Feather RP2040 DVI Video Synth

Created by Liz Clark

https://learn.adafruit.com/feather-rp2040-dvi-video-synth

Last updated on 2023-08-29 04:54:22 PM EDT
# Table of Contents

**Overview**  
- Parts

**Circuit Diagram**  

**Installation**  

**Code the Video Synth**  
- Project Code
- Declarations
- The Setup
- The Loop
- Helpers and Functions

**Acrylic Enclosure**  

**Wiring**  
- Wire the Potentiometers
- Potentiometers Meet Proto
- Buttons Meet Proto

**Assembly**  
- FeatherWing Doubler
- Potentiometers Meet Acrylic
- Top Stand-Offs
- Bottom Stand-Offs
- Potentiometer Knobs

**Use**  

**Radar Target**  

**Static Triangles**  

**Synthwave**  

**Wavy Lines**  

**Orbits**  

**More Synth Creations**
Overview

In this project, you'll use a Feather RP2040 with DVI Output to build a fun and funky video synth. The Feather runs Arduino code written with the Adafruit Fork of the PicoDVI library. All you'll need is USB power and an HDMI monitor to start visually vibing with shapes, colors and static.

A button lets you advance through the five different programmed animations.
There are four potentiometers that control various parameters for each animation, including RGB color values and speed.

A special `shebang` button unleashes glitchy and unruly easter eggs for each animation, ranging from randomness to (simulated) black holes.

Parts

Adafruit Feather RP2040 with DVI Output Port - Works with HDMI
Wouldn't it be cool if you could display images and graphics from a microcontroller directly to an HDMI monitor or television? We think so! So we designed this RP2040 Feather that...

https://www.adafruit.com/product/5710
FeatherWing Proto - Prototyping Add-on For All Feather Boards
A Feather board without ambition is a Feather board without FeatherWings! This is the FeatherWing Proto - a prototyping add-on for all Feather boards. Using our...
https://www.adafruit.com/product/2884

FeatherWing Doubler - Prototyping Add-on For All Feather Boards
This is the FeatherWing Doubler - a prototyping add-on and more for all Feather boards. This is similar to our
https://www.adafruit.com/product/2890

Panel Mount 10K Log Potentiometer (Breadboard Friendly)
This potentiometer is a two-in-one, good in a breadboard or with a panel. It's a log taper 10K ohm potentiometer, with a grippy shaft. It's smooth and easy to...
https://www.adafruit.com/product/3391

16mm Illuminated Pushbutton - Blue Momentary
A switch is a switch, and an LED is an LED, but this LED illuminated button is a lovely combination of both! It's a medium sized button, large enough to press easily but not too...
https://www.adafruit.com/product/1477
<table>
<thead>
<tr>
<th>Item Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x Silicone Wire 30 AWG wire - various colors</td>
<td><a href="https://www.adafruit.com/product/2051">https://www.adafruit.com/product/2051</a></td>
</tr>
<tr>
<td>1 x M2.5 Screws and Stand-offs</td>
<td><a href="https://www.adafruit.com/product/3299">https://www.adafruit.com/product/3299</a></td>
</tr>
<tr>
<td>Black Nylon Machine Screw and Stand-off Set – M2.5 Thread</td>
<td></td>
</tr>
<tr>
<td>1 x USB C cable</td>
<td><a href="https://www.adafruit.com/product/5044">https://www.adafruit.com/product/5044</a></td>
</tr>
<tr>
<td>USB C to A cable - data and power</td>
<td></td>
</tr>
<tr>
<td>1 x HDMI cable</td>
<td><a href="https://www.adafruit.com/product/2422">https://www.adafruit.com/product/2422</a></td>
</tr>
<tr>
<td>6' HDMI cable for DVI output</td>
<td></td>
</tr>
<tr>
<td>1 x 720p Monitor</td>
<td><a href="https://www.adafruit.com/product/1667">https://www.adafruit.com/product/1667</a></td>
</tr>
<tr>
<td>7&quot; 1280x800 monitor</td>
<td></td>
</tr>
</tbody>
</table>

16mm Illuminated Pushbutton - Yellow Momentary
A button is a button, and an LED is a LED, but this LED illuminated button is a lovely combination of both! It's a medium sized button, large enough to press easily but not too big...
[https://www.adafruit.com/product/1441](https://www.adafruit.com/product/1441)
Circuit Diagram

The Feather RP2040 DVI and a FeatherWing Proto are both plugged into a FeatherWing Doubler.

Right Button (Next Animation Button)

- Button input to FeatherWing Proto D5
- Button GND to FeatherWing Proto GND
- Button LED to FeatherWing Proto D6
- Button LED GND to FeatherWing Proto GND

Left Button (Shebang Button)

- Button input to FeatherWing Proto D9
- Button GND to FeatherWing Proto GND
- Button LED to FeatherWing Proto D10
- Button LED GND to FeatherWing Proto GND

Potentiometer 0

- Pot left leg to FeatherWing Proto GND
- Pot wiper to FeatherWing Proto A0
- Pot right leg to FeatherWing Proto 3.3V
Potentiometer 1

- Pot left leg to FeatherWing Proto GND
- Pot wiper to FeatherWing Proto A1
- Pot right leg to FeatherWing Proto 3.3V

Potentiometer 2

- Pot left leg to FeatherWing Proto GND
- Pot wiper to FeatherWing Proto A2
- Pot right leg to FeatherWing Proto 3.3V

Potentiometer 3

- Pot left leg to FeatherWing Proto GND
- Pot wiper to FeatherWing Proto A3
- Pot right leg to FeatherWing Proto 3.3V

Installation

PicoDVI relies on the Earle Philhower III Arduino core for programming — an optional package that makes most RP2040 boards work in the Arduino environment. If you've previously followed any guides for our RP2040-based boards, you likely already have this installed...just check that you’re up to date with the latest (3.1.0 or newer).

If that sounds unfamiliar, [this guide walks through the process](#).

Once installed, the Arduino IDE Tools→Board menu will include a rollover for “Raspberry Pi RP2040 Boards,” and you can find and select whatever board type you’re using (e.g. Feather RP2040 DVI or Raspberry Pi Pico).
Next, the PicoDVI library can be installed from the Arduino Library Manager. From the Sketch menu...

Sketch ➔ Include Library ➔ Manage Libraries...

Enter “picodvi” in the search field and look for PicoDVI - Adafruit Fork in the results. Click “Install,” then “Close.”

Our version of PicoDVI depends on the Adafruit_GFX library. The Library Manager should install this automatically if not already present, but if using an older version of the Arduino IDE you might need to search for and install it manually.

This is our “fork” of the original PicoDVI project, meaning as much of the original code is preserved with minimal changes. What we’ve done is add an Arduino-compliant C++ wrapper to make this command-line library work with the friendlier Arduino IDE, and implemented simple raster framebuffers for drawing. All the original stuff is there if you want to dig in and learn, though the original examples as written won’t build in the Arduino IDE.

Here’s our fork on GitHub, and Luke Wren’s original project.

---

### Code the Video Synth

After installing the Raspberry Pi Pico/RP2040 board support package and the Adafruit Fork of the PicoDVI library, as described in the Installation page in this guide, you can prepare to upload the DVI video synth code below to the Feather RP2040 DVI.

Under Tools, select the Adafruit Feather RP2040 DVI as your target board. For Boot Stage 2, select W25Q080 QSPI /4. Then under Port, select the COM port for your Feather RP2040 DVI.
Open the project code below in the Arduino IDE and upload it to your Feather RP2040 DVI.

**Project Code**

```c
// SPDX-FileCopyrightText: 2023 Liz Clark for Adafruit Industries
// SPDX-License-Identifier: MIT

#include <PicoDVI.h>

DVIGFX16 display(DVI_RES_320x240p60, adafruit_feather_dvi_cfg);

// colors
#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
#define GREEN 0x07E0
#define CYAN 0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE 0xFFFF

// button and led pins
const int indexPin = 5;
const int ledPin = 6;
```
const int shebangPin = 9;
const int shebangLed = 10;

// pot pins
int pot0 = A0;
int pot1 = A3;
int pot2 = A2;
int pot3 = A1;

int potVal0;
int potVal1;
int potVal2;
int potVal3;

#define N_TRI 75
struct {
  int16_t pos[2]; // position (X,Y)
  int8_t  vel[2]; // velocity (X,Y)
} tri[N_TRI];

int h;
int w;

int synth_index = 0;
int last_r = 0;
int last_smo1R = 0;
int last_c = 0;
int radi;
int cir_color;
int triangle_count = 0;
int sunX = 166;
int sunY = 113;

int index_reading;
bool index_state = false;
bool is_static = false;
bool is_target = false;
bool is_wavylines = false;
bool is_synthwave = false;
bool is_orbits = false;
bool shebang_pressed = false;

int rate1 = 0;
int rate2 = 100;
int rate3 = 50;
int rate4 = 210;

int last_i = 121;
int cycle = 0;

uint8_t r,g,b;
uint16_t rgb;
uint16_t bgr;
uint16_t brg;
uint16_t gbr;

float last_x1 = 100;
float last_y1 = 120;
float last_x2 = 100;
float last_y2 = 120;
float last_x3 = 100;
float last_y3 = 120;
float last_x4 = 100;
float last_y4 = 120;

void setup() {
  if (!display.begin()) { // Blink LED if insufficient RAM
    pinMode(LED_BUILTIN, OUTPUT);
  }
}
for (;;) digitalWrite(LED_BUILTIN, (millis() / 500) & 1);
}

pinMode(indexPin, INPUT_PULLUP);
pinMode(ledPin, OUTPUT);
pinMode(shebangPin, INPUT_PULLUP);
pinMode(shebangLed, OUTPUT);

w = display.width();
h = display.height();

is_target = true;
}

void loop() {
index_reading = button_listener(indexPin, ledPin);

if (synth_index == 0) {
is_orbits = false;
if (is_target == false) {
display.fillScreen(BLACK);
is_target = true;
}
else {
animate_target();
}
}
else if (synth_index == 1) {
is_target = false;
if (is_static == false) {
display.fillScreen(BLACK);
begin_triangles();
}
else {
animate_static();
}
}
else if (synth_index == 2) {
is_static = false;
if (is_synthwave == false) {
display.fillScreen(BLACK);
begin_synthwave();
}
else {
animate_synthwave();
}
}
else if (synth_index == 3) {
is_synthwave = false;
if (is_wavylines == false) {
display.fillScreen(BLACK);
is_wavylines = true;
}
else {
animate_wavylines();
}
}
else {
is_wavylines = false;
if (is_orbits == false) {
display.fillScreen(BLACK);
draw_stars(5000);
is_orbits = true;
}
else {
animate_orbits();
}
}
int shebang_listener(int pin) {
    int z = digitalRead(pin);
    return z;
}

int button_listener(int pin, int led) {
    int i = digitalRead(pin);
    if (i == LOW and index_state == false) {
        digitalWrite(led, HIGH);
        synth_index++;
        if (synth_index > 4) {
            synth_index = 0;
        }
        index_state = true;
        delay(200);
    }
    if (i == HIGH and index_state == true) {
        index_state = false;
        delay(200);
        digitalWrite(led, LOW);
    }
    return i;
}

void begin_synthwave() {
    sunX = 166;
    sunY = 113;
    draw_gradient(0, 0, w, 130);
    display.fillCircle(sunX, sunY, 65, MAGENTA);
    display.fillCircle(sunX, sunY, 60, RED);
    display.fillRect(0, 120, w, h, BLUE);
    display.drawFastHLine(0, 120, w, WHITE);
    display.drawLine(0, 136, 34, 120, WHITE);
    display.drawLine(0, 188, 76, 120, WHITE);
    display.drawLine(34, 240, 113, 120, WHITE);
    display.drawLine(117, 240, 148, 120, WHITE);
    display.drawLine(198, 240, 182, 120, WHITE);
    display.drawLine(294, 240, 216, 120, WHITE);
    display.drawLine(320, 176, 255, 120, WHITE);
    display.drawLine(320, 133, 297, 120, WHITE);
    is_synthwave = true;
}

void animate_synthwave() {
    int s = shebang_listener(shebangPin);
    if (s == LOW) {
        digitalWrite(shebangLed, HIGH);
        shebang_pressed = true;
        for (int i=146; i > 121; i-=2) {
            sunY = sunY - 2;
            index_reading = button_listener(indexPin, ledPin);
            potVal0 = analog_map(pot0, 25, 75);
            display.fillCircle(sunX, sunY, 65, MAGENTA);
            display.fillCircle(sunX, sunY, 60, RED);
            display.fillRect(0, 120, w, h, BLUE);
            display.drawFastHLine(0, 120, w, WHITE);
            display.drawLine(0, 136, 34, 120, WHITE);
            display.drawLine(0, 188, 76, 120, WHITE);
            display.drawLine(34, 240, 113, 120, WHITE);
            display.drawLine(117, 240, 148, 120, WHITE);
            display.drawLine(198, 240, 182, 120, WHITE);
            display.drawLine(294, 240, 216, 120, WHITE);
            display.drawLine(320, 176, 255, 120, WHITE);
            display.drawLine(320, 133, 297, 120, WHITE);
        }
    }
}
display.drawLine(320, 176, 255, 120, WHITE);
display.drawLine(320, 133, 297, 120, WHITE);
display.drawFastHLine(0, 120, w, WHITE);
display.drawFastHLine(0, last_i, w, BLUE);
display.drawFastHLine(0, i, w, WHITE);
display.drawFastHLine(0, last_i+25, w, BLUE);
display.drawFastHLine(0, i+25, w, WHITE);
display.drawFastHLine(0, last_i+50, w, BLUE);
display.drawFastHLine(0, i+50, w, WHITE);
display.drawFastHLine(0, last_i+75, w, BLUE);
display.drawFastHLine(0, i+75, w, WHITE);
display.drawFastHLine(0, last_i+100, w, BLUE);
display.drawFastHLine(0, i+100, w, WHITE);
last_i = i;
if (index_reading == LOW) {
    break;
}
millisDelay(potVal0);
}
else {
    if (shebang_pressed == true) {
        sunX = 166;
        sunY = 113;
        draw_gradient(0, 0, w, 130);
        display.fillCircle(sunX, sunY, 65, MAGENTA);
        display.fillCircle(sunX, sunY, 60, RED);
        display.fillRect(0, 120, w, h, BLUE);
        display.drawFastHLine(0, 120, w, WHITE);
        shebang_pressed = false;
    }
digitalWrite(shebangLed, LOW);
}
for (int i=121; i < 146; i+=2) {
    index_reading = button_listener(indexPin, ledPin);
    potVal0 = analog_map(pot0, 25, 75);
    display.drawFastHLine(0, 120, w, WHITE);
    display.drawLine(0, 136, 34, 120, WHITE);
    display.drawLine(0, 188, 76, 120, WHITE);
    display.drawLine(34, 240, 113, 120, WHITE);
    display.drawLine(117, 240, 148, 120, WHITE);
    display.drawLine(198, 240, 182, 120, WHITE);
    display.drawLine(294, 240, 182, 120, WHITE);
    display.drawLine(320, 176, 255, 120, WHITE);
    display.drawLine(320, 133, 297, 120, WHITE);
    display.drawFastHLine(0, 120, w, BLUE);
    display.drawFastHLine(0, last_i, w, BLUE);
    display.drawFastHLine(0, i, w, WHITE);
    display.drawFastHLine(0, last_i+25, w, BLUE);
    display.drawFastHLine(0, i+25, w, WHITE);
    display.drawFastHLine(0, last_i+50, w, BLUE);
    display.drawFastHLine(0, i+50, w, WHITE);
    display.drawFastHLine(0, last_i+75, w, BLUE);
    display.drawFastHLine(0, i+75, w, WHITE);
    display.drawFastHLine(0, last_i+100, w, BLUE);
    display.drawFastHLine(0, i+100, w, WHITE);
    last_i = i;
    if (index_reading == LOW) {
        break;
    }
millisDelay(potVal0);
}
sunX = 166;
sunY = 113;

void animate_orbits() {
index_reading = button_listener(indexPin, ledPin);
potVal1 = analog_map(pot1, 1, 8);
potVal2 = analog_map(pot2, 1, 8);
potVal3 = analog_map(pot3, 1, 8);
potVal0 = analog_map(pot0, 1, 8);
int s = shebang_listener(shebangPin);
if (s == HIGH) {
    if (shebang_pressed == true) {
        display.fillScreen(BLACK);
        draw_stars(5000);
        shebang_pressed = false;
    }
    digitalWrite(shebangLed, LOW);
    rate1 = rate1 + potVal1;
    if (rate1 > 360) {
        rate1 = 1;
    }
}
rate2 = rate2 + potVal2;
if (rate2 > 361) {
    rate2 = 1;
}
rate3 = rate3 + potVal3;
if (rate3 > 361) {
    rate3 = 1;
}
rate4 = rate4 + potVal0;
if (rate4 > 361) {
    rate4 = 1;
}
display.fillCircle(160, 120, 20, YELLOW);
display.fillCircle(last_x1, last_y1, 4, 0);
float x1 = sin(2*rate1*2*3.14/100);
float y1 = cos(2*rate1*2*3.14/100);
display.fillCircle(160+45*x1, 120-45*y1, 4, BLUE);
last_x1 = 160+45*x1;
last_y1 = 120-45*y1;
display.fillCircle(last_x2, last_y2, 10, 0);
float x2 = sin(2*rate2*2*3.14/100);
float y2 = cos(2*rate2*2*3.14/100);
display.fillCircle(160+60*x2, 120-60*y2, 10, RED);
last_x2 = 160+60*x2;
last_y2 = 120-60*y2;
display.fillCircle(last_x3, last_y3, 8, 0);
float x3 = sin(2*rate3*2*3.14/100);
float y3 = cos(2*rate3*2*3.14/100);
display.fillCircle(160+95*x3, 120-95*y3, 8, MAGENTA);
last_x3 = 160+95*x3;
last_y3 = 120-95*y3;
display.fillCircle(last_x4, last_y4, 14, 0);
float x4 = sin(2*rate4*2*3.14/100);
float y4 = cos(2*rate4*2*3.14/100);
display.fillCircle(160+150*x4, 120-150*y4, 14, GREEN);
last_x4 = 160+150*x4;
last_y4 = 120-150*y4;
} else {
    digitalWrite(shebangLed, HIGH);
    shebang_pressed = true;
    int black_hole = 1;
    int falling_y1 = last_y1;
    int falling_y2 = last_y2;
    int falling_y3 = last_y3;
    int falling_y4 = last_y4;
    for (int i = 0; i < 250; i ++) {
        s = shebang_listener(shebangPin);
        draw_stars(500);
    }
display.fillCircle(last_x1, falling_y1 + i, 4, BLUE);
display.fillCircle(last_x2, falling_y2 + i, 10, RED);
display.fillCircle(last_x3, falling_y3 + i, 8, MAGENTA);
display.fillCircle(last_x4, falling_y4 + i, 14, GREEN);
display.fillCircle(160, 120, black_hole + i, BLACK);
if (s == HIGH) {
    break;
}  
millisDelay(50);
}  
display.fillScreen(BLACK);
draw_stars(5000);
}  
millisDelay(75);  
}
void begin_triangles() {
    for (int i=0; i<N_TRI; i++) {
        tri[i].pos[0] = 10 + random(w - 20);
        tri[i].pos[1] = 10 + random(h - 20);
        do {
            tri[i].vel[0] = 2 - random(5);
            tri[i].vel[1] = 2 - random(5);
        } while ((tri[i].vel[0] == 0) && (tri[i].vel[1] == 0));
    }  
is_static = true;
}
void animate_static() {
    index_reading = button_listener(indexPin, ledPin);
    potVal1 = analog_map(pot1, 0, 255);
    potVal2 = analog_map(pot2, 0, 255);
    potVal3 = analog_map(pot3, 0, 255);
    potVal0 = analog_map(pot0, 0, N_TRI);
    for (int i = 1; i < triangle_count; i++) {
        make_triangle(tri[i].pos[0], tri[i].pos[1], 20, 20, bgr);  
        tri[i].pos[0] += tri[i].vel[0];  
        if ((tri[i].pos[0] <= 0) || (tri[i].pos[0] >= display.width())) {
            tri[i].vel[0] *= -1;
        }
        tri[i].pos[1] += tri[i].vel[1];  
        if ((tri[i].pos[1] <= 0) || (tri[i].pos[1] >= display.height())) {
            tri[i].vel[1] *= -1;
        }
    }
    int s = shebang_listener(shebangPin);
    if (s == LOW) {
        digitalWrite(shebangLed, HIGH);
        triangle_count = N_TRI;
        clear_static();
    }
    else {
        digitalWrite(shebangLed, LOW);
        triangle_count = potVal0;
        draw_static(1000, rgb);
    }
}
void animate_target() {
    radi = 0;
    for (int i=0; i<240; i+=10){
        index_reading = button_listener(indexPin, ledPin);  
        potVal1 = analog_map(pot1, 0, 255);
        potVal2 = analog_map(pot2, 0, 255);
        potVal3 = analog_map(pot3, 0, 255);
        potVal0 = analog_map(pot0, 25, 75);
rgb = color_mixer(potVal1, potVal2, potVal3);
bgr = color_mixer(potVal3, potVal2, potVal1);
rgb = rgb*3;
bgr = bgr*2;
int s = shebang_listener(shebangPin);
if (s == LOW) {
    digitalWrite(shebangLed, HIGH);
    radi = random(5, 75);
    display.fillCircle(160, 120, last_r, BLACK);
} else {
    digitalWrite(shebangLed, LOW);
    radi = radi + 2;
}
display.fillCircle(160, 120, radi, rgb);
display.drawLine(160, 120, i, 240, bgr);
last_r = radi;
if (index_reading == LOW) {
    break;
}
millisDelay(potVal0);
}

for (int i=320; i>0; i-=10){
    index_reading = button_listener(indexPin, ledPin);
potVal1 = analog_map(pot1, 0, 255);
potVal2 = analog_map(pot2, 0, 255);
potVal3 = analog_map(pot3, 0, 255);
potVal0 = analog_map(pot0, 25, 75);
rgb = color_mixer(potVal1, potVal2, potVal3);
bgr = color_mixer(potVal3, potVal2, potVal1);
rgb = rgb*3;
bgr = bgr*2;
int s = shebang_listener(shebangPin);
if (s == LOW) {
    digitalWrite(shebangLed, HIGH);
    radi = random(5, 75);
    display.fillCircle(160, 120, last_r, BLACK);
} else {
    digitalWrite(shebangLed, LOW);
    radi = radi + 2;
    if (radi > 120) {
        radi = 120;
    }
}
display.fillCircle(160, 120, radi, rgb);
display.drawLine(160, 120, i, 240, bgr);
last_r = radi;
if (index_reading == LOW) {
    break;
}
millisDelay(potVal0);
}

for (int i=240; i>0; i-=10){
    index_reading = button_listener(indexPin, ledPin);
potVal1 = analog_map(pot1, 0, 255);
potVal2 = analog_map(pot2, 0, 255);
potVal3 = analog_map(pot3, 0, 255);
potVal0 = analog_map(pot0, 25, 75);
rgb = color_mixer(potVal1, potVal2, potVal3);
bgr = color_mixer(potVal3, potVal2, potVal1);
rgb = rgb*3;
bgr = bgr*2;
int s = shebang_listener(shebangPin);
if (s == LOW) {
    digitalWrite(shebangLed, HIGH);
    radi = random(5, 75);
} else {
  digitalWrite(shebangLed, LOW);
  radi = radi - 3;
}

display.fillCircle(160, 120, last_r, BLACK);
display.fillCircle(160, 120, radi, rgb);
display.drawLine(160, 120, 0, i, bgr);
last_r = radi;
if (index_reading == LOW) {
  break;
}
millisDelay(potVal0);
}

for (int i=0; i<320; i+=10){
  index_reading = button_listener(indexPin, ledPin);
  potVal1 = analog_map(pot1, 0, 255);
  potVal2 = analog_map(pot2, 0, 255);
  potVal3 = analog_map(pot3, 0, 255);
  potVal0 = analog_map(pot0, 25, 75);
  rgb = color_mixer(potVal1, potVal2, potVal3);
  bgr = color_mixer(potVal3, potVal2, potVal1);
  if (index_reading == LOW) {
    break;
  }
  millisDelay(potVal0);
}

void animate_wavylines() {
  for (int i=0; i<h; i+=10){
    index_reading = button_listener(indexPin, ledPin);
    potVal1 = analog_map(pot1, 0, 255);
    potVal2 = analog_map(pot2, 0, 255);
    potVal3 = analog_map(pot3, 0, 255);
    potVal0 = analog_map(pot0, 25, 75);
    rgb = color_mixer(potVal1, potVal2, potVal3);
    gbr = color_mixer(potVal2, potVal3, potVal1);
    if (index_reading == LOW) {
      break;
    }
    millisDelay(potVal0);
  }
}

void animate_wavylines() {
  for (int i=0; i<h; i+=10){
    index_reading = button_listener(indexPin, ledPin);
    potVal1 = analog_map(pot1, 0, 255);
    potVal2 = analog_map(pot2, 0, 255);
    potVal3 = analog_map(pot3, 0, 255);
    potVal0 = analog_map(pot0, 25, 75);
    rgb = color_mixer(potVal1, potVal2, potVal3);
    gbr = color_mixer(potVal2, potVal3, potVal1);
    int s = shebang_listener(shebangPin);
    if (s == LOW) {
      digitalWrite(shebangLed, HIGH);
      shebang_pressed = true;
      display.drawLine(random(0, w), random(0, h), w, i, rgb*3);
    } else {
      digitalWrite(shebangLed, LOW);
      if (shebang_pressed == true) {
        display.fillScreen(BLACK);
        cycle = 0;
      }
    }
  }
}
shebang_pressed = false;

} // end while

for (int i=1; i<400; i++) {
    display.drawPixel(random(0, w), random(0, h), gbr);
}

display.drawLine(0, 0, w, i, rgb*3);
}

if (index_reading == LOW) {
    break;
}

millisDelay(potVal0);
}

for (int i=w; i>0; i-=10) {
    index_reading = button_listener(indexPin, ledPin);
    potVal1 = analog_map(pot1, 0, 255);
    potVal2 = analog_map(pot2, 0, 255);
    potVal3 = analog_map(pot3, 0, 255);
    potVal0 = analog_map(pot0, 25, 75);
    rgb = color_mixer(potVal1, potVal2, potVal3);
    int s = shebang_listener(shebangPin);
    if (s == LOW) {
        digitalWrite(shebangLed, HIGH);
        shebang_pressed = true;
        display.drawLine(random(0, w), random(0, h), i, h, rgb*3);
    } else {
        digitalWrite(shebangLed, LOW);
        if (shebang_pressed == true) {
            display.fillScreen(BLACK);
            cycle = 0;
            shebang_pressed = false;
        }
        display.drawLine(0, 0, i, h, rgb*3);
    }
    if (index_reading == LOW) {
        break;
    }
    millisDelay(potVal0);
}

for (int i=0; i<w; i+=10) {
    index_reading = button_listener(indexPin, ledPin);
    potVal1 = analog_map(pot1, 0, 255);
    potVal2 = analog_map(pot2, 0, 255);
    potVal3 = analog_map(pot3, 0, 255);
    potVal0 = analog_map(pot0, 25, 75);
    brg = color_mixer(potVal3, potVal1, potVal2);
    int s = shebang_listener(shebangPin);
    if (s == LOW) {
        digitalWrite(shebangLed, HIGH);
        shebang_pressed = true;
        display.drawLine(random(0, w), random(0, h), i, h, rgb*3);
    } else {
        digitalWrite(shebangLed, LOW);
        if (shebang_pressed == true) {
            display.fillScreen(BLACK);
            cycle = 0;
            shebang_pressed = false;
        }
        display.drawLine(0, 0, i, h, brg*3);
    }
    if (index_reading == LOW) {
        break;
    }
    millisDelay(potVal0);
}

for (int i=h; i>0; i-=10) {
    index_reading = button_listener(indexPin, ledPin);
potVal1 = analog_map(pot1, 0, 255);
potVal2 = analog_map(pot2, 0, 255);
potVal3 = analog_map(pot3, 0, 255);
potVal0 = analog_map(pot0, 25, 75);

brg = color_mixer(potVal3, potVal1, potVal2);

int s = shebang_listener(shebangPin);
if (s == LOW) {
  digitalWrite(shebangLed, HIGH);
  shebang_pressed = true;
  display.drawLine(random(0, w), random(0, h), w, i, rgb*3);
}
else {
  digitalWrite(shebangLed, LOW);
  if (shebang_pressed == true) {
    display.fillScreen(BLACK);
    cycle = 0;
    shebang_pressed = false;
  }
  display.drawLine(0, 0, w, i, brg*3);
}

if (index_reading == LOW) {
  break;
}

millisDelay(potVal0);

clear_static();
cycle++;
if (cycle > 4) {
  display.fillScreen(BLACK);
  cycle = 0;
}


void make_triangle(uint16_t x1, uint16_t y1, uint16_t side_1, uint16_t side_2, uint16_t color) {
  color = color_mixer(potVal3, potVal2, potVal1);
  uint16_t x2 = x1 + side_1;
  uint16_t y2 = y1 + side_2;
  display.fillTriangle(x1, y1, x2, y2, x2, y1, color*2);
}

int analog_map(int x, int minMap, int maxMap) {
  int z = analogRead(x);
  z = map(z, 0, 1023, minMap, maxMap);
  return z;
}

uint16_t color_mixer(int int_r, int int_g, int int_b) {
  uint16_t mixed = ((int_r & 0xf8) << 8) + ((int_g & 0xfc) << 3) + (int_b >>3);
  return mixed;
}

void draw_static(int num_stars, uint16_t color_order) {
  color_order = color_mixer(potVal1, potVal2, potVal3);
  for (int i=1; i<num_stars; i++){
    display.drawPixel(random(0, 320), random(0, 240), color_order*3);
    display.drawPixel(random(0, 320), random(0, 240), 0x0000);
  }
}

void draw_stars(int num_stars) {
  for (int i=1; i<num_stars; i++){
    display.drawPixel(random(0, 320), random(0, 240), WHITE);
  }
}

void clear_static() {
  for (int i=1; i<15000; i++){
The code has four parts: the constants/variables/pin declarations, the setup, the loop and helper/graphics functions. This will be a high level overview of how the code works. The functionality of the various animation modes will be described in the Use pages.

**Declarations**

At the top of the code, if you want to change the pins that the potentiometers and buttons are assigned, you can edit these variables:

```c
// button and led pins
const int indexPin = 5;
const int ledPin = 6;
const int shebangPin = 9;
const int shebangLed = 10;

// pot pins
int pot0 = A0;
int pot1 = A3;
int pot2 = A2;
int pot3 = A1;
```

The rest of the declarations are variables and constants that are used in the loop and the graphics functions.

**The Setup**

In the setup, the `display` is started. If there is insufficient RAM, the onboard LED will blink, letting you know there is a problem. The pin directions are defined for the
buttons and button LEDs. is_target is set to true, to align with the first animation that will play in the loop.

```cpp
void setup() {
  if (!display.begin()) { // Blink LED if insufficient RAM
    pinMode(LED_BUILTIN, OUTPUT);
    for (;;) digitalWrite(LED_BUILTIN, (millis() / 500) & 1);
  }
  pinMode(indexPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
  pinMode(shebangPin, INPUT_PULLUP);
  pinMode(shebangLed, OUTPUT);
  w = display.width();
  h = display.height();
  is_target = true;
}
```

**The Loop**

In the loop, a series of if / else if statements check the value of synth_index. synth_index has a value between 0 and 4 and increases by 1 every time the button on indexPin is pressed, wrapping back around to 0 if the value is greater than 4.

When synth_index changes in value, the booleans for the various animations are changed in value and display.fillScreen(BLACK) is called to clear the screen for the next animation. Each animation is a function and some animations have a starter function (ex: begin_triangles()) with code that would be placed in setup() if the script was only running a single animation.

```cpp
void loop() {
  index_reading = button_listener(indexPin, ledPin);

  if (synth_index == 0) {
    is_orbits = false;
    if (is_target == false) {
      display.fillScreen(BLACK);
      is_target = true;
    } else {
      animate_target();
    }
  } else if (synth_index == 1) {
    is_target = false;
    if (is_static == false) {
      display.fillScreen(BLACK);
      begin_triangles();
    } else {
      animate_static();
    }
  }
```
else {
    is_wavylines = false;
    if (is_orbits == false){
        display.fillScreen(BLACK);
        draw_stars(5000);
        is_orbits = true;
    } else {
        animate_orbits();
    }
}

Helpers and Functions

There are 18 functions and helpers in the script. In summary, here is what each one does:

- **shebang_listener** - returns the result of `digitalRead` for the button on `shebangPin`. The intended functionality for that button is to initiate the special easter egg function for each animation.
- **button_listener** - returns the result of `digitalRead` for the button on `indexPin`. Additionally, it lights up the LED on `indexLed` and advances the value of `synth_index`.
- **begin_synthwave** - Starter helper for the `animate_synthwave` animation. It draws the initial shapes.
- **animate_synthwave** - Synthwave animation function. By default, it advances the `y` coordinate of the horizontal lines. If the shebang button is pressed, the `y` coordinates reverse and the `y` coordinate of the "sun" decreases, making it go up on the screen.
- **animate_orbits** - Planetary orbit animation function. Four "planets" rotate around the "sun" with "stars" in the background. The speed of each orbit is affected by the potentiometers. If the shebang button is pressed, the "sun" turns into a black hole, I mean, turns black with an increasing radius value while the `y` coordinate of the "planets" increases, making it look like they're falling.
- **begin_triangles** - Starter helper for the `animate_static` animation. It draws the initial triangles and is an adjustment of the [bouncing circles example](https://example.com) in the PicoDVI library.
- **animate_static** - Static with bouncing triangles animation function. The RGB colors and the number of triangles are controlled by the potentiometers. If the shebang button is pressed, the static disappears and all of the triangles bounce around the screen.
- **animate_target** - Breathing circle with spinning lines animation function. The RGB colors and speed are controlled by the potentiometers. The circle's radius increases and decreases as the lines are drawn around the screen from the
center of the circle. If the shebang button is pressed, then the circle's radius is randomized.

• **animate_wavylines** - Lines drawing diagonally back and forth with static animation function. The RGB colors and speed are controlled by the potentiometers. The lines originate in the top left corner of the screen and the color of the lines are inverted as they return. If the shebang button is pressed, the origin of the lines is randomized across the screen.

• **make_triangle** - A helper function creates triangles with a color defined by the potentiometers 1-3.

• **analog_map** - A helper function that takes the analog reading from a pin and then uses **map** to map the analog reading to a passed range. It returns the mapped value.

• **color_mixer** - A helper function that takes three integer values for RGB and returns a 16-bit RGB565 color.

• **draw_static** - Draws single pixels in a color and in black randomly around the screen.

• **draw_stars** - Draws single pixels in a color randomly around the screen. The same as **draw_static**, but without the black pixels.

• **clear_static** - Draws 15,000 pixels in black around the screen.

• **gradient_colors** - Creates a 16-bit RGB565 gradient between red and blue.

• **draw_gradient** - Draws the gradient with horizontal lines around a defined area of the screen.

• **millisDelay** - A helper function to create a non-blocking delay with **millis()** with the same syntax as **delay()**.

---

**Acrylic Enclosure**
Two mounting plates can be cut from acrylic with a laser cutter, CNC or your preferred manual method. The files have cutouts for the components and holes for M2.5 screws for mounting the FeatherWing Doubler.

If you aren't a fan of acrylic, or other similar material, .STL files are also available to 3D print the mounting plates.

Wiring

Wire the Potentiometers

Cut, splice and tin three pieces of wire.
Solder a wire to each of the three legs of the potentiometer.

Repeat this process for the remaining three potentiometers.

Potentiometers Meet Proto

Solder the potentiometer ground wires to the GND rail on the FeatherWing Proto.
Solder the potentiometers positive wires to the 3.3V rail on the FeatherWing Proto.

Solder potentiometer 0 wiper wire to pin A0 on the FeatherWing Proto.

Solder potentiometers 1, 2 and 3 wipers to pins A1, A2 and A3 respectively on the FeatherWing Proto.
Solder the ground pins on the buttons together. Then, solder wires to the LED anode and button output pins. The LED anode and LED ground are marked with + and - symbols.

Insert both buttons with their wires soldered into the acrylic plate.

Solder the buttons' wires to the FeatherWing Proto:

- Button GND wires to FeatherWing Proto GND rail
- Button 1 output to FeatherWing Proto D5
- Button 1 LED to FeatherWing Proto D6
- Button 2 output to FeatherWing Proto D9
- Button 2 LED to FeatherWing Proto D10
Assembly

FeatherWing Doubler

Plug the Feather RP2040 DVI and the FeatherWing Proto into a FeatherWing Doubler.
Potentiometers Meet Acrylic

Insert the potentiometers into the four mounting holes. Secure them with the included nut.

Top Stand-Offs

Insert an M2.5 stand-off and M2.5 spacer into all four of the mounting holes on the FeatherWing Proto and the two mounting holes on the USB side of the Feather RP2040 DVI.

Attach the top piece of acrylic to the stand-offs with M2.5 screws.
Bottom Stand-Offs

Add M2.5 stand-offs to the M2.5 spacers via the bottom of the FeatherWing Doubler.

Attach the bottom piece of acrylic to the stand-offs with M2.5 screws.

Potentiometer Knobs

Add some potentiometer knobs to the four potentiometers.
Use

Connect the Feather RP2040 USB port to a USB cable for power. Then, connect the DVI port to a monitor with an HDMI cable. The Feather will turn on and begin displaying the video synth animations, beginning with the target radar animation.

To change animations, press the button connected to pin 5 to advance.

There are five different animations available to play on the video synth. The following pages will go through the functionality for each animation.
The radar target animation has a circle in the center that grows and shrinks in size as a series of lines are drawn from the center around the screen.

Use potentiometers 1, 2 and 3 to adjust the colors in the animation. The circle color will be the inverse of the lines color.
Use potentiometer 0 to adjust the speed of the animation.

If you press the shebang button, the radius of the circle will be randomized.

Static Triangles
Static triangles is a riff on the bouncing circles example in the PicoDVI library. The background is filled with colored static as triangles bounce around the screen.

Use potentiometers 1, 2 and 3 to adjust the colors in the animation. The static color will be the inverse of the triangles color.

Use potentiometer 0 to affect the number of triangles in the static.

If you press the shebang button, the static disappears and the screen fills with all of the bouncing triangles.
Synthwave

The synthwave animation is more of a screensaver for vibing to the synthwave aesthetic. However, it serves as a fun example for looping motion and a gradient background.

Potentiometer 0 affects the speed of the movement of the grid. The other potentiometers are not used in this example.
If you press the shebang button, the grid will begin moving backgrounds as the sun rises into the sky.

Wavy Lines

The wavy lines animation generates lines from the to left corner back and forth across the screen with static slowly building behind the lines.
Potentiometer 1, 2 and 3 affect the color of the lines and static. The lines moving right to left will be the inverse color of the lines moving left to right.

Potentiometer 0 affects the speed of the animation.

The shebang button causes the lines to originate from random coordinates across the screen. Releasing it clears the screen.
Orbits

Orbits shows four "planets" orbiting around a "sun" with "stars" in the background.

All four of the potentiometers control the speed of the orbit for each "planet".

The shebang button causes a black hole to open and the planets to melt. Releasing the button restores order to the solar system.
More Synth Creations

If you're looking for even more video synth examples with the Feather RP2040 DVI, be sure to check out our friend TodBot's video synth experiments. He's done some great examples with lots of color and MIDI.