// Digital Guy Fawkes Mask

// Arduino Nano

// Keyestudio HT16K33 8x8 Matrix

// VCC +5v - +5v

// GND - GND

// SCL Clock Pin A5

// SDA Data Pin A4

// Keyestudio 8x8 Matrix - Left Eye (No Solder Tabs)

// Keyestudio 8x8 Matrix - Right Eye (No Solder Tabs)

// Keyestudio 8x8 Matrix - Left Part of Mouth (Solder Tabs A0)

// Keyestudio 8x8 Matrix - Middle Part of Mouth (Solder Tabs A2)

// Keyestudio 8x8 Matrix - Right Part of Mouth (Solder Tabs A0 and A2)

#include <Arduino.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include "Adafruit\_LEDBackpack.h"

#define MATRIX\_EYES 0

#define MATRIX\_MOUTH\_LEFT 1

#define MATRIX\_MOUTH\_MIDDLE 2

#define MATRIX\_MOUTH\_RIGHT 3

Adafruit\_8x8matrix matrix[4] = {

 Adafruit\_8x8matrix(), Adafruit\_8x8matrix(),

 Adafruit\_8x8matrix(), Adafruit\_8x8matrix() };

static const uint8\_t matrixAddr[] = { 0x70, 0x71, 0x72, 0x73 };

static const uint8\_t PROGMEM

 blinkImg[][8] = {

 { B00111100,

 B01111110,

 B11111111,

 B11111111,

 B11111111,

 B11111111,

 B01111110,

 B00111100 },

 { B00000000,

 B01111110,

 B11111111,

 B11111111,

 B11111111,

 B11111111,

 B01111110,

 B00111100 },

 { B00000000,

 B00000000,

 B00111100,

 B11111111,

 B11111111,

 B11111111,

 B00111100,

 B00000000 },

 { B00000000,

 B00000000,

 B00000000,

 B00111100,

 B11111111,

 B01111110,

 B00011000,

 B00000000 },

 { B00000000,

 B00000000,

 B00000000,

 B00000000,

 B10000001,

 B01111110,

 B00000000,

 B00000000 } },

 mouthImg[][24] = {

 { B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B01111111, B11111111, B11111110,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000 },

 { B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00111111, B11111111, B11111100,

 B00000111, B00000000, B11100000,

 B00000000, B11111111, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000 },

 { B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00111111, B11111111, B11111100,

 B00001000, B00000000, B00010000,

 B00000110, B00000000, B01100000,

 B00000001, B11000011, B10000000,

 B00000000, B00111100, B00000000,

 B00000000, B00000000, B00000000 },

 { B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00111111, B11111111, B11111100,

 B00100000, B00000000, B00000100,

 B00010000, B00000000, B00001000,

 B00001100, B00000000, B00110000,

 B00000011, B10000001, B11000000,

 B00000000, B01111110, B00000000 },

 { B00000000, B00000000, B00000000,

 B00000000, B00111100, B00000000,

 B00011111, B11000011, B11111000,

 B00000011, B10000001, B11000000,

 B00000000, B01111110, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000 },

 { B00000000, B00111100, B00000000,

 B00000000, B11000011, B00000000,

 B00001111, B00000000, B11110000,

 B00000001, B00000000, B10000000,

 B00000000, B11000011, B00000000,

 B00000000, B00111100, B00000000,

 B00000000, B00000000, B00000000,

 B00000000, B00000000, B00000000 } };

uint8\_t

 blinkIndex[] = { 1, 2, 3, 4, 3, 2, 1 },

 blinkCountdown = 100,

 gazeCountdown = 75,

 gazeFrames = 50,

 mouthPos = 0,

 mouthCountdown = 10;

int8\_t

 eyeX = 3, eyeY = 3,

 newX = 3, newY = 3,

 dX = 0, dY = 0;

void setup() {

 randomSeed(analogRead(A0));

 for(uint8\_t i=0; i<4; i++) {

 matrix[i].begin(matrixAddr[i]);

 }

}

void loop() {

 matrix[MATRIX\_EYES].clear();

 matrix[MATRIX\_EYES].drawBitmap(0, 0,

 blinkImg[

 (blinkCountdown < sizeof(blinkIndex)) ?

 blinkIndex[blinkCountdown] :

 0

 ], 8, 8, LED\_ON);

 if(--blinkCountdown == 0) blinkCountdown = random(5, 180);

 if(--gazeCountdown <= gazeFrames) {

 matrix[MATRIX\_EYES].fillRect(

 newX - (dX \* gazeCountdown / gazeFrames),

 newY - (dY \* gazeCountdown / gazeFrames),

 2, 2, LED\_OFF);

 if(gazeCountdown == 0) {

 eyeX = newX; eyeY = newY;

 do {

 newX = random(7); newY = random(7);

 dX = newX - 3; dY = newY - 3;

 } while((dX \* dX + dY \* dY) >= 10);

 dX = newX - eyeX;

 dY = newY - eyeY;

 gazeFrames = random(3, 15);

 gazeCountdown = random(gazeFrames, 120);

 }

 } else {

 matrix[MATRIX\_EYES].fillRect(eyeX, eyeY, 2, 2, LED\_OFF);

 }

 drawMouth(mouthImg[mouthPos]);

 if(--mouthCountdown == 0) {

 mouthPos = random(6);

 mouthCountdown = ((mouthPos == 0) && (random(5) == 0)) ?

 random(10, 40) :

 random(2, 8);

 }

 for(uint8\_t i=0; i<4; i++) matrix[i].writeDisplay();

 delay(20);

}

void drawMouth(const uint8\_t \*img) {

 for(uint8\_t i=0; i<3; i++) {

 matrix[MATRIX\_MOUTH\_LEFT + i].clear();

 matrix[MATRIX\_MOUTH\_LEFT + i].drawBitmap(i \* -8, 0, img, 24, 8, LED\_ON);

 }

}

// 'roboface' example sketch for Adafruit I2C 8x8 LED backpacks:

//

// www.adafruit.com/products/870 www.adafruit.com/products/1049

// www.adafruit.com/products/871 www.adafruit.com/products/1050

// www.adafruit.com/products/872 www.adafruit.com/products/1051

// www.adafruit.com/products/959 www.adafruit.com/products/1052

//

// Requires Adafruit\_LEDBackpack and Adafruit\_GFX libraries.

// For a simpler introduction, see the 'matrix8x8' example.

//

// This sketch demonstrates a couple of useful techniques:

// 1) Addressing multiple matrices (using the 'A0' and 'A1' solder

// pads on the back to select unique I2C addresses for each).

// 2) Displaying the same data on multiple matrices by sharing the

// same I2C address.

//

// This example uses 5 matrices at 4 addresses (two share an address)

// to animate a face:

//

// 0 0

//

// 1 2 3

//

// The 'eyes' both display the same image (always looking the same

// direction -- can't go cross-eyed) and thus share the same address

// (0x70). The three matrices forming the mouth have unique addresses

// (0x71, 0x72 and 0x73).

//

// The face animation as written is here semi-random; this neither

// generates nor responds to actual sound, it's simply a visual effect

// Consider this a stepping off point for your own project. Maybe you

// could 'puppet' the face using joysticks, or synchronize the lips to

// audio from a Wave Shield (see wavface example). Currently there are

// only six images for the mouth. This is often sufficient for simple

// animation, as explained here:

// http://www.idleworm.com/how/anm/03t/talk1.shtml

//

// Adafruit invests time and resources providing this open source code,

// please support Adafruit and open-source hardware by purchasing

// products from Adafruit!

//

// Written by P. Burgess for Adafruit Industries.

// BSD license, all text above must be included in any redistribution.