



**HAGIWO**

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

**follow**



## \$9 Raspberry 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 Raspberry pi RP2040, so that's a reminder.



**HAGIWO**

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

 **follow**

[ \$9 ] DIY eurorack modular synth Raspberry pi VCO with Seeed XIA...



## background

This is the 53rd work of his modular synth.

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



**HAGIWO**

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

**follow**

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



**HAGIWO** 7 か月前

Which DIY project are you interested in?

184 票

Raspberry pie pico wavetable VCO 41%

MOOG type 4-pole filter 27%

CV recorder 16%

Behringer TD-3 MOD 3%

3340 analog VCO 14%

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.



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

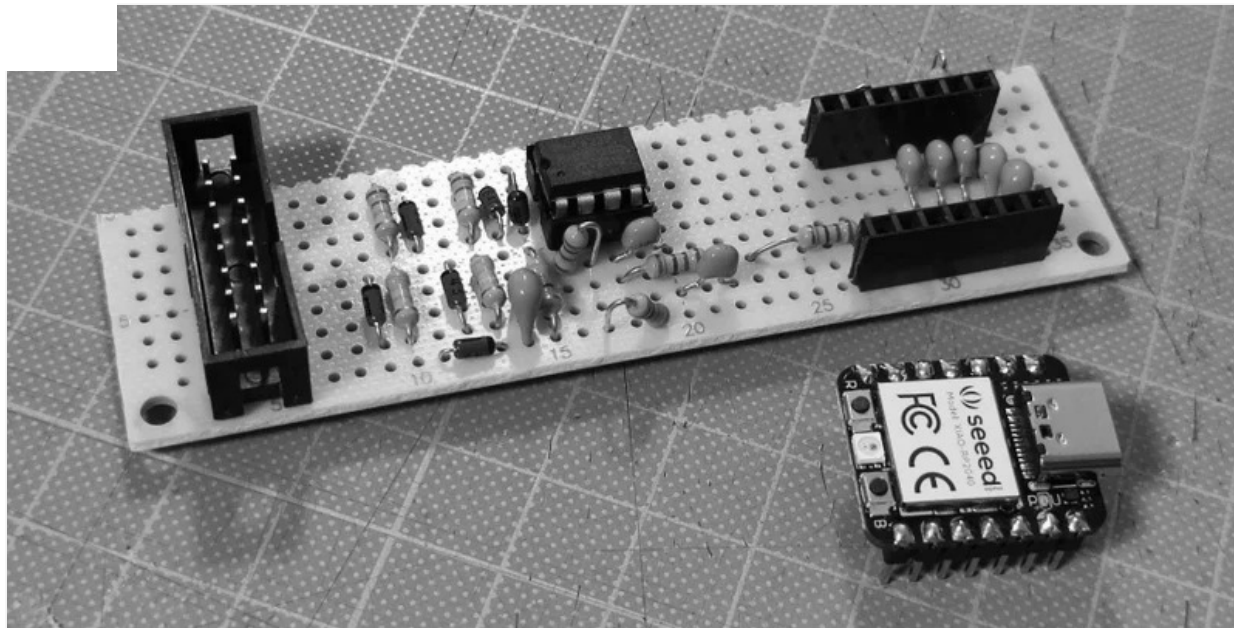
In FM mode, it can produce industrial waveforms that include soft sounds and overtones.



**HAGIWO**

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

follow



**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)

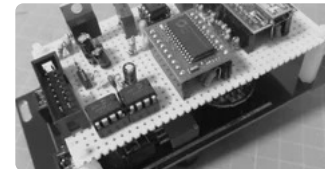
### モジュラーシンセ自作で使う安価な部品一覧

メモのついでに。適時更新。コストと品質の観点で記載、主観による。alixpressはリンクを貼ってもすぐに切れるので、画像で残す。値段は購入当時のもの。昨今は値動きが激しい。品質は私の経験値。よって母数は数...

♡ 17



HAGIWO/ハギヲ  
2021/11/19 16:18



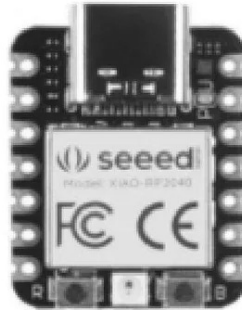
The Seeed XIAO RP2040 is a microcontroller board equipped with the same RP2040 MCU as the Raspberry pico. The number of AD converters is 3 pi pico, while the XIAO has 4 converters, which is easy to use.



**HAGIWO**

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

 **follow**



Seeed XIAO RP2040-Arduino、  
MicroPython、CircuitPythonを...

**\$ 5.40**

**hardware**







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.

ire 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  
seeedstudio.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。



HAGIWO

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

+ follow

## 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

ヒカ

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](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  
ず



HAGIWO

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

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

+ follow

## 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, 0.043945,
    0.048828, 0.053711, 0.058594, 0.063477, 0.068359, 0.073242, 0.078125, 0.083008, 0.087891, 0.092774,
    0.097656, 0.102539, 0.107422, 0.112305, 0.117188, 0.122071, 0.126954, 0.131837, 0.136719, 0.141602,
    0.146485, 0.151368, 0.15625, 0.161133, 0.166016, 0.170899, 0.175781, 0.180664, 0.185547, 0.19043,
    0.195313, 0.200195, 0.205078, 0.209961, 0.214844, 0.219727, 0.224609, 0.229492, 0.234375, 0.239258,
    0.244141, 0.249024, 0.253906, 0.258789, 0.263672, 0.268555, 0.273438, 0.278321, 0.283204, 0.288087,
    0.292969, 0.297852, 0.302735, 0.307618, 0.3125, 0.317383, 0.322266, 0.327149, 0.332031, 0.336914,
    0.341797, 0.34668, 0.351563, 0.356446, 0.361329, 0.366212, 0.371095, 0.375978, 0.380861, 0.385744,
    0.390625, 0.395508, 0.400391, 0.405274, 0.410157, 0.415039, 0.419922, 0.424805, 0.429688, 0.434571,
    0.439454, 0.444337, 0.44922, 0.454103, 0.458986, 0.463869, 0.468752, 0.473635, 0.478518, 0.483401,
    0.488284, 0.493167, 0.49805, 0.502933, 0.507816, 0.512699, 0.517582, 0.522465, 0.527348, 0.532231,
    0.537114, 0.541997, 0.54688, 0.551763, 0.556646, 0.561529, 0.566412, 0.571295, 0.576178, 0.581061,
    0.585944, 0.590827, 0.59571, 0.600593, 0.605476, 0.610359, 0.615242, 0.620125, 0.625008, 0.629891,
    0.634774, 0.639657, 0.64454, 0.649423, 0.654306, 0.659189, 0.664072, 0.668955, 0.673838, 0.678721,
    0.683604, 0.688487, 0.69337, 0.698253, 0.703136, 0.708019, 0.712902, 0.717785, 0.722668, 0.727551,
    0.732434, 0.737317, 0.7422, 0.747083, 0.751966, 0.756849, 0.761732, 0.766615, 0.771498, 0.776381,
    0.781264, 0.786147, 0.79103, 0.795913, 0.800796, 0.805679, 0.810562, 0.815445, 0.820328, 0.825211,
    0.830094, 0.834977, 0.83986, 0.844743, 0.849626, 0.854509, 0.859392, 0.864275, 0.869158, 0.874041,
    0.878924, 0.883807, 0.88869, 0.893573, 0.898456, 0.903339, 0.908222, 0.913105, 0.917988, 0.922871,
    0.927754, 0.932637, 0.93752, 0.942403, 0.947286, 0.952169, 0.957052, 0.961935, 0.966818, 0.971701,
    0.976584, 0.981467, 0.98635, 0.991233, 0.996116, 1.001, 1.005883, 1.010766, 1.015649, 1.020532,
    1.025415, 1.030298, 1.035181, 1.040064, 1.044947, 1.04983, 1.054713, 1.059596, 1.064479, 1.069362,
    1.074245, 1.079128, 1.084011, 1.088894, 1.093777, 1.09866, 1.103543, 1.108426, 1.113309, 1.118192,
    1.123075, 1.127958, 1.132841, 1.137724, 1.142607, 1.14749, 1.152373, 1.157256, 1.162139, 1.167022,
    1.171905, 1.176788, 1.181671, 1.186554, 1.191437, 1.19632, 1.201203, 1.206086, 1.210969, 1.215852,
    1.220735, 1.225618, 1.230501, 1.235384, 1.240267, 1.24515, 1.250033, 1.254916, 1.259799, 1.264682,
    1.269565, 1.274448, 1.279331, 1.284214, 1.289097, 1.29398, 1.298863, 1.303746, 1.308629, 1.313512,
    1.318395, 1.323278, 1.328161, 1.333044, 1.337927, 1.34281, 1.347693, 1.352576, 1.357459, 1.362342,
    1.367225, 1.372108, 1.376991, 1.381874, 1.386757, 1.39164, 1.396523, 1.401406, 1.406289, 1.411172,
    1.416055, 1.420938, 1.425821, 1.430704, 1.435587, 1.44047, 1.445353, 1.450236, 1.455119, 1.460002,
    1.464885, 1.469768, 1.474651, 1.479534, 1.484417, 1.4893, 1.494183, 1.499066, 1.503949, 1.508832,
    1.513715, 1.518598, 1.523481, 1.528364, 1.533247, 1.53813, 1.543013, 1.547896, 1.552779, 1.557662,
    1.562545, 1.567428, 1.572311, 1.577194, 1.582077, 1.58696, 1.591843, 1.596726, 1.601609, 1.606492,
    1.611375, 1.616258, 1.621141, 1.626024, 1.630907, 1.63579, 1.640673, 1.645556, 1.650439, 1.655322,
    1.660205, 1.665088, 1.669971, 1.674854, 1.679737, 1.68462, 1.689503, 1.694386, 1.699269, 1.704152,
    1.709035, 1.713918, 1.718801, 1.723684, 1.728567, 1.73345, 1.738333, 1.743216, 1.748099, 1.752982,
    1.757865, 1.762748, 1.767631, 1.772514, 1.777397, 1.78228, 1.787163, 1.792046, 1.796929, 1.801812,
    1.806695, 1.811578, 1.816461, 1.821344, 1.826227, 1.83111, 1.835993, 1.840876, 1.845759, 1.850642,
    1.855525, 1.860408, 1.865291, 1.870174, 1.875057, 1.87994, 1.884823, 1.889706, 1.894589, 1.899472,
    1.904355, 1.909238, 1.914121, 1.919004, 1.923887, 1.92877, 1.933653, 1.938536, 1.943419, 1.948302,
    1.953185, 1.958068, 1.962951, 1.967834, 1.972717, 1.9776, 1.982483, 1.987366, 1.992249, 1.997132,
    2.002015, 2.006898, 2.011781, 2.016664, 2.021547, 2.02643, 2.031313, 2.036196, 2.041079, 2.045962,
    2.050845, 2.055728, 2.060611, 2.065494, 2.070377, 2.07526, 2.080143, 2.085026, 2.089909, 2.094792,
    2.099675, 2.104558, 2.109441, 2.114324, 2.119207, 2.12409, 2.128973, 2.133856, 2.138739, 2.143622,
    2.148505, 2.153388, 2.158271, 2.163154, 2.168037, 2.17292, 2.177803, 2.182686, 2.187569, 2.192452,
    2.197335, 2.202218, 2.207101, 2.211984, 2.216867, 2.22175, 2.226633, 2.231516, 2.236399, 2.241282,
    2.246165, 2.251048, 2.255931, 2.260814, 2.265697, 2.27058, 2.275463, 2.280346, 2.285229, 2.290112,
    2.294995, 2.299878, 2.304761, 2.309644, 2.314527, 2.31941, 2.324293, 2.329176, 2.334059, 2.338942,
    2.343825, 2.348708, 2.353591, 2.358474, 2.363357, 2.36824, 2.373123, 2.378006, 2.382889, 2.387772,
    2.392655, 2.397538, 2.402421, 2.407304, 2.412187, 2.41707, 2.421953, 2.426836, 2.431719, 2.436602,
    2.441485, 2.446368, 2.451251, 2.456134, 2.461017, 2.4659, 2.470783, 2.475666, 2.480549, 2.485432,
    2.490315, 2.495198, 2.500081, 2.504964, 2.509847, 2.51473, 2.519613, 2.524496, 2.529379, 2.534262,
    2.539145, 2.544028, 2.548911, 2.553794, 2.558677, 2.56356, 2.568443, 2.573326, 2.578209, 2.583092,
    2.587975, 2.592858, 2.597741, 2.602624, 2.607507, 2.61239, 2.617273, 2.622156, 2.627039, 2.631922,
    2.636805, 2.641688, 2.646571, 2.651454, 2.656337, 2.66122, 2.666103, 2.670986, 2.675869, 2.680752,
    2.685635, 2.690518, 2.695401, 2.700284, 2.705167, 2.71005, 2.714933, 2.719816, 2.724699, 2.729582,
    2.734465, 2.739348, 2.744231, 2.749114, 2.753997, 2.75888, 2.763763, 2.768646, 2.773529, 2.778412,
    2.783295, 2.788178, 2.793061, 2.797944, 2.802827, 2.80771, 2.812593, 2.817476, 2.822359, 2.827242,
    2.832125, 2.837008, 2.841891, 2.846774, 2.851657, 2.85654, 2.861423, 2.866306, 2.871189, 2.876072,
    2.880955, 2.885838, 2.890721, 2.895604, 2.900487, 2.90537, 2.910253, 2.915136, 2.920019, 2.924902,
    2.929785, 2.934668, 2.939551, 2.944434, 2.949317, 2.9542, 2.959083, 2.963966, 2.968849, 2.973732,
    2.978615, 2.983498, 2.988381, 2.993264, 2.998147, 3.00303, 3.007913, 3.012796, 3.017679, 3.022562,
    3.027445, 3.032328, 3.037211, 3.042094, 3.046977, 3.05186, 3.056743, 3.061626, 3.066509, 3.071392,
    3.076275, 3.081158, 3.086041, 3.090924, 3.095807, 3.10069, 3.105573, 3.110456, 3.115339, 3.120222,
    3.125105, 3.129988, 3.134871, 3.139754, 3.144637, 3.14952, 3.154403, 3.159286, 3.164169, 3.169052,
    3.173935, 3.178818, 3.183701, 3.188584, 3.193467, 3.19835, 3.203233, 3.208116, 3.213, 3.217883,
    3.222766, 3.227649, 3.232532, 3.237415, 3.242298, 3.247181, 3.252064, 3.256947, 3.26183, 3.266713,
    3.271596, 3.276479, 3.281362, 3.286245, 3.291128, 3.296011, 3.300894, 3.305777, 3.31066, 3.315543,
    3.320426, 3.325309, 3.330192, 3.335075, 3.339958, 3.344841, 3.349724, 3.354607, 3.35949, 3.364373,
    3.369256, 3.374139, 3.379022, 3.383905, 3.388788, 3.393671, 3.398554, 3.403437, 3.40832, 3.413203,
    3.418086, 3.422969, 3.427852, 3.432735, 3.437618, 3.442501, 3.447384, 3.452267, 3.45715, 3.462033,
    3.466916, 3.471799, 3.476682, 3.481565, 3.486448, 3.491331, 3.496214, 3.501097, 3.50598, 3.510863,
    3.515746, 3.520629, 3.525512, 3.530395, 3.535278, 3.540161, 3.545044, 3.549927, 3.55481, 3.559693,
    3.564576, 3.569459, 3.574342, 3.579225, 3.584108, 3.588991, 3.593874, 3.598757, 3.60364, 3.608523,
    3.613406, 3.618289, 3.623172, 3.628055, 3.632938, 3.637821, 3.642704, 3.647587, 3.65247, 3.657353,
    3.662236, 3.667119, 3.672, 3.676883, 3.681766, 3.686649, 3.691532, 3.696415, 3.701298, 3.706181,
    3.711064, 3.715947, 3.72083, 3.725713, 3.730596, 3.735479, 3.740362, 3.745245, 3.750128, 3.755011,
    3.759894, 3.764777, 3.76966, 3.774543, 3.779426, 3.784309, 3.789192, 3.794075, 3.798958, 3.803841,
    3.808724, 3.813607, 3.81849, 3.823373, 3.828256, 3.833139, 3.838022, 3.842905, 3.847788, 3.852671,
    3.857554, 3.862437, 3.86732, 3.872203, 3.877086, 3.881969, 3.886852, 3.891735, 3.896618, 3.901501,
    3.906384, 3.911267, 3.91615, 3.921033, 3.925916, 3.930799, 3.935682, 3.940565, 3.945448, 3.950331,
    3.955214, 3.960097, 3.96498, 3.969863, 3.974746, 3.979629, 3.984512, 3.989395, 3.994278, 3.999161,
    4.004044, 4.008927, 4.01381, 4.018693, 4.023576, 4.028459, 4.033342, 4.038225, 4.043108, 4.047991,
    4.052874, 4.057757, 4.06264, 4.067523, 4.072406, 4.077289, 4.082172, 4.087055, 4.091938, 4.096821,
    4.101704, 4.106587, 4.11147, 4.116353, 4.121236, 4.126119, 4.131, 4.135883, 4.140766, 4.145649,
    4.150532, 4.155415, 4.160298, 4.165181, 4.170064, 4.174947, 4.17983, 4.184713, 4.189596, 4.194479,
    4.199362, 4.204245, 4.209128, 4.214011, 4.218894, 4.223777, 4.22866, 4.233543, 4.238426, 4.243309,
    4.248192, 4.253075, 4.257958, 4.262841, 4.267724, 4.272607, 4.27749, 4.282373, 4.287256, 4.292139,
    4.297022, 4.301905, 4.306788, 4.311671, 4.316554, 4.321437, 4.32632, 4.331203, 4.336086, 4.340969,
    4.345852, 4.350735, 4.355618, 4.360501, 4.365384, 4.370267, 4.37515, 4.380033, 4.384916, 4.389799,
    4.394682, 4.399565, 4.404448, 4.409331, 4.414214, 4.419097, 4.42398, 4.428863, 4.433746, 4.438629,
    4.443512, 4.448395, 4.453278, 4.458161, 4.463044, 4.467927, 4.47281, 4.477693, 4.482576, 4.487459,
    4.492342, 4.497225, 4.502108, 4.506991, 4.511874, 4.516757, 4.52164, 4.526523, 4.531406, 4.536289,
    4.541172, 4.546055, 4.550938, 4.555821, 4.560704, 4.565587, 4.57047, 4.575353, 4.580236, 4.585119,
    4.590002, 4.594885, 4.599768, 4.604651, 4.609534, 4.614417, 4.6193, 4.624183, 4.629066, 4.633949,
    4.638832, 4.643715, 4.648598, 4.653481, 4.658364, 4.663247, 4.66813, 4.673013, 4.677896, 4.682779,
    4.687662, 4.692545, 4.697428, 4.702311, 4.707194, 4.712077, 4.71696, 4.721843, 4.726726, 4.731609,
    4.736492, 4.741375, 4.746258, 4.751141, 4.756024, 4.760907, 4.76579, 4.770673, 4.775556, 4.780439,
    4.785322, 4.790205, 4.795088, 4.800001, 4.804884, 4.809767, 4.81465, 4.819533, 4.824416, 4.829299,
    4.834182, 4.839065, 4.843948, 4.848831, 4.853714, 4.858597, 4.86348, 4.868363, 4.873246, 4.878129,
    4.883012, 4.887895, 4.892778, 4.897661, 4.902544, 4.907427, 4.91231, 4.917193, 4.922076, 4.926959,
    4.931842, 4.936725, 4.941608, 4.946491, 4.951374, 4.956257, 4.96114, 4.966023, 4.970906, 4.975789,
    4.980672, 4.985555, 4.990438, 4.995321, 5.000204, 5.005087, 5.00997, 5.014853, 5.019736, 5.024619,
    5.029502, 5.034385, 5.039268, 5.044151, 5.049034, 5.053917, 5.0588, 5.063683, 5.068566, 5.073449,
    5.078332, 5.083215, 5.088098, 5.092981, 5.097864, 5.102747, 5.10763, 5.112513, 5.117396, 5.122279,
    5.127162, 5.132045, 5.136928, 5.141811, 5.146694, 5.151577, 5.15646, 5.161343, 5.166226, 5.171109,
    5.175992, 5.180875, 5.185758, 5.190641, 5.195524, 5.200407, 5.20529, 5.210173, 5.215056, 5.219939,
    5.224822, 5.229705, 5.234588, 5.239471, 5.244354, 5.249237, 5.25412, 5.259003, 5.263886, 5.268769,
    5.273652, 5.278535, 5.283418, 5.288301, 5.293184, 5.298067, 5.30295, 5.307833, 5.312716, 5.317599,
    5.322482, 5.327365, 5.332248, 5.337131, 5.342014, 5.346897, 5.35178, 5.356663, 5.361546, 5.366429,
    5.371312, 5.376195, 5.381078, 5.385961, 5.390844, 5.395727, 5.40061, 5.405493, 5.410376, 5.415259,
    5.420142, 5.425025, 5.429908, 5.434791, 5.439674, 5.444557, 5.44944, 5.454323, 5.459206, 5.464089,
    5.468972, 5.473855, 5.478738, 5.483621, 5.488504, 5.493387, 5.49827, 5.503153, 5.508036, 5.512919,
    5.517802, 5.522685, 5.527568, 5.532451, 5.537334, 5.542217, 5.5471, 5.551983, 5.556866, 5.561749,
    5.566632, 5.571515, 5.576398, 5.581281, 5.586164, 5.591047, 5.59593, 5.600813, 5.605696, 5.610579,
    5.615462, 5.620345, 5.625228, 5.630111, 5.635, 5.639883, 5.644766, 5.649649, 5.654532, 5.659415,
    5.664298, 5.669181, 5.674064, 5.678947, 5.68383, 5.688713, 5.693596, 5.698479, 5.703362, 5.708245,
    5.713128, 5.718011, 5.722894, 5.727777, 5.73266, 5.737543, 5.742426, 5.747309
```



**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); // resolution
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

// -----frequency 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 Cl
}
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 / 256)) * 1023 + 512) / 2;  
            }  
            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 / 256)) * 1023 + 512) / 2;  
            }  
            break;  
    }  
}
```



**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 * i
  )
  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 * i
  )
  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 * i
  )
  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 * i
  )
  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**