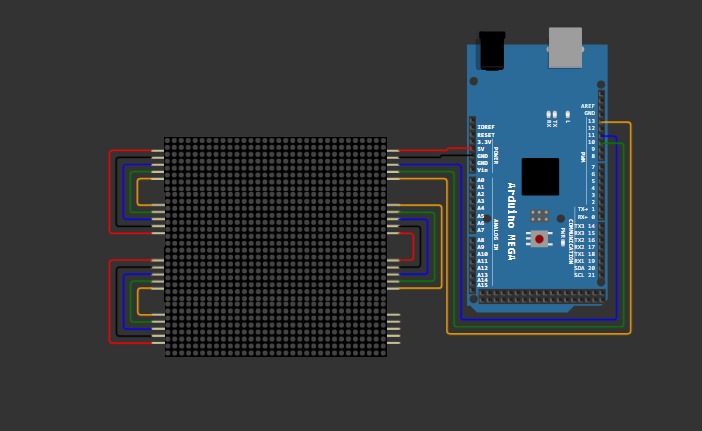
**Layers in LCD using Arduino**



#define CLK 13

#define DIN 11

#define CS 10

#define X\_SEGMENTS 4

#define Y\_SEGMENTS 4

#define NUM\_SEGMENTS (X\_SEGMENTS \* Y\_SEGMENTS)

// a framebuffer to hold the state of the entire matrix of LEDs

// laid out in raster order, with (0, 0) at the top-left

byte fb[8 \* NUM\_SEGMENTS];

void shiftAll(byte send\_to\_address, byte send\_this\_data)

{

digitalWrite(CS, LOW);

for (int i = 0; i < NUM\_SEGMENTS; i++) {

shiftOut(DIN, CLK, MSBFIRST, send\_to\_address);

shiftOut(DIN, CLK, MSBFIRST, send\_this\_data);

}

digitalWrite(CS, HIGH);

}

void setup() {

Serial.begin(115200);

pinMode(CLK, OUTPUT);

pinMode(DIN, OUTPUT);

pinMode(CS, OUTPUT);

// Setup each MAX7219

shiftAll(0x0f, 0x00); //display test register - test mode off

shiftAll(0x0b, 0x07); //scan limit register - display digits 0 thru 7

shiftAll(0x0c, 0x01); //shutdown register - normal operation

shiftAll(0x0a, 0x0f); //intensity register - max brightness

shiftAll(0x09, 0x00); //decode mode register - No decode

}

void loop() {

static int16\_t sx1 = 15 << 8, sx2 = sx1, sy1, sy2;

sx1 = sx1 - (sy1 >> 6);

sy1 = sy1 + (sx1 >> 6);

sx2 = sx2 - (sy2 >> 5);

sy2 = sy2 + (sx2 >> 5);

static byte travel = 0;

travel--;

byte \*dst = fb;

byte output = 0;

int8\_t x\_offset = (sx1 >> 8) - X\_SEGMENTS \* 4;

int8\_t y\_offset = (sx2 >> 8) - Y\_SEGMENTS \* 4;

uint8\_t screenx, screeny, xroot, yroot;

uint16\_t xsumsquares, ysumsquares, xnextsquare, ynextsquare;

int8\_t x, y;

// offset the origin in screen space

x = x\_offset;

y = y\_offset;

ysumsquares = x\_offset \* x\_offset + y \* y;

yroot = int(sqrtf(ysumsquares));

ynextsquare = yroot\*yroot;

// Quadrant II (top-left)

screeny = Y\_SEGMENTS \* 8;

while (y < 0 && screeny) {

x = x\_offset;

screenx = X\_SEGMENTS \* 8;

xsumsquares = ysumsquares;

xroot = yroot;

if (x < 0) {

xnextsquare = xroot \* xroot;

while (x < 0 && screenx) {

screenx--;

output <<= 1;

output |= ((xroot + travel) & 8) >> 3;

if (!(screenx & 7))

\*dst++ = output;

xsumsquares += 2 \* x++ + 1;

if (xsumsquares < xnextsquare)

xnextsquare -= 2 \* xroot-- - 1;

}

}

// Quadrant I (top right)

if (screenx) {

xnextsquare = (xroot + 1) \* (xroot + 1);

while (screenx) {

screenx--;

output <<= 1;

output |= ((xroot + travel) & 8) >> 3;

if (!(screenx & 7))

\*dst++ = output;

xsumsquares += 2 \* x++ + 1;

if (xsumsquares >= xnextsquare)

xnextsquare += 2 \* ++xroot + 1;

}

}

ysumsquares += 2 \* y++ + 1;

if (ysumsquares < ynextsquare)

ynextsquare -= 2 \* yroot-- - 1;

screeny--;

}

// Quadrant III (bottom left)

ynextsquare = (yroot + 1) \* (yroot + 1);

while (screeny) {

x = x\_offset;

screenx = X\_SEGMENTS \* 8;

xsumsquares = ysumsquares;

xroot = yroot;

if (x < 0) {

xnextsquare = xroot \* xroot;

while (x < 0 && screenx) {

screenx--;

output <<= 1;

output |= ((xroot + travel) & 8) >> 3;

if (!(screenx & 7))

\*dst++ = output;

xsumsquares += 2 \* x++ + 1;

if (xsumsquares < xnextsquare)

xnextsquare -= 2 \* xroot-- - 1;

}

}

// Quadrant IV (bottom right)

if (screenx) {

xnextsquare = (xroot + 1) \* (xroot + 1);

while (screenx--) {

output <<= 1;

output |= ((xroot + travel) & 8) >> 3;

if (!(screenx & 7))

\*dst++ = output;

xsumsquares += 2 \* x++ + 1;

if (xsumsquares >= xnextsquare)

xnextsquare += 2 \* ++xroot + 1;

}

}

ysumsquares += 2 \* y++ + 1;

if (ysumsquares >= ynextsquare)

ynextsquare += 2 \* ++yroot + 1;

screeny--;

}

show();

}

void set\_pixel(uint8\_t x, uint8\_t y, uint8\_t mode) {

byte \*addr = &fb[x / 8 + y \* X\_SEGMENTS];

byte mask = 128 >> (x % 8);

switch (mode) {

case 0: // clear pixel

\*addr &= ~mask;

break;

case 1: // plot pixel

\*addr |= mask;

break;

case 2: // XOR pixel

\*addr ^= mask;

break;

}

}

void safe\_pixel(uint8\_t x, uint8\_t y, uint8\_t mode) {

if ((x >= X\_SEGMENTS \* 8) || (y >= Y\_SEGMENTS \* 8))

return;

set\_pixel(x, y, mode);

}

// turn off every LED in the framebuffer

void clear() {

byte \*addr = fb;

for (byte i = 0; i < 8 \* NUM\_SEGMENTS; i++)

\*addr++ = 0;

}

// send the raster order framebuffer in the correct order

// for the boustrophedon layout of daisy-chained MAX7219s

void show() {

for (byte row = 0; row < 8; row++) {

digitalWrite(CS, LOW);

byte segment = NUM\_SEGMENTS;

while (segment--) {

byte x = segment % X\_SEGMENTS;

byte y = segment / X\_SEGMENTS \* 8;

byte addr = (row + y) \* X\_SEGMENTS;

if (segment & X\_SEGMENTS) { // odd rows of segments

shiftOut(DIN, CLK, MSBFIRST, 8 - row);

shiftOut(DIN, CLK, LSBFIRST, fb[addr + x]);

} else { // even rows of segments

shiftOut(DIN, CLK, MSBFIRST, 1 + row);

shiftOut(DIN, CLK, MSBFIRST, fb[addr - x + X\_SEGMENTS - 1]);

}

}

digitalWrite(CS, HIGH);

}

}