HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

$9 Rasberry pi VCO with Seeed XIAO RP2040 Eurorack Modular Synthesizer

13

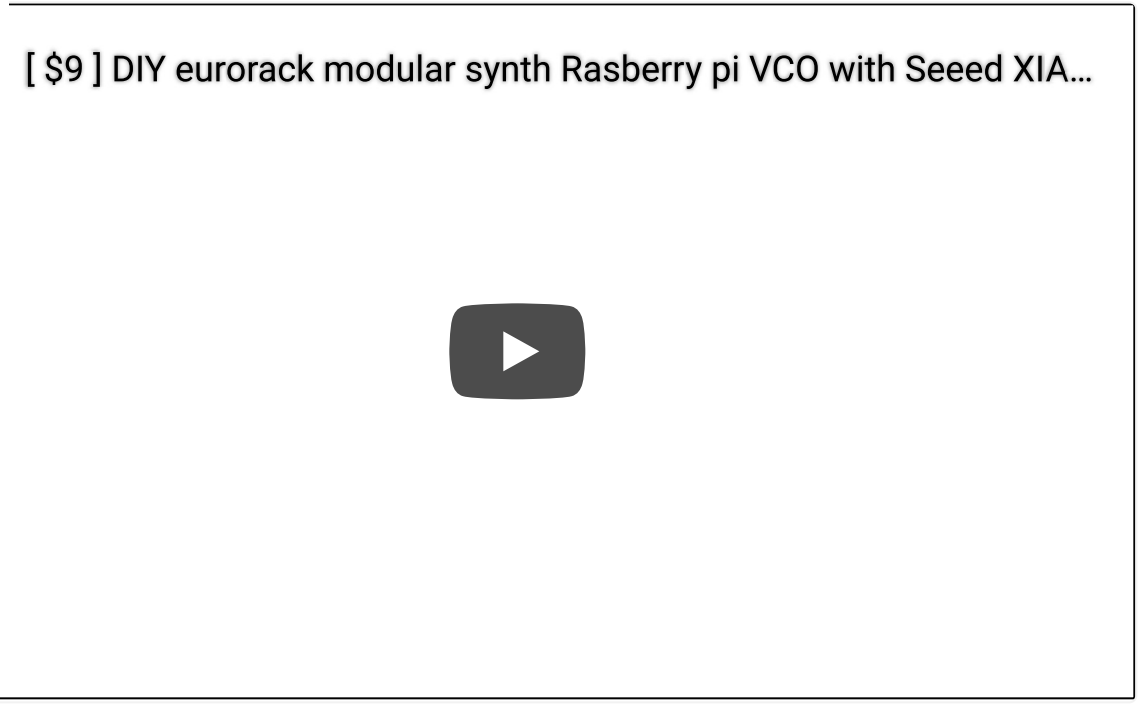
HAGIWO
July 7, 2022 3:15 AM

I made my own VCO for the modular synthesizer using the Seeed XIAO RP2040 equipped with the Rasberry pi RP2040, so that's a reminder.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow



background

This is the 53rd work of his modular synth.

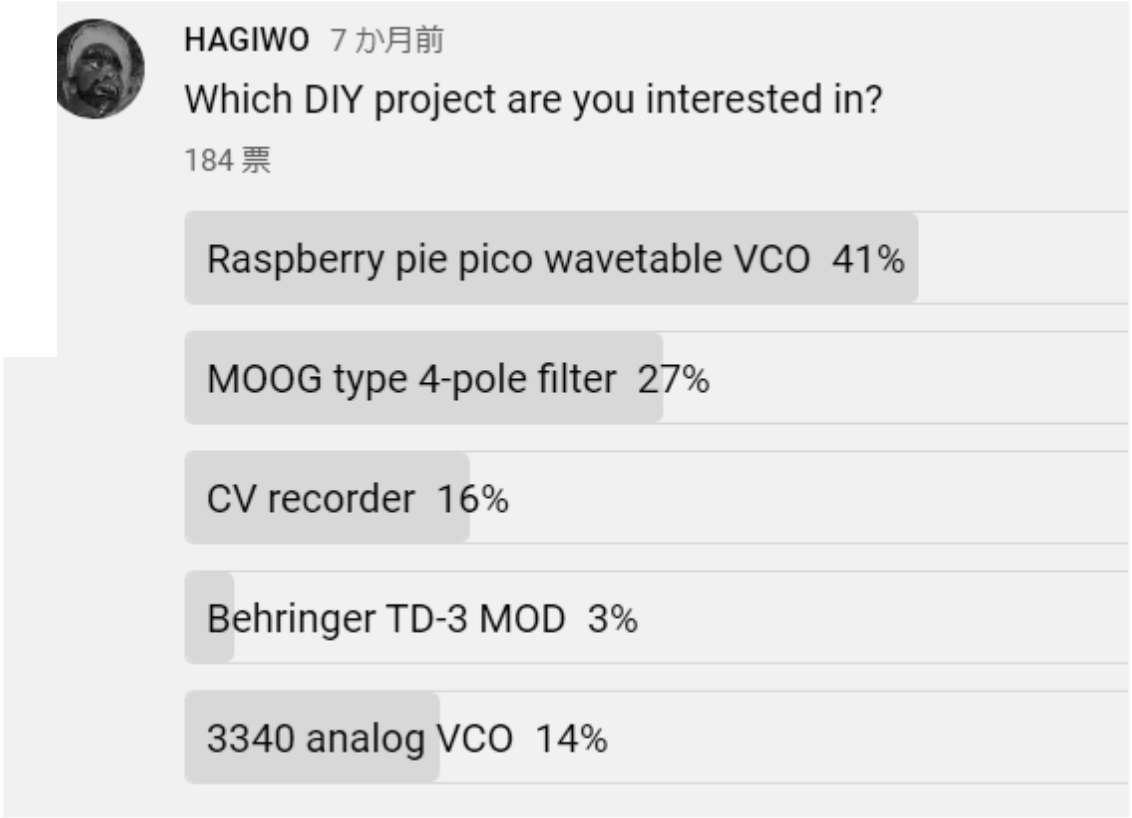
I took a poll on my Youtube channel and found the Rasberry pi pico VCO to

be the most popular. However, the Rasberry pi pico had only a 3-pin AD converter, which was somewhat inconvenient and difficult to use as a VCO.

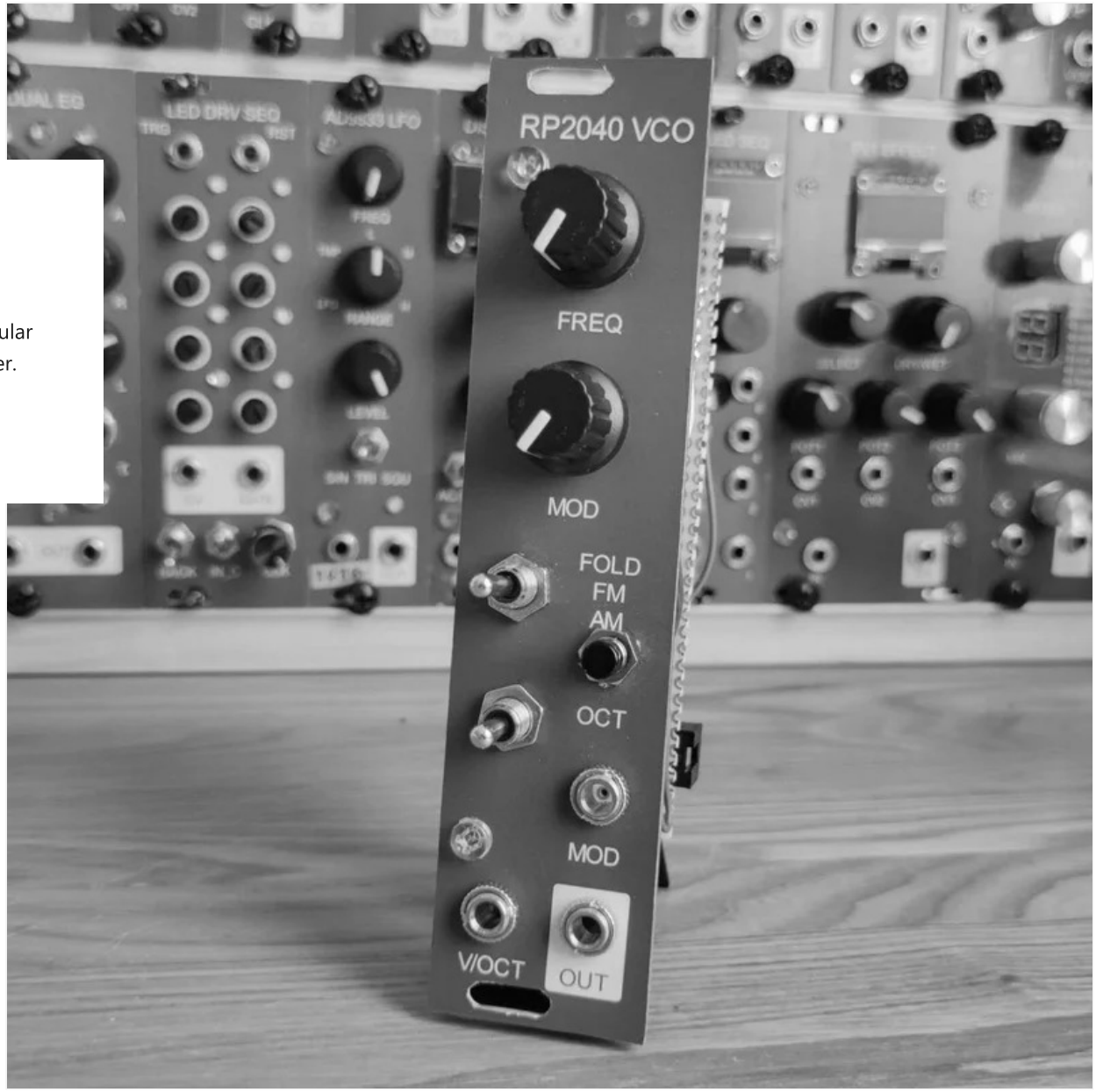
HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow



Meanwhile, a small microcomputer board called Seeed Xiao RP2040 was released by Seeed studio. Since the ADC has 4 pins, I thought that I could make a VCO with the minimum configuration, so I planned this module.



HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

Specs of the production

Eurorack Standard 3U 6HP Size

Power Supply: Operates from a single 45mA (at 5V)

5V supply.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

VCO with three modes: Wavefold, FM and AM.

st of audio output is reduced by using PWM.

ode has eight built-in waveforms.

FREQ POT : Adjust the frequency. Since it is for tuning, the frequency range that can be adjusted is narrow.

MOD POT : Adjusted the modulation effect.

MODE SW : toggle switch to switch

modeOCT SW : Switch octaves. Three ranges of -1,0,+1.

PUSH SW : Switch the timbre. 8 tones to choose from in each mode.

MOD CV : Adjusted the modulation level. The input range is 0-5V.

V/oct : Frequency controlled. The input range is 0-5V and the resolution is 10bit.

OUT: Audio output, output range is 5Vp-p.

HAGIWO

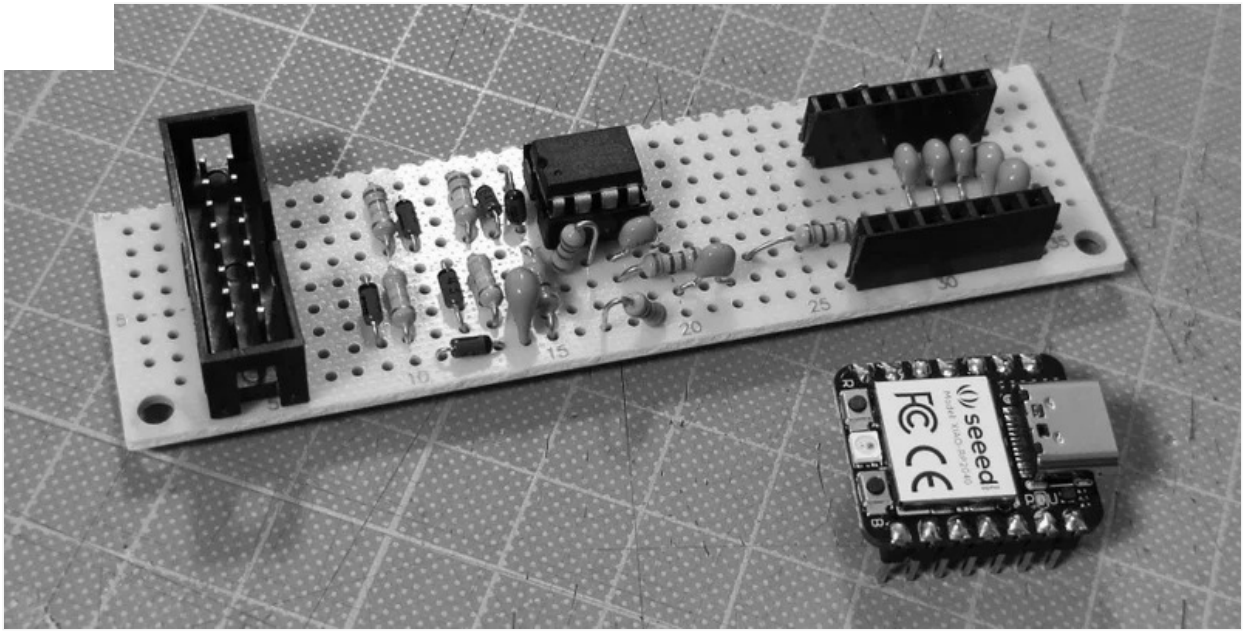
Someone who started modular synths. I'm a legacy engineer.

follow

In Wavefold mode, it can also be used as a VCO for basic waveforms.

In FM mode, it can produce industrial waveforms that include soft sounds and overtones.

Mode is intended for use in percussion of metallic sounds, and has built-ds containing non-integer overtones.



Production cost

Total amount about 1100 yen ----------

front panel 100 yen Seeed Xiao RP2040 600 yen Toggle SW 20 yen * 2pcs

Variable resistance 30 yen

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

p MPC6232 45 yen

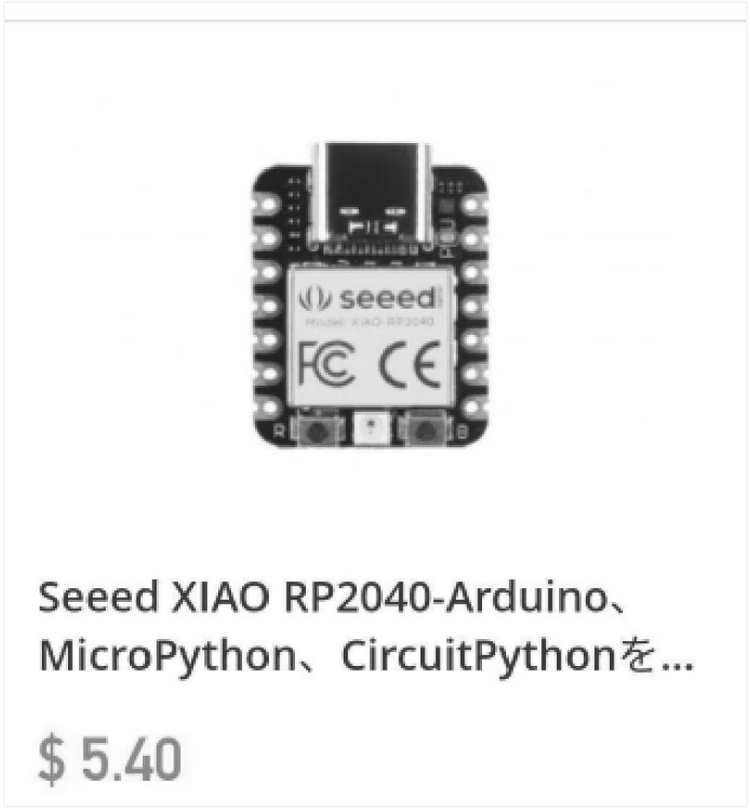
, etc. (For general-purpose components, refer to the link below)

The Seeed XIAO RP2040 is a microcontroller board equipped with the same RP2040 MCU as the Rasberry pico. The number of AD converters is 3 pi pico, while the XIAO has 4 converters, which is easy to use.

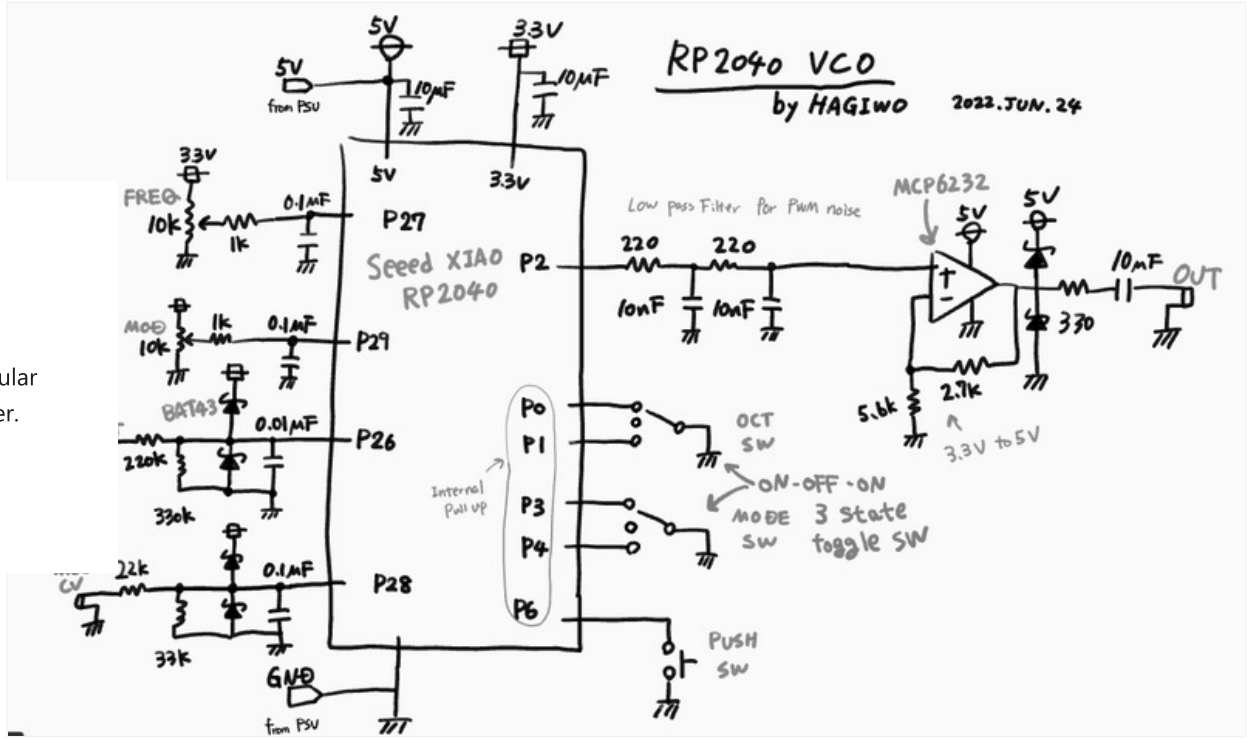
HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow



hardware



Since the ADC can only read up to the upper limit of 3.3 V, the input range is extended to 5 V by dividing the voltage with a resistor.

The error of the voltage divider of the V/oct circuit is realized by software calibration.

Since the audio output is PWM, the output circuit has a 2pole lowpass filter. It is amplified from 3.3 V p-p to 5 V p-p by an op amp.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

Because PWM is used, some harmonic noise is generated. If you are worried, you may want to tune the low-pass filter.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

JL/25 Append

Switch is a 3state switch with ON-OFF-ON.

ıre several versions of the Seeed XIAO RP2040 schematic.
The circuit diagram labeled Ver. 1.08 is old, and several circuits are different, so it should not be used as a reference. Please refer to the circuit diagram written as the latest Ver. 1.22.

software

Considerations for the Seeed XIAO RP2040

The XIAO RP2040 can be developed with the Arduino IDE.

There is also a caveat here.

WHEN I INSTALLED THE BOARD MANAGER IN THE SOFTWARE SETUP

PROCEDURE ON THE OFFICIAL WIKI, THE WRITE FAILED IN MY ENVIRONMENT.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

RP2040 with Arduino - Seeed Wiki

Product Document

edstudio.com

When I introduced the board manager introduced in adafruit, I succeeded in writing.

Program RP2040 in Arduino

In this guide, you'll learn how to install Earle Philhower's

learn.adafruit.com

同様の不具合はほかの人も経験しているようなので、もしArduino IDEで書き込みに失敗する場合は、ボードマネージャーを疑ってみるとよいかもしれない。

XIAO RP2040の洗礼を受ける - しぐれめも

5月の連休ですし、何か作りたいよねと思ってXIAO RP2040 を買いました。連休前に発注して連休前に無事GET。何気に

ure.hateblo.jp

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

出力

bleに波形を格納し、一定の周期で割り込みをて、wavetableを読み出すことで音声出力をしている。wavetableは10bit、256サンプル。

Interfaeという雑誌の2021年8月号にRasberry pi picoを使ったシンセサイザーの記事があり、そのソースコードを参考にした。私の知識では書いてあることの9割は理解できなかったのだが、割り込み処理の考え方を学ぶことができた。

ソースコードは下記のリンク先でも公開されている。感謝だ！

https://github.com/risgk/pico_synth_ex/blob/main/pico_synth_ex.c

こちらの記事も参考になる。割り込みの考え方が丁寧に解説されている。

https://blog.boochow.com/article/pico-pwm-sound.html

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

ソフトウェアキャリブレーション

V/oct回路の分圧抵抗の誤差をなくすため、キャリブレーション定数がある。

calbの値を0.7〜1.3くらいの範囲で調整することでキャリブレーションをす

ｚ

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
at calb = 1.19;//calibration for reduce resistance error
```

宣伝：オープンソースプロジェクトの支援をお願いします

DIYモジュラーシンセのオープンソースプロジェクトを継続するために、

patreonというサービスでパトロンを募集しています。

コーヒー一杯の支援をいただけると嬉しいです。

また、パトロン限定のコンテンツも配信しています。

ソースコード

粗末だが公開する。悪い点があれば指摘を貰えると嬉しい。

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
#include <hardware/pwm.h>

int k = 1;
   at AM_mod = 1;//AM modulation rate
   at f = 0;
    slice_num = 0;
   at osc_freq = 0;
   l push_sw, old_push_sw;//push sw
   at mod = 0;
    wavetable[256];//1024 resolution , 256 rate
   at mod_wavetable[256];//1st modulated wavetable
    mod2_wavetable[256];//2nd modulated wavetable
   at calb = 1.15;//calibration for reduce resistance error
     adc, freq_pot, oct_sw, mode;
int waveform = 0;
int f0 = 35;//base osc frequency
long timer = 0;//use for making AM sinewave

const static float voctpow[1230] = {//Covers 6(=1230) octaves. If it is 1230 or more, the
  0,  0.004882, 0.009765, 0.014648, 0.019531, 0.024414, 0.029296, 0.034179, 0.039062, 0.0
};
float freq_table[2048];

void setup1() {
}

void setup()
{
  pinMode(0, INPUT_PULLUP);//oct select
  pinMode(1, INPUT_PULLUP);//oct select
  pinMode(6, INPUT_PULLUP);//push sw
  pinMode(3, INPUT_PULLUP);//mode select
  pinMode(4, INPUT_PULLUP);//mode select
  timer = micros();

  table_set();//set wavetable
  //-------------------octave select------------------------------
  for (int i = 0; i < 1230; i++) {//Covers 6(=1230) octaves. If it is 1230 or more, the c
    freq_table[i] = f0 * pow(2, (voctpow[i]));
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow


```
  }
  for (int i = 0; i < 2048 - 1230; i++) {
    freq_table[i + 1230] = 6;
  }

   /-------------------PWM setting------------------------------
   pio_set_function(2, GPIO_FUNC_PWM);// set GP2 function PWM
   lice_num = pwm_gpio_to_slice_num(2);// GP2 PWM slice

   wm_clear_irq(slice_num);
   wm_set_irq_enabled(slice_num, true);
   rq_set_exclusive_handler(PWM_IRQ_WRAP, on_pwm_wrap);
   rq_set_enabled(PWM_IRQ_WRAP, true);

   /set PWM frequency
   wm_set_clkdiv(slice_num, 1);//=sysclock/((resolution+1)*frequency)
  pwm_set_wrap(slice_num, 1023);//resolutio
  pwm_set_enabled(slice_num, true);//PWM output enable


}

void  on_pwm_wrap() {
  pwm_clear_irq(slice_num);
  f = f + osc_freq;
  if (f > 255) {
    f = 0;
  }
  int k = (int)f;
  pwm_set_chan_level(slice_num, PWM_CHAN_A, mod2_wavetable[k] + 511);
}

void loop()
{
  old_push_sw = push_sw;

  //-------------------octave select------------------------------
  if (digitalRead(0) == 1 && digitalRead(1) == 1) {
    oct_sw = 1;
  }
  else if (digitalRead(0) == 0 && digitalRead(1) == 1) {
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
  oct_sw = 3;
}
else if (digitalRead(0) == 1 && digitalRead(1) == 0) {
  oct_sw = 0;
}

 /  -------------------mode select------------------------------
 f (digitalRead(4) == 1 && digitalRead(3) == 1) {
  mode = 1;//fm

 lse if (digitalRead(4) == 0 && digitalRead(3) == 1) {
  mode = 0;//AM

 lse if (digitalRead(4) == 1 && digitalRead(3) == 0) {
  mode = 2;//wavefolder


//  -------------------frequeny calculation------------------------------
adc = analogRead(26) * calb;//Correct resistance errors
adc = constrain(adc , 0, 1225) ;//Covers 6(=1220) octaves. If it is 1230 or more, the c
freq_pot = map(analogRead(27), 0, 1023, 0, 127);
osc_freq = freq_table[adc + freq_pot]; // V/oct apply
osc_freq = 256 * osc_freq / 122070 * (1 + oct_sw);

//  -------------------mod parameter set------------------------------
if (mode == 0) {//fold
  mod = constrain(analogRead(29) + analogRead(28), 0, 1023) * 0.0036 + 0.90; //28 is CV
}
else if (mode == 1) {//FM
  mod = constrain(analogRead(29) + analogRead(28), 0, 1023) / 8;
}
else if (mode == 2) {//AM
  mod = 1023 - constrain(analogRead(29) + analogRead(28), 0, 1023) ;
}

//  -------------------push sw------------------------------
push_sw = digitalRead(6);
if (push_sw == 0 && old_push_sw == 1) {//when push sw ON
  waveform++;//change waveform
  if (waveform > 7) {
    waveform = 0;
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
  }
  if (mode == 0 || mode == 2) {
    table_set();
  }
}

 oid loop1() {//modulation
 f (mode == 0 ) {//wavefold
  for (int i = 0; i < 256; i++) {
    mod_wavetable[i] = wavetable[i] * mod;
  }
  for (int i = 0; i < 256; i++) {  //fold
    if (mod_wavetable[i] > 511 && mod_wavetable[i] < 1023 + 512) {
      mod2_wavetable[i] = 1024 - mod_wavetable[i];
    }
    else if (mod_wavetable[i] < -512 && mod_wavetable[i] > -1024 - 512) {
      mod2_wavetable[i] = -1023 - mod_wavetable[i];
    }
    else if (mod_wavetable[i] < -1024 - 511) {
      mod2_wavetable[i] = 2048 + mod_wavetable[i];
    }
    else if (mod_wavetable[i] > 1023 + 511) {
      mod2_wavetable[i] = -2047 + mod_wavetable[i];
    }
    else {
      mod2_wavetable[i] = mod_wavetable[i] ;
    }
  }
}
else if (mode == 1) {//FM
  switch  (waveform) {
    case 0:
      for (int i = 0; i < 256; i++) {  //FM1
        mod2_wavetable[i] = (sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 3 * i
      }
      break;

    case 1:
      for (int i = 0; i < 256; i++) {  //FM2
        mod2_wavetable[i] = (sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 3 * i
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
      }
      break;

    case 2:
      for (int i = 0; i < 256; i++) {  //FM3
        mod2_wavetable[i] = (sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 5 * i
      }
      break;

    case 3:
      for (int i = 0; i < 256; i++) {  //FM4
        mod2_wavetable[i] = (sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 9 * i
      }
      break;

    case 4:
      for (int i = 0; i < 256; i++) {  //FM5
        mod2_wavetable[i] = ((sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 19 *
      }
      break;

    case 5:
      for (int i = 0; i < 256; i++) {  //FM6
        mod2_wavetable[i] = ((sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 7 *
      }
      break;

    case 6:
      for (int i = 0; i < 256; i++) {  //FM7
        mod2_wavetable[i] = ((sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 13 *
      }
      break;

    case 7:
      for (int i = 0; i < 256; i++) {  //FM8
        mod2_wavetable[i] = (sin(2 * M_PI * i  / 256 + mod / 128 * sin(2 * M_PI * 11 *
      }
      break;
  }
}
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
  else if (mode == 2) {//AM
    if (timer + mod <= micros()) {
      k++;
      if (k > 63) {
        k = 0;
      }
      AM_mod = sin( 2 * M_PI * k  / 63);//make modulation sine wave
      for (int i = 0; i < 255 ; i++) {
        mod2_wavetable[i] = wavetable[i] * AM_mod;//multiply AM sine wave
      }
      timer = micros();
    }
```

```
void table_set() {//make wavetable

  if (mode == 0) { //wavefold
    switch  (waveform) {
      case 0:
        for (int i = 0; i < 256; i++) {  //saw
          wavetable[i] = i * 4 - 512;
        }
        break;

      case 1:
        for (int i = 0; i < 256; i++) {  //sin
          wavetable[i] = (sin(2 * M_PI * i  / 256)) * 511;
        }
        break;

      case 2:
        for (int i = 0; i < 128; i++) {  //squ
          wavetable[i] = 511;
          wavetable[i + 128] = -511;
        }
        break;

      case 3:
        for (int i = 0; i < 128; i++) {  //tri
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
        wavetable[i] = i * 8 - 511;
        wavetable[i + 128] = 511 - i * 8;
      }
      break;

    case 4:
      for (int i = 0; i < 128; i++) {  //oct saw
        wavetable[i] = i * 4 - 512 + i * 2;
        wavetable[i + 128] = i * 2 - 256 + i * 4;
      }
      break;

    case 5:
      for (int i = 0; i < 256; i++) {  //FM1
        wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 3 * i  / 256)) ) * 511
      }
      break;

    case 6:
      for (int i = 0; i < 256; i++) {  //FM2
        wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 7 * i  / 256))) * 511;
      }
      break;

    case 7:
      for (int i = 0; i < 256; i++) {  //FM3
        wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 4 * i  / 256 + sin(2 *
      }
      break;
  }
}
else if (mode == 2) { //AM
  switch  (waveform) {
    case 0:
      for (int i = 0; i < 256; i++) {  //saw
        wavetable[i] = (sin(2 * M_PI * i  / 256)) * 511;
      }
      break;

    case 1:
      for (int i = 0; i < 256; i++) {  //fm1
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 3 * i  / 256)) ) * 511
    }
    break;

  case 2:
    for (int i = 0; i < 256; i++) {  //fm2
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 5 * i  / 256)) ) * 511
    }
    break;

  case 3:
    for (int i = 0; i < 256; i++) {  //fm3
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 4 * i  / 256 + sin(2 *
    }
    break;

  case 4:
    for (int i = 0; i < 256; i++) {  //non-integer multiplets fm1
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 1.28 * i  / 256)) ) *
    }
    break;

  case 5:
    for (int i = 0; i < 256; i++) {  //non-integer multiplets fm2
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 3.19 * i  / 256)) ) *
    }
    break;

  case 6:
    for (int i = 0; i < 256; i++) {  //non-integer multiplets fm3
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 2.3 * i  / 256 + sin(2
    }
    break;

  case 7:
    for (int i = 0; i < 256; i++) {  //non-integer multiplets fm3
      wavetable[i] = (sin(2 * M_PI * i  / 256 + sin(2 * M_PI * 6.3 * i  / 256 + sin(2
    }
    break;
}
```

```
        }
    }
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow