.h>

// Connection chart for 8x32 LED array

// MAX7219[0] UNO

// ---------- ---

// VCC 5V

// GND GND

// DIN D11

// CS D10

// CLK D13

// set pin and matrix parameters

int pinCS = 10; // Attach CS to this pin, DIN to MOSI and CLK to SCK (cf http://arduino.cc/en/Reference/SPI )

int numberOfHorizontalDisplays = 4;

int numberOfVerticalDisplays = 1;

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

// variables for drawing the graph. The value of wid determines the number of bands displayed.

// each bar is two pixels wide.

int x,y,wid; // for (col,row) addressing of pixels, wid is number of columns apart for bars (range 2-11)

float pix[32], adj; // variables for determining whether a pixel is filled in or not

void setup()

{

matrix.setIntensity(5); // Use a value between 0 and 15 for brightness

// Adjust to your own needs

matrix.setPosition(0, 0, 0); // The first display is at <0, 0>

matrix.setPosition(1, 0, 1);

matrix.setPosition(2, 0, 2);

matrix.setPosition(3, 0, 3);

matrix.setRotation(0, 1); // rotate 90 deg clockwise

matrix.setRotation(1, 1); // rotate 90 deg clockwise

matrix.setRotation(2, 1); // rotate 90 deg clockwise

matrix.setRotation(3, 1); // rotate 90 deg clockwise

// Boot Screen

matrix.fillScreen(HIGH); // draw TEST in reverse color text

matrix.drawChar(4,1,'T',LOW,HIGH,1);

matrix.drawChar(10,1,'E',LOW,HIGH,1);

matrix.drawChar(16,1,'S',LOW,HIGH,1);

matrix.drawChar(22,1,'T',LOW,HIGH,1);

matrix.write();

delay(1000);

matrix.fillScreen(HIGH); // all pixels high

matrix.write();

delay(1000);

matrix.fillScreen(LOW); // draw PASS in normal text

matrix.drawChar(4,0,'P',HIGH,LOW,1);

matrix.drawChar(10,0,'A',HIGH,LOW,1);

matrix.drawChar(16,0,'S',HIGH,LOW,1);

matrix.drawChar(22,0,'S',HIGH,LOW,1);

matrix.write();

delay(1000);

randomSeed(analogRead(0)); // set random seed for random display, or you can feed data values to each instance of pix[]

// set number of bands by adjusting the value of wid

// wid value # bands

// 2 16

// 3 11

// 4 8

// 5 7

// 6 6

// 7 5

// 8 4

// 9 4

// 10 4

// 11 3

wid=3; // set number of bands as per the table above

for (x=0;x<32;x++) // set height of all bands = 4 pixels initially

{

pix[x]=4.0;

}

}

void loop()

{

matrix.fillScreen(LOW); // clear screen

for (x=0; x<32; x+=wid) // column level stuff goes here

{

adj=(random(0,8)-3.5)/4.0; // calculate increment of bar for band, it will be [-1, 0 or 1]

pix[x]=pix[x]+adj; // calculate new bar height

if (pix[x]<0) pix[x]=0; // correct bar height if it's too short

if (pix[x]>8) pix[x]=8; // correct bar height if it's too tall

for (y=0;y<8;y++) // pixel level stuff goes here

{

if (pix[x]>y) // if true, set pixel HIGH

{

matrix.drawPixel(x,7-y,HIGH); // bars are two pixls wide. y pixels start with y[0] at the top and y[7] at the bottom

matrix.drawPixel(x+1,7-y,HIGH); // flipping the write location keeps them in proper graph format

}

}

}

matrix.write();

delay(80);

}