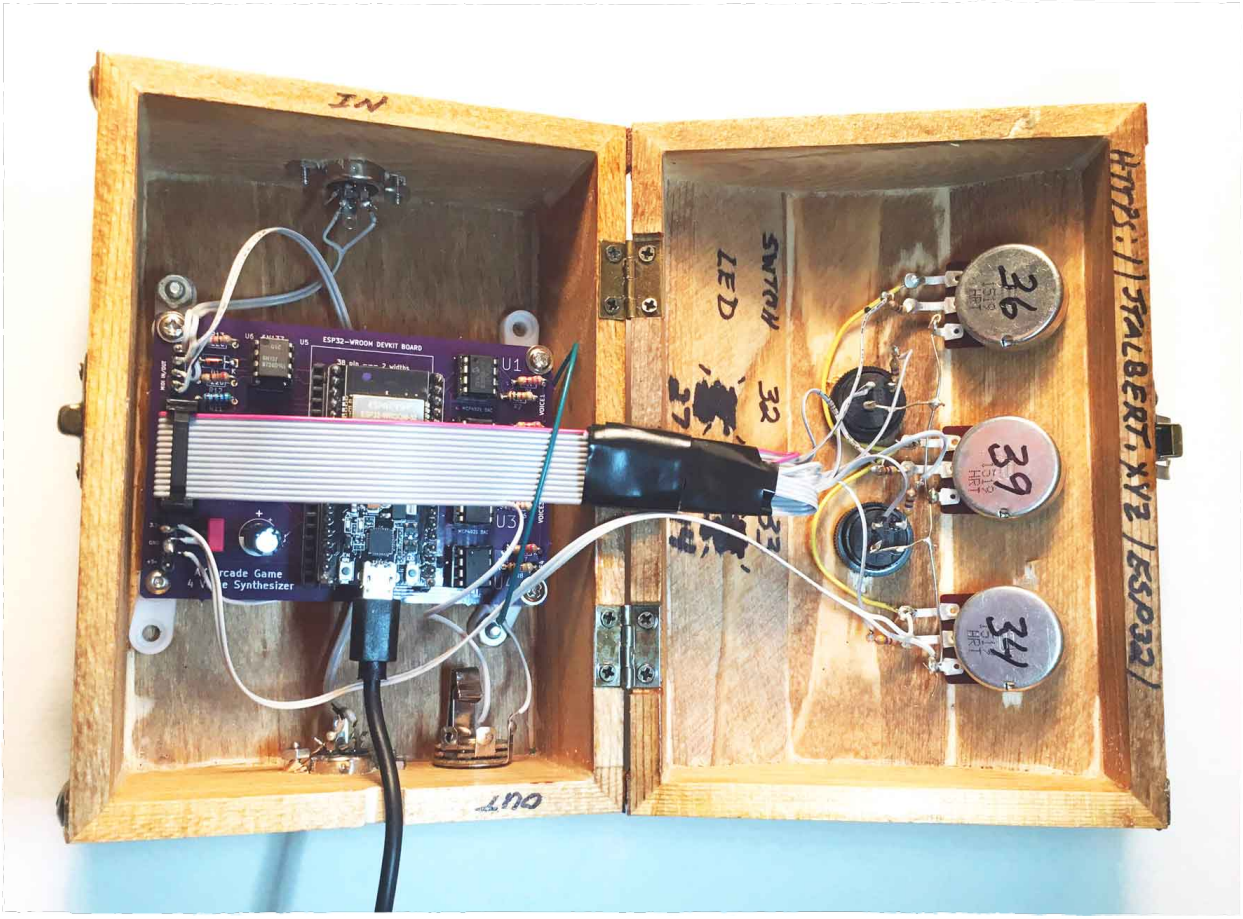
ESP32 Projects

John Talbert - September 2022

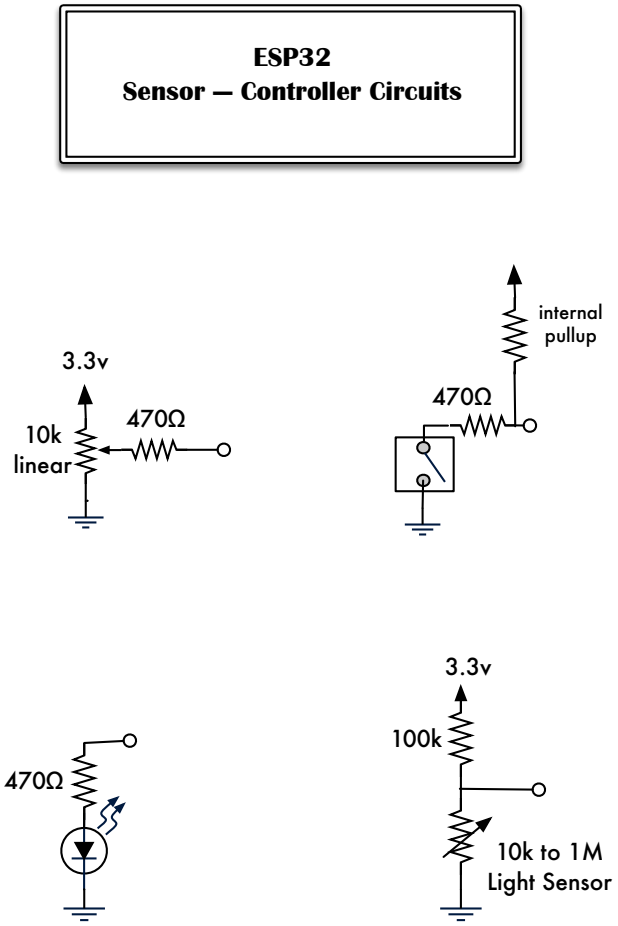
These are enclosures built for the two ESP32 microprocessor circuit boards described in detail on the jtalbert.xyz/esp32/ website, along with this entry — the ESP32 4-Voice Synthesizer and the ESP32 Audio Codec.

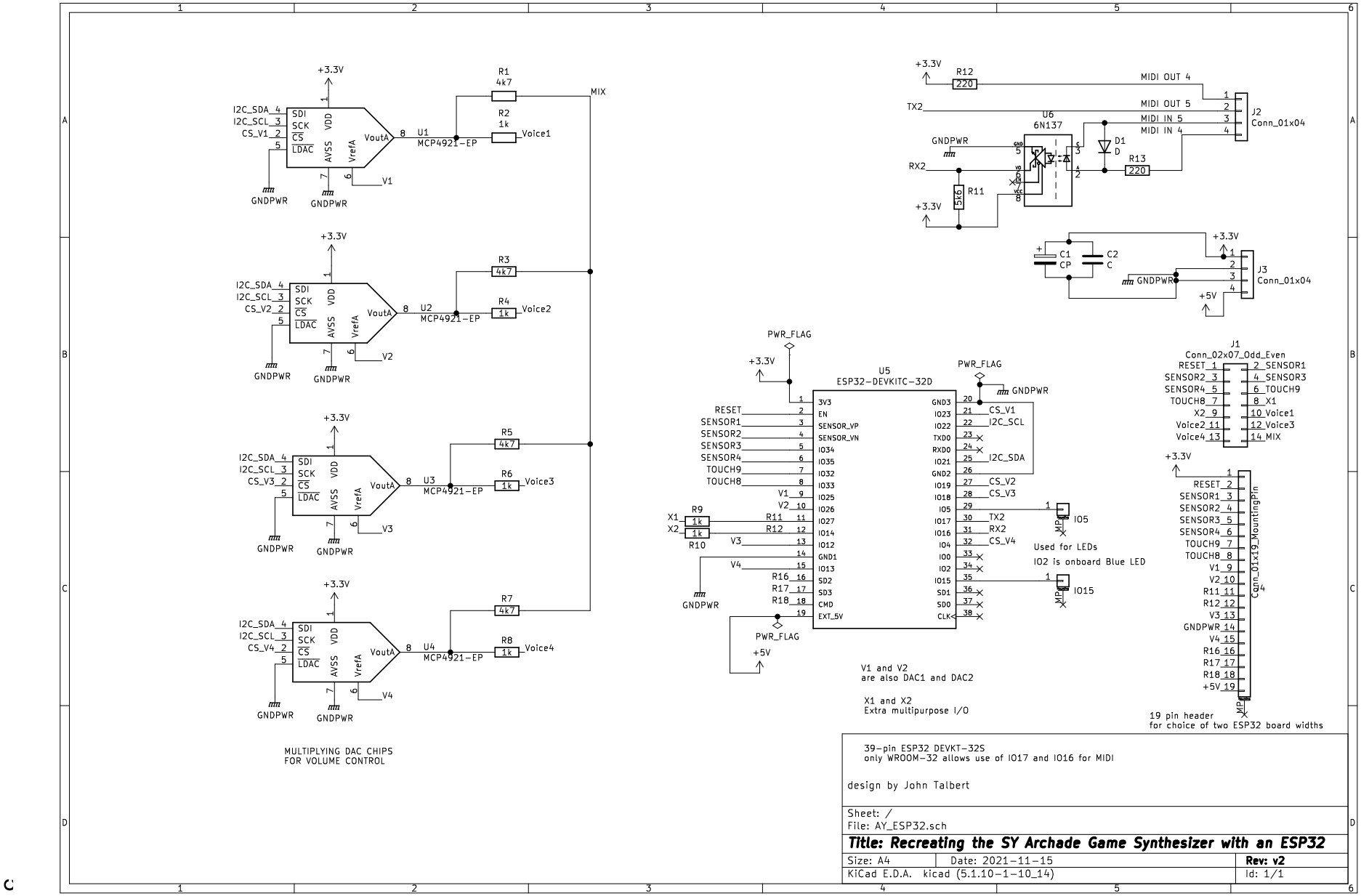


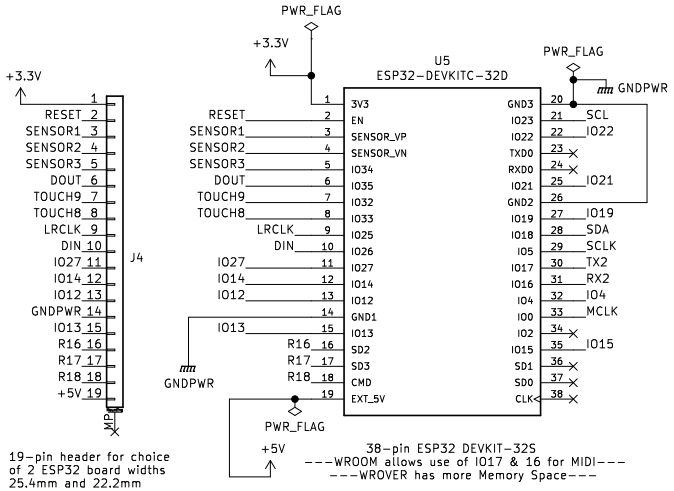
Each enclosure has 3 controller potentiometers, two pushbutton switches, two LED lights, an Audio Output Jack (the Codec also has an Audio Input Jack), and MIDI Input and Output 5-pin DIN Jacks.

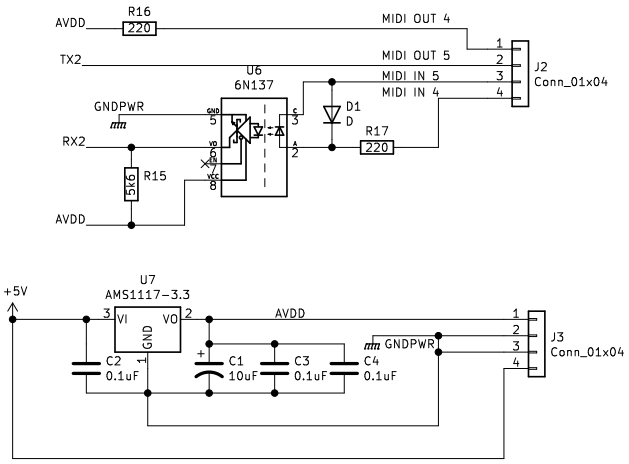
The circuits are both powered by and programmed through a micro USB cable attached to the ESP32 Dev Kit (WROOM) microprocessor.

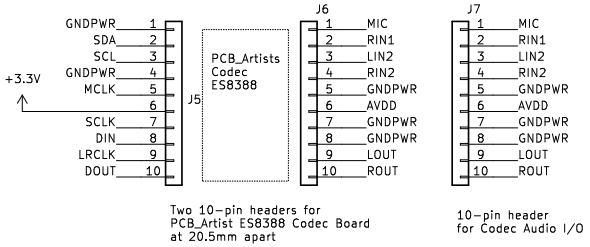
The circuits are shown here. Note the 470 Ohm resistor used in the Switch and Potentiometer circuits. This small resistor protects the micro pin from short circuiting to Ground in the event the pin is mistakenly programmed as an output instead of an input.

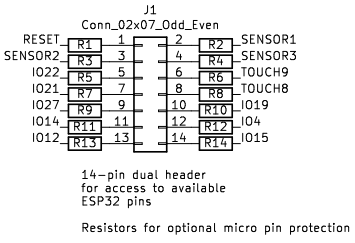



+3.3V
R1
4k7
MIX
I2C_SDA
I2C_SCL
CS_V1
SDI
SCK
CS
LDAC
VDD
AVSS
VrefA
VoutA
U1
MCP4921-EP
R2
1k
Voice1
V1
GNDPWR
R3
4k7
CS_V2
U2
R4
Voice2
V2
CS_V3
U3
R5
R6
Voice3
V3
CS_V4
U4
R7
R8
Voice4
V4
MULTIPLYING DAC CHIPS
FOR VOLUME CONTROL
R12
220
MIDI OUT 4
TX2
MIDI OUT 5
MIDI IN 5
MIDI IN 4
J2
Conn_01x04
U6
6N137
D1
D
R13
220
RX2
R11
5k6
C1
CP
C2
C
J3
Conn_01x04
+5V
PWR_FLAG
U5
ESP32-DEVKITC-32D
3V3
EN
SENSOR_VP
SENSOR_VN
IO34
IO35
IO32
IO33
IO25
IO26
IO27
IO14
IO12
GND1
IO13
SD2
SD3
CMD
EXT_5V
GND3
IO23
IO22
TXD0
RXD0
IO21
GND2
IO19
IO18
IO5
IO17
IO16
IO4
IO0
IO2
IO15
SD1
SD0
CLK
RESET
SENSOR1
SENSOR2
SENSOR3
SENSOR4
TOUCH9
TOUCH8
R9
R10
X1
X2
Used for LEDs
IO2 is onboard Blue LED
V1 and V2
are also DAC1 and DAC2
X1 and X2
Extra multipurpose I/O
J1
Conn_02x07_Odd_Even
SENSOR1
SENSOR3
TOUCH9
X1
Voice1
Voice3
MIX
J4
Conn_01x19_MountingPin
19 pin header
for choice of two ESP32 board widths
39-pin ESP32 DEVKT-32S
only WROOM-32 allows use of IO17 and IO16 for MIDI
design by John Talbert
Sheet: /
File: AY_ESP32.sch
Title: Recreating the SY Archade Game Synthesizer with an ESP32
Size: A4
Date: 2021-11-15
Rev: v2
KiCad E.D.A. kicad (5.1.10-1-10_14)
Id: 1/1









design by John Talbert

Sheet: /
File: ESP32_Codec.sch

Title: PCB_Artist Codec connected to ESP32 Dev Kit w/MIDI

Size: A4	Date: 2022-01-07	**Rev: v1**
KiCad E.D.A. kicad (5.1.10-1-10_14)		Id: 1/1

The Controllers mounted on the enclosures are connected by ribbon cables to the following ESP32 pins.

IO36	Pot 1
IO39	Pot 2
IO34	Pot 3
IO32	Pushbutton 1
IO33	Pushbutton 2
IO27	LED 1 (integrated in Pushbutton 1)
IO14	LED 2 (integrated in Pushbutton 2)
IO17	Serial2 transmit for MIDI Output
IO16	Serial2 receive for MIDI Input

The AY 4-Voice Synth board has been slightly modified from the one described in detail. IO25 and IO26 are connected to the first two Multiplying DAC chips to take advantage of their special 8-bit DAC output capabilities. IO27 and IO14 are then available for other functions as X1 and X2. (Here they are used for LEDs.)

The Codec enclosure's input and output jacks are connected to single header pins that can be moved to any of the codec audio inputs and outputs available on a 10-pin codec header (refer to the codec circuit diagram).

What follows are a few simple Arduino sketches to get you started. More involved sketches can be found in the other sections on the jtalbert.xyz/esp32/ website page.

Be sure to install the "ESP32 Arduino" Library onto your Arduino programming app from the Tools/Manage Libraries… menu. This is a library created by Espressif, the company that created the ESP32 micro. (https://github.com/espressif/arduino-esp32).

Select "ESP32 Dev Module" for the Board in the Arduino programming IDE app.

The **first sketch** sets up all the Controller devices and prints out running values from the Arduino IDE Monitor. Use this sketch as a template to start all your own sketches with setting up the controllers.

The **second sketch** is a test for both MIDI Input and Output. Connect the box to any MIDI keyboard synthesizer using standard MIDI cables to connect MIDI Out from the box to MIDI In of the Keyboard and MIDI In from the box to MIDI Out of the Keyboard Synth. Set the MIDI keyboard to send and receive on MIDI Channel 1.

The **third sketch** works with only the 4-Voice Synth board. It demonstrates how to use the 4 multiplying DAC chips as volume controls for the 4 voices. It also demonstrates how to use the special 8-bit DAC function on IO pins 25 and 26.

A simple sketch for audio effects with codec board was not included here. A codec chip requires a rather involved initialization routine before it can be used. The ESP32 Codec section on the website demonstrates several special libraries that can run this particular codec.

NOTE: Loading Programs from the Arduino IDE may require you to press the LOAD pushbutton on the ESP32 DEV Module. There is a RESET button and a LOAD button on the Micro board. The LOAD button is next to the pin labeled CLK.

Initiate the LOAD from the Arduino IDE. Watch the program go through its Compile and wait for it to attempt to Upload the code onto the ESP32. If it gets stuck here press the LOAD button and hold it until you see the Upload starting.

```
/*
 * Use Tools/SerialMonitor to print all pot and switch values and watch them change

 * ESP32 ADCs are 12-bit (0 to 4095)

 */

// ~~~~~~~~~~~~~~~~~~  CONSTANTS/VARIALBES  ~~~~~~~~~~~~~~~~

const byte LED1 = 27;
const byte LED2 = 14;

const byte POT1 = 36;
const byte POT2 = 39;
const byte POT3 = 34;

const byte KEY1 = 32;
const byte KEY2 = 33;

short pot1;
short pot2;
short pot3;

bool key1;
bool key2;

int x=0;
```

```
// ~~~~~~~~~~~~~~~~~~~  FUNCTIONS  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loadSensors(){      // load all current sensor values
      pot1 =   analogRead(POT1) ;
      pot2 =   analogRead(POT2) ;
      pot3 =   analogRead(POT3) ;

      key1 = digitalRead(KEY1);
      key2 = digitalRead(KEY2);
}

int switchCombo(){
    int result = !key2 + (!key1 * 2);

    switch (result) {
      case 0:
            digitalWrite(LED2, LOW);
            digitalWrite(LED1, LOW);
            break;
      case 1:
            digitalWrite(LED2, HIGH);
            digitalWrite(LED1, LOW);
            break;
      case 2:
            digitalWrite(LED2, LOW);
            digitalWrite(LED1, HIGH);
            break;
      case 3:
            digitalWrite(LED2, HIGH);
            digitalWrite(LED1, HIGH);
            break;
    }
    return result;
}


// ~~~~~~~~~~~~~~~~~~~  SETUP  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void setup() {

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);

  pinMode(KEY1, INPUT_PULLUP);  //engage internal pullup resistor
  pinMode(KEY2, INPUT_PULLUP);
  Serial.begin(115200);  //Set the Arduino Monitor to this value
}
```

```
// ~~~~~~~~~~~~~~~~~~~  LOOP  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void loop() {

  loadSensors();
  x = switchCombo();

   Serial.print("pot1 = ");
  Serial.print(pot1);
    Serial.print("  pot2 = ");
  Serial.print(pot2);
    Serial.print("   pot3 = ");
  Serial.print(pot3);

 Serial.print("      ");

   Serial.print("  switches ");
  Serial.print(key1);
  Serial.print(key2);
  Serial.print("    ");
  Serial.println(x);

     delay(500);           // wait for a half second
}
```

```
/*
              MIDI INPUT and OUTPUT tested with a MIDI input CallBack Function

              On each NOTE On message received, 2 extra arpeggiated notes will be played.
              Pot 1 sets the playback speed.
              Pot 2 sets the note spread

              Switch 1 plays random notes at speed set by Pot 3.
*/
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//                  CONSTANTS and Variables
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//
//On an ESP32 WROOM MIDI can be set up on Serial2 --> RX2 and TX2 on pins 16 and 17.

#include <MIDI.h>  // version 4.x.x

MIDI_CREATE_INSTANCE(HardwareSerial, Serial2, MIDI);


// ~~~~~~~~~~~~~~~~~~  CONSTANTS/VARIALBES  ~~~~~~~~~~~~~~~~
//                  GIOP Pin Assignments

const uint8_t POT1 = 36;
const uint8_t POT2 = 39;
const uint8_t POT3 = 34;

short pot1;
short pot2;
short pot3;

const uint8_t SWITCH1 = 32;
const uint8_t SWITCH2 = 33;

bool switch1;
bool switch2;

const uint8_t MIDI_TX2 = 17;   //MIDI I/O (can also serve as LED on MIDI Out)
const uint8_t MIDI_RX2 = 16;   //serial function only on WROOM, won't work on WROVER ESP32s

const byte LED1 = 27;
const byte LED2 = 14;

const uint8_t LED3 = 2;  //Blue LED on 32S boards
```

```
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//                    Callback MIDI_In Test function
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// user created Callback Function for MIDI Input Test

 void myHandleNoteOn(byte channel, byte note, byte velocity){

  //Serial.println(" Saw a NoteOn ");

  int x = (pot1 >> 4) + 20;     //pot1 sets arpeggio speed

  int y = pot2;                 //pot2 sets arpeggio pitch range
  y = map(y, 0, 4095, 0, 20);

  delay(x);
  MIDI.sendNoteOn(note + y, velocity, 1);
  delay(x);
  MIDI.sendNoteOn(note + y + y, velocity, 1);
  delay(x);

  MIDI.sendNoteOn(note, 0, 1);
  MIDI.sendNoteOn(note + y, 0, 1);
  MIDI.sendNoteOn(note + y + y, 0, 1);
 }

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//                    SETUP()
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


void setup() {

delay(1000);

  MIDI.setHandleNoteOn(myHandleNoteOn); //for Callback MIDI In Test
  MIDI.begin(MIDI_CHANNEL_OMNI);

  // initialize Switches with pullup resistor
  pinMode(SWITCH1, INPUT_PULLUP);
  pinMode(SWITCH2, INPUT_PULLUP);

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
```

```
  digitalWrite(LED3, HIGH);  //flash board's blue LED
  delay(500);
  digitalWrite(LED3, LOW);
  delay(500);
  digitalWrite(LED3, HIGH);
  delay(500);
  digitalWrite(LED3, LOW);

  Serial.begin(115200);


} //End of Setup

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//                    MAIN LOOP
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loop() {

loadSensors();

// On MIDI.read() MIDI class will call Callback functions.
// User created callback function myHandleNoteOn() in section before setup()
// and MIDI.setHandleNoteOn(myHandleNoteOn) in setup() section

MIDI.read();

//Serial.println(!switch1);
if(!switch1){    // test MIDI Output with Switch 1 and Pot 3
  int note = random(30, 90);
  MIDI.sendNoteOn(note, 64, 1);
  delay((pot3 >> 4) + 10);
  MIDI.sendNoteOn(note, 0, 1);
}

} //End of Main Loop


// ~~~~~~~~~~~~~~~~~~~~  FUNCTIONS  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loadSensors(){      // load all current sensor values
      pot1 =   analogRead(POT1) ;
      pot2 =   analogRead(POT2) ;
      pot3 =   analogRead(POT3) ;

      switch1 = digitalRead(SWITCH1);
      switch2 = digitalRead(SWITCH2);
}

}
```

```
/*
  ESP32 AY_Synth Basic Use Setup
*/

//already declared in ESP32 library
//static const uint8_t SCL = 22;   //Serial Lines to MCP4921 Multiplier DACs
//static const uint8_t SDA = 21;

#include "MCP_DAC.h"  //MCP_DAC Library by Rob Tillaart
MCP4921 myAYDAC1(SDA, SCL);
MCP4921 myAYDAC2(SDA, SCL);
MCP4921 myAYDAC3(SDA, SCL);
MCP4921 myAYDAC4(SDA, SCL);


// ~~~~~~~~~~~~~~~~~~  CONSTANTS/VARIALBES  ~~~~~~~~~~~~~~~~
//                  GIOP Pin Assignments

const uint8_t POT1 = 36;
const uint8_t POT2 = 39;
const uint8_t POT3 = 34;

short pot1;
short pot2;
short pot3;

const uint8_t SWITCH1 = 32;
const uint8_t SWITCH2 = 33;

bool switch1;
bool switch2;

//already declared in ESP32 library
//const uint8_t DAC1 = 25;
//const uint8_t DAC2 = 26;

const uint8_t V1 = 25;  //Voice Pin inputs to 4 MCP4921 Multiplier DACs
const uint8_t V2 = 26;  //Note that V1 and V2 are also DAC1 and DAC2
const uint8_t V3 = 12;
const uint8_t V4 = 13;

const uint8_t CS_V1 = 23;   //Chip Selects to 4 MCP4921 Multiplier DACs
const uint8_t CS_V2 = 19;
const uint8_t CS_V3 = 18;
const uint8_t CS_V4 = 4;

const uint8_t MIDI_TX2 = 17;   //MIDI I/O
const uint8_t MIDI_RX2 = 16;   //functions only on WROOM, won't work on WROVER ESP32s
```

```
const uint8_t LED1 = 27;
const uint8_t LED2 = 14;
const uint8_t LED3 = 2;  //Blue LED on 32S boards

uint8_t Triangle[360];
uint8_t Sine[360];
uint8_t Square[360];
bool highlow = 0;
uint8_t increment = 0;

// ~~~~~~~~~~~~~~~~~~~  SETUP  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void setup() {

  myAYDAC1.begin(CS_V1);
  myAYDAC2.begin(CS_V2);
  myAYDAC3.begin(CS_V3);
  myAYDAC4.begin(CS_V4);

  // initialize Switches with pullup resistor
  pinMode(SWITCH1, INPUT_PULLUP);
  pinMode(SWITCH2, INPUT_PULLUP);

 //initialize DAC chip selects, LOW select, unselect all
  pinMode(CS_V1, OUTPUT);
  digitalWrite(CS_V1, HIGH);
  pinMode(CS_V2, OUTPUT);
  digitalWrite(CS_V2, HIGH);
  pinMode(CS_V3, OUTPUT);
  digitalWrite(CS_V3, HIGH);
  pinMode(CS_V4, OUTPUT);
  digitalWrite(CS_V4, HIGH);

 //initialize 4 digital voice inputs to Multiplier DACs
  pinMode(V1, OUTPUT);
  digitalWrite(V1, HIGH);
  pinMode(V2, OUTPUT);
  digitalWrite(V2, HIGH);
  pinMode(V3, OUTPUT);
  digitalWrite(V3, HIGH);
  pinMode(V4, OUTPUT);
  digitalWrite(V4, HIGH);

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
```

```
  digitalWrite(LED3, HIGH);  //flash board's blue LED
  delay(500);
  digitalWrite(LED3, LOW);
  delay(500);
  digitalWrite(LED3, HIGH);
  delay(500);
  digitalWrite(LED3, LOW);

  for(int deg = 0; deg < 360; deg++){     //calculate and fill waveform arrays
    Sine[deg] = int(128 + 80 * (sin(deg*PI/180)));
    Square[deg] = int(128 + 80 * (sin(deg*PI/180)+sin(3*deg*PI/180)/3+sin(5*deg*PI/180)/5+sin(7*deg*PI/
180)/7+sin(9*deg*PI/180)/9+sin(11*deg*PI/180)/11));
    Triangle[deg] = int(128 + 80 * (sin(deg*PI/180)+1/pow(3,2)*sin(3*deg*PI/180)+1/pow(5,2)*sin(5*deg*PI/
180)+1/pow(7,2)*sin(7*deg*PI/180)+1/pow(9,2)*sin(9*deg*PI/180)));
  }

  //Serial.begin(115200);

}

// ~~~~~~~~~~~~~~~~~~~  LOOP  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loop() {

   loadSensors();

//pot1 pans the voice volume between two voices
//switch1 selects one of two voice pairs to output
//pot2 sets frequency by changing number of samples drawn from waveform array
//pot3 sets frequency by changing a program delay value

if(switch1){
  myAYDAC1.analogWrite(pot1);
  myAYDAC2.analogWrite(4095 - pot1);
  myAYDAC3.analogWrite(0);
  myAYDAC4.analogWrite(0);
}
else {
  myAYDAC1.analogWrite(0);
  myAYDAC2.analogWrite(0);
  myAYDAC3.analogWrite(pot1);
  myAYDAC4.analogWrite(4095 - pot1);
}
```

```
  //create a tone in all 4 voices

  increment = 1 + (pot2 >> 5);  //sets the number of waveform samples loaded out of 360

for (int deg = 0; deg < 360; deg = deg + increment ){
  dacWrite(DAC1, Sine[deg]);
  //dacWrite(DAC2, Square[deg]);    //alternative waveforms for the two voice DAC pins
  //dacWrite(DAC1, Triangle[deg]);
  dacWrite(DAC2, random(deg));

  delayMicroseconds(pot3 >> 6);
}
  digitalWrite(V3, highlow);       //These two voice pins are digital outputs only
  highlow = !highlow;
  digitalWrite(V4, random(2));

}


// ~~~~~~~~~~~~~~~~~~~  FUNCTIONS  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void loadSensors(){      // load all current sensor values
      pot1 =   analogRead(POT1) ;
      pot2 =   analogRead(POT2) ;
      pot3 =   analogRead(POT3) ;

      switch1 = digitalRead(SWITCH1);
      switch2 = digitalRead(SWITCH2);
}
```