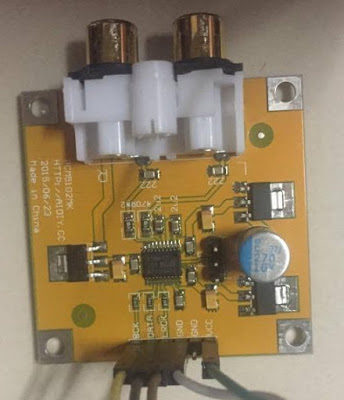
# **[Demo 26: How to use Arduino ESP32 I2S to play wav music file from sdcard](http://www.iotsharing.com/2017/07/how-to-use-arduino-esp32-i2s-to-play-wav-music-from-sdcard.html)**

1. **Introduction**  
   - ESP32 has two I2S peripherals. They can be configured to input and output sample data. They also supports [DMA](https://en.wikipedia.org/wiki/Direct_memory_access" \t "http://www.iotsharing.com/2017/07/_blank) to stream sample data without needing CPU operations. I2S output can also be routed directly to the Digital to Analog Converter output (GPIO25 and GPIO26) without needing [external I2S codec](https://vi.aliexpress.com/item/PCM5102-DAC-Decoder-I2S-Player-Assembled-Board-32Bit-384K-Beyond-ES9023-PCM1794-compatible-Raspberry-Pi/32678406932.html?spm=2114.55010608.4.2.jU683C" \t "http://www.iotsharing.com/2017/07/_blank).  
   - In this demo i will show you how to use Arduino ESP32 I2S to play **wav** music file from sdcard. I chose wav file because it is not compressed like mp3 file. So we need not to de-compress it.  
   - I used [external I2S codec](https://vi.aliexpress.com/item/PCM5102-DAC-Decoder-I2S-Player-Assembled-Board-32Bit-384K-Beyond-ES9023-PCM1794-compatible-Raspberry-Pi/32678406932.html?spm=2114.55010608.4.2.jU683C" \t "http://www.iotsharing.com/2017/07/_blank) for this demo, 2 speakers and 1 module [micro sdcard](http://www.iotsharing.com/2017/05/how-to-use-arduino-esp32-to-store-data-to-sdcard.html" \t "http://www.iotsharing.com/2017/07/_blank).

[](https://2.bp.blogspot.com/-0HAUwtfwAN0/WWIGfUgvaOI/AAAAAAAAEHU/a2L05ocwOsImQJHCxGt2H8q-ZDahPMfhgCLcBGAs/s1600/esp32_i2s_2.jpg)

**Figure: [external I2S codec](https://vi.aliexpress.com/item/PCM5102-DAC-Decoder-I2S-Player-Assembled-Board-32Bit-384K-Beyond-ES9023-PCM1794-compatible-Raspberry-Pi/32678406932.html?spm=2114.55010608.4.2.jU683C" \t "http://www.iotsharing.com/2017/07/_blank)**

[](https://3.bp.blogspot.com/-zrYW161XJYY/WWIGgYKIxBI/AAAAAAAAEHY/_J9i7tJxQkwt6MBxSWFK-T8E_onIHoOYwCLcBGAs/s1600/esp32_i2s_1.jpg)

**Figure: I used [external I2S codec](https://vi.aliexpress.com/item/PCM5102-DAC-Decoder-I2S-Player-Assembled-Board-32Bit-384K-Beyond-ES9023-PCM1794-compatible-Raspberry-Pi/32678406932.html?spm=2114.55010608.4.2.jU683C" \t "http://www.iotsharing.com/2017/07/_blank) for this demo, 2 speakers and 1 module micro sdcard**

**2. Hardware**

Connect hardware like below:

[ESP32 IO32 – CS MICROSD]  
[ESP32 IO14 – MOSI MICROSD]  
[ESP32 IO12 – MISO MICROSD]  
[ESP32 IO27 – SCK MICROSD]  
[ESP32 IO26 – I2S codec BCK]  
[ESP32 IO22 – I2S codec DATA]  
[ESP32 IO25 – I2S codec LRCK]  
[ESP32 GND – I2S codec GND]  
[ESP32 GND – GND MICROSD]  
[5V – VCC MICROSD]  
[5V – I2S codec]  
**3. Software**  
- We will re-use [Demo 7](http://www.iotsharing.com/2017/05/how-to-use-arduino-esp32-to-store-data-to-sdcard.html" \t "http://www.iotsharing.com/2017/07/_blank) for sdcard reading and I2S driver [here](http://esp-idf.readthedocs.io/en/latest/api-reference/peripherals/i2s.html" \t "http://www.iotsharing.com/2017/07/_blank). You can download the document about the wav file format [here](http://tiny.systems/software/soundProgrammer/WavFormatDocs.pdf" \t "http://www.iotsharing.com/2017/07/_blank).  
**Note**: You can down full project including wav file sample : <https://github.com/nhatuan84/esp32-i2s-sdcard-wav-player>

|  |
| --- |
| #include <mySD.h>  #include "driver/i2s.h"  #include "freertos/queue.h"  #define CCCC(c1, c2, c3, c4) ((c4 << 24) | (c3 << 16) | (c2 << 8) | c1)  /\* these are data structures to process wav file \*/  typedef enum headerState\_e {  HEADER\_RIFF, HEADER\_FMT, HEADER\_DATA, DATA  } headerState\_t;  typedef struct wavRiff\_s {  uint32\_t chunkID;  uint32\_t chunkSize;  uint32\_t format;  } wavRiff\_t;  typedef struct wavProperties\_s {  uint32\_t chunkID;  uint32\_t chunkSize;  uint16\_t audioFormat;  uint16\_t numChannels;  uint32\_t sampleRate;  uint32\_t byteRate;  uint16\_t blockAlign;  uint16\_t bitsPerSample;  } wavProperties\_t;  /\* variables hold file, state of process wav file and wav file properties \*/  File root;  headerState\_t state = HEADER\_RIFF;  wavProperties\_t wavProps;  //i2s configuration  int i2s\_num = 0; // i2s port number  i2s\_config\_t i2s\_config = {  .mode = (i2s\_mode\_t)(I2S\_MODE\_MASTER | I2S\_MODE\_TX),  .sample\_rate = 36000,  .bits\_per\_sample = I2S\_BITS\_PER\_SAMPLE\_16BIT,  .channel\_format = I2S\_CHANNEL\_FMT\_RIGHT\_LEFT,  .communication\_format = (i2s\_comm\_format\_t)(I2S\_COMM\_FORMAT\_I2S | I2S\_COMM\_FORMAT\_I2S\_MSB),  .intr\_alloc\_flags = ESP\_INTR\_FLAG\_LEVEL1, // high interrupt priority  .dma\_buf\_count = 8,  .dma\_buf\_len = 64 //Interrupt level 1  };    i2s\_pin\_config\_t pin\_config = {  .bck\_io\_num = 26, //this is BCK pin  .ws\_io\_num = 25, // this is LRCK pin  .data\_out\_num = 22, // this is DATA output pin  .data\_in\_num = -1 //Not used  };  //  void debug(uint8\_t \*buf, int len){  for(int i=0;i<len;i++){  Serial.print(buf[i], HEX);  Serial.print("\t");  }  Serial.println();  }  /\* write sample data to I2S \*/  int i2s\_write\_sample\_nb(uint32\_t sample){  return i2s\_write\_bytes((i2s\_port\_t)i2s\_num, (const char \*)&sample, sizeof(uint32\_t), 100);  }  /\* read 4 bytes of data from wav file \*/  int read4bytes(File file, uint32\_t \*chunkId){  int n = file.read((uint8\_t \*)chunkId, sizeof(uint32\_t));  return n;  }  /\* these are function to process wav file \*/  int readRiff(File file, wavRiff\_t \*wavRiff){  int n = file.read((uint8\_t \*)wavRiff, sizeof(wavRiff\_t));  return n;  }  int readProps(File file, wavProperties\_t \*wavProps){  int n = file.read((uint8\_t \*)wavProps, sizeof(wavProperties\_t));  return n;  }  void setup()  {  Serial.begin(115200);  Serial.print("Initializing SD card...");  if (!SD.begin(32, 14, 12, 27)) {  Serial.println("initialization failed!");  return;  }  Serial.println("initialization done.");  delay(1000);  /\* open wav file and process it \*/  root = SD.open("T.WAV");  if (root) {  int c = 0;  int n;  while (root.available()) {  switch(state){  case HEADER\_RIFF:  wavRiff\_t wavRiff;  n = readRiff(root, &wavRiff);  if(n == sizeof(wavRiff\_t)){  if(wavRiff.chunkID == CCCC('R', 'I', 'F', 'F') && wavRiff.format == CCCC('W', 'A', 'V', 'E')){  state = HEADER\_FMT;  Serial.println("HEADER\_RIFF");  }  }  break;  case HEADER\_FMT:  n = readProps(root, &wavProps);  if(n == sizeof(wavProperties\_t)){  state = HEADER\_DATA;  }  break;  case HEADER\_DATA:  uint32\_t chunkId, chunkSize;  n = read4bytes(root, &chunkId);  if(n == 4){  if(chunkId == CCCC('d', 'a', 't', 'a')){  Serial.println("HEADER\_DATA");  }  }  n = read4bytes(root, &chunkSize);  if(n == 4){  Serial.println("prepare data");  state = DATA;  }  //initialize i2s with configurations above  i2s\_driver\_install((i2s\_port\_t)i2s\_num, &i2s\_config, 0, NULL);  i2s\_set\_pin((i2s\_port\_t)i2s\_num, &pin\_config);  //set sample rates of i2s to sample rate of wav file  i2s\_set\_sample\_rates((i2s\_port\_t)i2s\_num, wavProps.sampleRate);  break;  /\* after processing wav file, it is time to process music data \*/  case DATA:  uint32\_t data;  n = read4bytes(root, &data);  i2s\_write\_sample\_nb(data);  break;  }  }  root.close();  } else {  Serial.println("error opening test.txt");  }  i2s\_driver\_uninstall((i2s\_port\_t)i2s\_num); //stop & destroy i2s driver  Serial.println("done!");  }  void loop()  {  } |