FireBeetle Covers-Camera&Audio Media Board SKU:DFR0498

Introduction

DFRobot FireBeetle series are low power consumption development modules designed for Internet of Things (IoT). The FireBeetle Covers-Camera & Audio Media Board is a multimedia device for IoT that provides interfaces to connect NAU8822 CODEC IIS, OV7725 camera, SD card (SDIO), earphone and microphone.

Moreover, it equips with a mini MIC input interface. The NAU8822 CODEC IIS can drive both 12@8Ω BTL loudspeaker and 40mW@16Ω earphone. The direct connection is supported. Meanwhile, NAU8822 supports DAC sound signal acquisition and programmable microphone amplifier. The recording is available when input voices with onboard MIC interface or microphone then save it to SD card. What's more, the function to take photos is available when connected to an OV7725 CAMERA.

With any FireBeetle main boards (e.g. ESP32 main board), the FireBeetle Covers-Camera & Audio Media Board could be a MP3, a recorder, a camera. Once connected to the Internet, it could be an Internet radio, a cloud image recognition.

NOTE

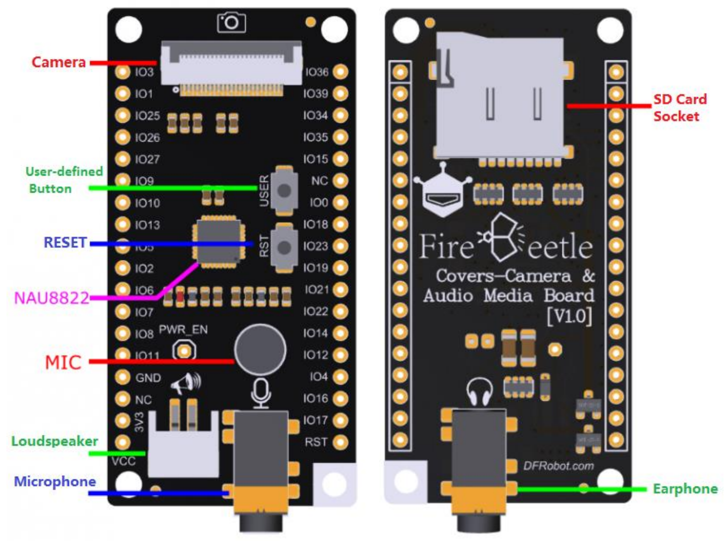
This media board only supports FireBeetle ESP32 main board at present, for other main boards do not equipped with IIS interface.
The flat ribbon cable of camera is relatively easy-to-break, please be careful and light in connection. The camera shot must be opposite to the SD card slot.
Do not support video shooting and transmission.

Specification

- Operating Voltage: 3.7V~5.5V (VCC)
- Output Voltage: 3.3V
- User-defined Button: tested by IO16(DI)
- RESET x1
- SD Card:
- Bus Interface: SDIO Protocol
- Default Transmission Rate: 10MHz
- Max. Transmission Rate: 20MHz
- Camera OV7725: (data as below is just for reference; some functions could not be realized by ESP32 now).
- Photosensitive/Light-sensitive Array: 640*480
- Optical Size: 1/6"
- Angle of View: 25°
- SCCB Standard Interface
- Output Pixel Format: Raw RGB，RGB(RGB4:2:2 ,RGB565/555/444)，YCbCr (4:2:2)
- Image Size: VGA,QVGA and CIF to 40x30
- VarioPixel Subsampling
- Auto Image Control: auto exposure control (AEC), auto gain control (AGC), auto white balance (AWB), auto band filter and auto black level (ABL) calibration.
- Image Quality Control: saturation, tone, gamma, clarity and anti-interference
- ISP: noise reduction and defect calibration
- Lens Shading Correction
- Auto Saturation Adjustment
- Frame Synchronization
- Fixed-focus
- NAU8822:
- DAC:94dB SNR , -84db THD
- ADC:90dB SNR , -80dB THD
- Integrated BTL loudspeaker drive: 1W @ 8Ω
- Integrated earphone drive: 40mW @ 16Ω
- Integrated programmable microphone amplifier
- Typical Sampling Rate: 8KHz, 48KHz, 96KHz, 192KHz
- Standard Video Interface: PCM and I2S
- MIC:
- Type: Electret Capacitor
- Output: analog
- Direction: omnidirectional

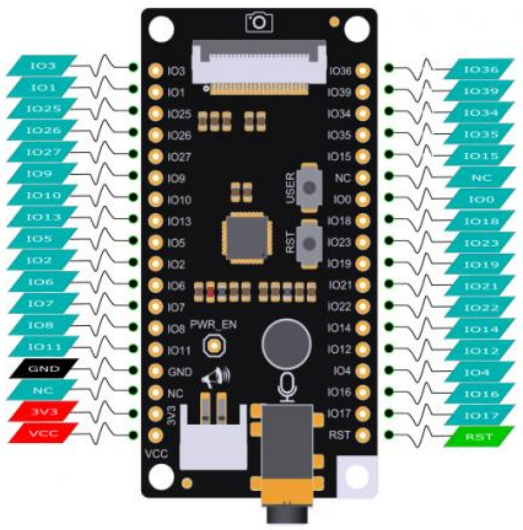
- Frequency Range: 100Hz~15KHz
- Sensitivity: -43dB ±5dB @ 94dB SPL
- Signal-to-noise Ratio: 58Db

Function Diagram



FireBeetle Covers-Camera&Audio Media Board

Board Overview



DFR0498 PinOut

ESP32	SD Card	NAU8822	Camera
IO0		DACIN	RST
IO1			
IO2	DATA0		
IO3			
IO4	DATA1		
IO5		BLCK	D3
IO6			
IO7			
IO8			
IO9			
IO10			
IO11			
IO12	DATA2		
IO13	DATA3		
IO14	CLK		
IO15	CMD		
IO16			
IO17		LRCK	D2
IO18			D4
IO19			D5
IO21			XCLK
IO22		MCLK	PCLK
IO23			HREF
IO25			VSYNC
IO26		SDIO	SDA

IO27		SCLK	SCL
IO34			D8
IO35			D9
IO36			D6
IO39		ADCOUT	D7

Tutorial

Requirements

- Hardware
- ESP32 x1
- FireBeetle Covers-Camera&Audio Media Board x1
- OV7725 Camera x1
- Micro SD Card (FAT32 File System) x1
- Software
- Arduino IDE (Version requirements: V1.8+), click to Download Arduino IDE from Arduino®

Play

Download programs as below to ESP32 main control board and plug a SD Card with test1.wav and test2.wav

Sample Code (Music Play)

Please download and install FireBeetle Covers-Camera&Audio Media Board library files.

How to install Libraries in Arduino IDE

```
/*!

 * @file DFRobot_IIS.h

 * @brief DFRobot's IIS Player Module

 * @n IIS Module for how to begin to play a WAV file,how to excute orders pau
se,continue,stop and play the next

 *    Insert sd card with test1.wav and test2.wav.
 *    Call the function by pressing user key to control music player
 *    The Module would operate as follows when pressed: play test1.wav>>pause
>>continue>>mute>>Volume:50>>stop>>play test2.wav
 */

#include <Wire.h>
#include "DFRobot_IIS.h"

DFRobot_IIS iis;
const int buttonPin = 16;
int i=0;

void setup() {
  Serial.begin(115200);
  iis.SDCardInit();                                  // Init SD card
  iis.init(AUDIO);                                   // Init Audio mode
  iis.setHeadphonesVolume(50);                       // Set Headphones Volume f
rom 0 to 99
  iis.setSpeakersVolume(50);                         // Set Speakers   Volume f
rom 0 to 99
  iis.initPlayer();                                  // Init Music player
  Serial.println("Init player");
  iis.playMusic("/test1.WAV");                       // Choose music file
  Serial.println("Ready to play");
  delay(500);
}

void loop(){
    static unsigned long timepoint = millis();
    static byte count = 0;
    if(millis()-timepoint > 200){
        if((!digitalRead(buttonPin))&&(i==0||i==2)){
            timepoint = millis();
            iis.playerControl(PLAY);               // Start or continue playi
ng Music
            Serial.println("play");
            i++;
        }else if((!digitalRead(buttonPin))&&i==1){
            timepoint = millis();
            iis.playerControl(PAUSE);                   // Pause playing
            Serial.println("pause");
            i++;
        }else if((!digitalRead(buttonPin))&&i==3){
            timepoint = millis();
            iis.muteSpeakers();                         // Mute mode
            iis.muteHeadphones();
            Serial.println("mute mode");
            i++;
        }else if((!digitalRead(buttonPin))&&i==4){
            timepoint = millis();
            iis.setHeadphonesVolume(50);
            iis.setSpeakersVolume(50);
            Serial.println("Volume=50");
            i++;
        }else if((!digitalRead(buttonPin))&&i==5){
            timepoint = millis();
            iis.playerControl(STOP);                    // Stop playing
            Serial.println("Stop");
            i++;
        }else if((!digitalRead(buttonPin))&&i==6){
            timepoint = millis();
            iis.playMusic("/test2.WAV");                // Change music file
            iis.playerControl(PLAY);                    // Play test2.wav
            i=1;
        }
        delay(100);
    }
}
```

- Program Function: The Module would operate as follows when pressed: play test1.wav>>pause>>continue>>mute>>Volume:50>>stop>>play test2.wav

- Functions:
- `SDCardInit():`
- Initialize SD card in the beginning.
- `init(AUDIO):`
- Enter AUDIO mode, state its mode (AUDIO / CAMERA) in use when the SD card is initialized.
- `setHeadphonesVolume(50),setSpeakersVolume(0):`
- Set earphone volume and loudspeaker volume separately. They can be used at the same time and the volume range is 0~99. It should be set before the player initialization and can be used to change volumes in the play.
- `muteSpeakers(),muteHeadphones():`
- Mute: enable loudspeaker and earphone to Mute, available in the play.
- `initPlayer():`
- Initialize the player: to prepare for calling the player function in AUDIO
- `playMusic("/test1.WAV"):`
- Select a music file (test1.wav at here) that stored in the SD card and call to play.
- `playerControl(PLAY),playerControl(PAUSE),playerControl(STOP):`
- Player control: to play, pause and stop the music play in order.

Record

Once the board connected to microphone by MIC, the other onboard microphone is disabled.

Download programs as below to ESP32 main control board and plug a SD Card.

```
/*!
 * @file DFRobot_IIS.h
 * @brief DFRobot's IIS Record Module
 * @n IIS Module for Record sound and Save WAV file in SD card
 *    Insert sd card
 *    Call the function by pressing user key to control recorder
 *    This Module will record sound and save it as record.wav then play recor
d.wav
 */

#include <Wire.h>
#include <DFRobot_IIS.h>

DFRobot_IIS iis;
const int buttonPin = 16;
enum Status{
    ePrepare,
    eRecording,
    eStop
}eState=ePrepare;

void setup(){
    Serial.begin(115200);
    iis.SDCardInit();                                   // SD card init
    iis.init(AUDIO);                                    // Init Audio mode
    iis.initRecorder();                                 // Init recorder
    iis.initPlayer();                                   // Init musice player
    iis.record("/record.WAV");                          // Enter file name to save
recording
    Serial.println("Ready to record");
}

void loop(){
    static unsigned long timepoint = millis();
    static byte count = 0;
    if(millis()-timepoint > 200){
        if((!digitalRead(buttonPin))&&eState==ePrepare){
            timepoint = millis();
            iis.recorderControl(BEGIN);                 // Begin recording
            Serial.println("Recording");
```

```
                eState=eRecording;
            }else if((!digitalRead(buttonPin))&&eState==eRecording){
                timepoint = millis();
                iis.recorderControl(STOP);              // Stop recording
                Serial.println("Stop and save data");
                eState=eStop;
            }else if((!digitalRead(buttonPin))&&eState==eStop){
                timepoint = millis();
                iis.playMusic("/record.WAV");           // Play your record
                iis.setSpeakersVolume(50);
                iis.setHeadphonesVolume(50);
                iis.playerControl(PLAY);
                eState=ePrepare;
            }
            delay(100);
        }
    }
```

Program Function:

Pressing to record when downloaded, repressing the button to stop and save the file: record.wav to SD card and press to play it.

- Functions:
- `SDCardInit():`
- Initialize SD card in the beginning.
- `init(AUDIO):`
- Enter AUDIO mode, state its mode (AUDIO / CAMERA) in use when the SD card is initialized.
- `initRecorder():`
- Initialize the recorder: to prepare for calling the record function in AUDIO.
- `record("/record.WAV"):`
- Name the record file that will be saved to the SD card, call and play.

- `recorderControl(BEGIN),recorderControl(STOP):`

- Recorder control: to play, pause, stop and save in order.

Take Photos

- Due to the limitation of ESP32 processing performance, it only supports 320x240 QVGA resolution for now.
- And the network transmission picture only supports GRAYSCALE photo, the next version will supports colorful image.
- Fixed focus, no need to adjust.

Download programs as below to ESP32 main control board and plug a SD Card

```
/*!
 * @file DFRobot_IIS.h
 * @brief DFRobot's IIS Camera Module
 * @n IIS Module for take photo and save BMP file in SD card
 *    Insert sd card
 *    Call the function by pressing user key to take photo
 *    This Module will take photo and save as photo1.bmp,photo2.bmp by pressi
ng user key
 */

#include <Wire.h>
#include "DFRobot_IIS.h"

DFRobot_IIS iis;
const int buttonPin = 16;
int i=0;
const char* SSID    ="yourSSID";
const char* PASSWORD="SSID password";
void setup(){
    Serial.begin(115200);
    pinMode(buttonPin, INPUT);
    iis.SDCardInit();                                //SD card init
    iis.init(CAMERA);                                  //Init Camera mode
    iis.connectNet(SSID,PASSWORD);                     //Connect wifi
    iis.setFramesize(QVGA);                            //Set photo size QVGA:32
0x240
    iis.setPixformat(GRAYSCALE);                       //Set pixelformat GRAYSC
ALE
    iis.sendPhoto();                                   //Send photo to net
    delay(100);
    Serial.println("Ready to take photo");
}

void loop(){
    static unsigned long timepoint = millis();
    static byte count = 0;
    if(millis()-timepoint > 20){
       timepoint = millis();
       if(!digitalRead(buttonPin)&& i == 0){
          Serial.println("take photo1");
          iis.snapshot("/photo1.bmp");                 //Take photo and save it
as photo1.bmp in SD card
          Serial.println("done");
          i=1;
        }
        if(!digitalRead(buttonPin)&& i == 1){
          Serial.println("take photo2");
          iis.snapshot("/photo2.bmp");                 //Take photo and save it
as photo1.bmp in SD card
          Serial.println("done");
          i=0;
        }
   }
}
```

- Program Function: Press the button to take a photo, saving as photo1.bmp. Press the button again to take another photo, saving as photo2.bmp. Please pay attention that you can keep pressing button to take photo, but it will cover photo1 and photo2 in turn.
- Functions:

- `SDCardInit():`

- Initialize SD card in the beginning.

- `init(CAMERA):`

- Enter CAMERA mode, state its mode (AUDIO / CAMERA) in use when the SD card is initialized.

- `connectNet(SSID,PASSWORD):`

- Connect to Wi-Fi, SSID and PASSWORD are the name and the password of Wi-Fi been used.

- `setFramesize(QVGA):`

- Set image frame size and it need to be called in CAMERA mode, supporting QQVGA (160x120) QQVGA2 (128x160) QICF (176x144) HQVGA (240x160) QVGA (320x240).

- `iis.setPixformat(GRAYSCALE):`

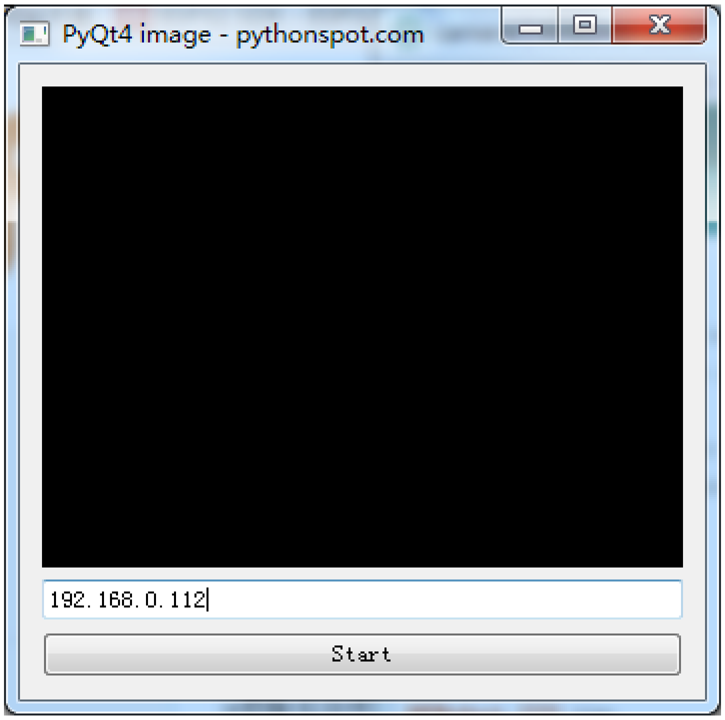
- Set pixel format and it need to be called in CAMERA mode, supporting RGB555 (colorful), GRAYSCALE.

- `sendPhoto():`

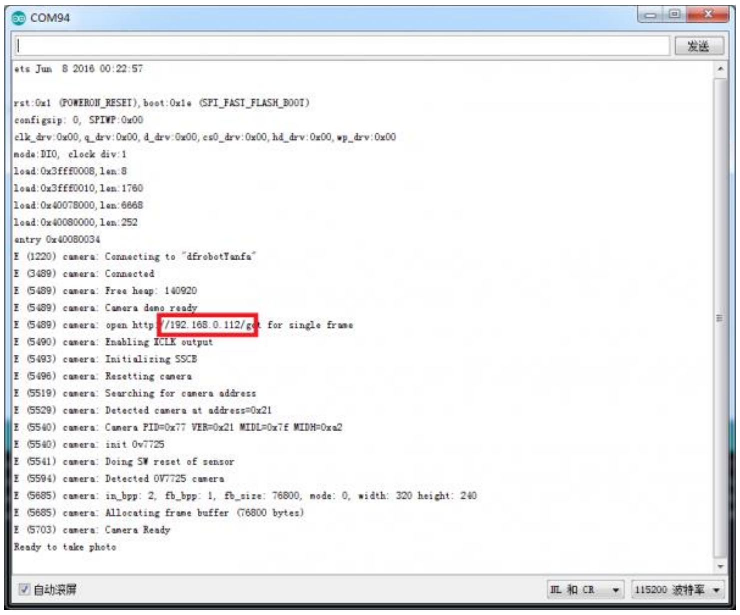
- Connect to Wi-Fi and transmit CAMERA data to the Internet, the CAMERA shooting (GRAYSCALE) can be checked by referring the tutorial and image as below.

- `snapshot("/photo1.bmp"):`

- Function to take a photo: Name the image file (photo1.bmp in SD card at here) to be saved, call the function to take a photo and save.
- Tutorial to get a photo:
- Click to download the software tool in need.
- Open the software tool.

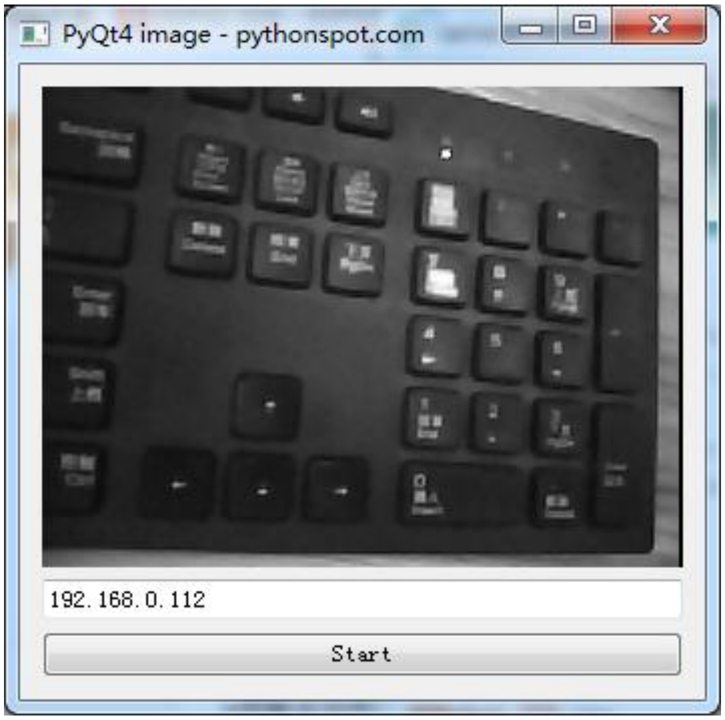


- Run the program and open the serial port to check IP address.



```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x1e (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0008,len:8
load:0x3fff0010,len:1760
load:0x40078000,len:6668
load:0x40080000,len:252
entry 0x40080034
E (1220) camera: Connecting to "dfrobotYanfa"
E (3489) camera: Connected
E (5489) camera: Free heap: 140920
E (5489) camera: Camera demo ready
E (5489) camera: open http://192.168.0.112/get for single frame
E (5490) camera: Enabling XCLK output
E (5493) camera: Initializing SSCB
E (5496) camera: Resetting camera
E (5519) camera: Searching for camera address
E (5529) camera: Detected camera at address=0x21
E (5540) camera: Camera PID=0x77 VER=0x21 MIDL=0x7f MIDH=0xa2
E (5540) camera: init 0v7725
E (5541) camera: Doing SW reset of sensor
E (5594) camera: Detected OV7725 camera
E (5685) camera: in_bpp: 2, fb_bpp: 1, fb_size: 76800, mode: 0, width: 320 height: 240
E (5685) camera: Allocating frame buffer (76800 bytes)
E (5703) camera: Camera Ready
Ready to take photo
```

- Input IP address. (PC and ESP32 main board should be in a same network segment), click start to take a photo in a real time.



SD Card Read & Write

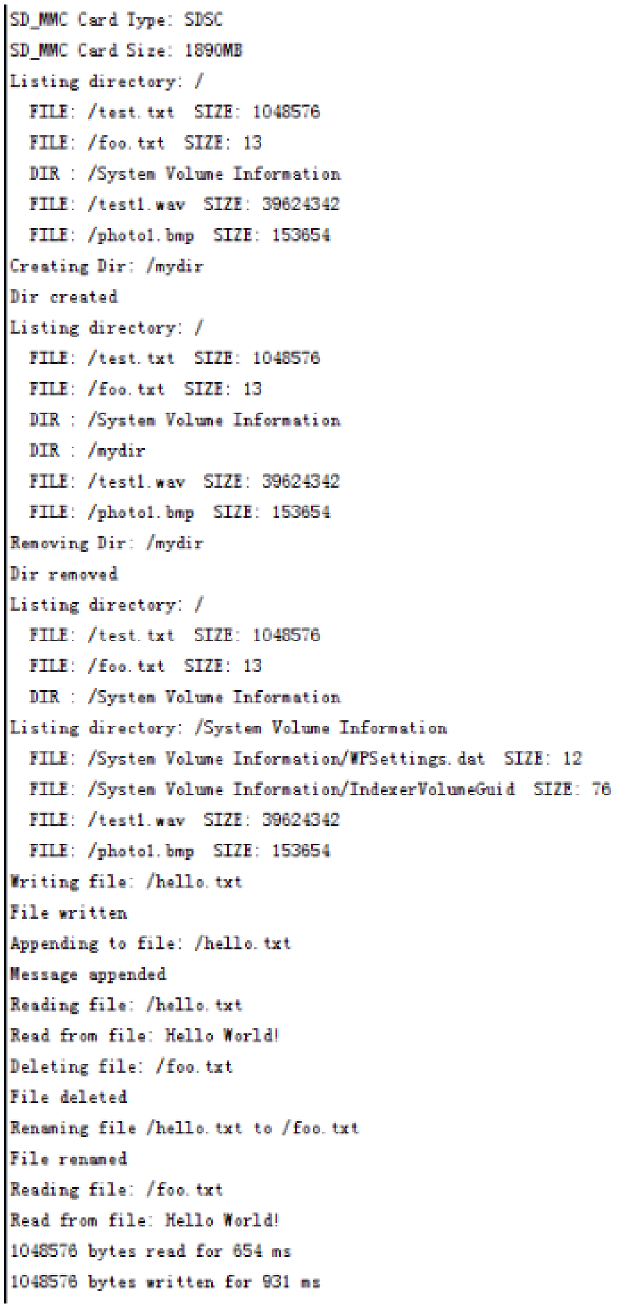
- ESP32 SDMMC library of official Arduino can be directly used in SD card socket of FireBeetle Covers-Camera&Audio Media Board.
- Plug SD card and download the below program to ESP32 mainboard.

```
#include "FS.h"
#include "SD_MMC.h"

/*Print file list*/
void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listing directory: %s\n", dirname);

    File root = fs.open(dirname);
    if(!root){
        Serial.println("Failed to open directory");
        return;
    }
    if(!root.isDirectory()){
        Serial.println("Not a directory");
        return;
    }

    File file = root.openNextFile();
    while(file){
        if(file.isDirectory()){
            Serial.print("  DIR : ");
            Serial.println(file.name());
            if(levels){
                listDir(fs, file.name(), levels -1);
            }
        } else {
            Serial.print("  FILE: ");
            Serial.print(file.name());
            Serial.print("  SIZE: ");
            Serial.println(file.size());
        }
        file = root.openNextFile();
    }
}

/*Create content*/
void createDir(fs::FS &fs, const char * path){
    Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
        Serial.println("Dir created");
    } else {
        Serial.println("mkdir failed");
    }
}

/*Delete content*/
```

```
void removeDir(fs::FS &fs, const char * path){
    Serial.printf("Removing Dir: %s\n", path);
    if(fs.rmdir(path)){
        Serial.println("Dir removed");
    } else {
        Serial.println("rmdir failed");
    }
}

/*Read file*/
void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);

    File file = fs.open(path);
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }

    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
}

/*Write file*/
void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
}

/*Append Writefile*/
void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
}

/*Rename file*/
void renameFile(fs::FS &fs, const char * path1, const char * path2){
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2)) {
        Serial.println("File renamed");
    } else {
        Serial.println("Rename failed");
    }
}

/*Delete file*/
```

```
void deleteFile(fs::FS &fs, const char * path){
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("File deleted");
    } else {
        Serial.println("Delete failed");
    }
}

/*Test Read&White speed*/
void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file){
        len = file.size();
        size_t flen = len;
        start = millis();
        while(len){
            size_t toRead = len;
            if(toRead > 512){
                toRead = 512;
            }
            file.read(buf, toRead);
            len -= toRead;
        }
        end = millis() - start;
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }
    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }

    size_t i;
    start = millis();
    for(i=0; i<2048; i++){
        file.write(buf, 512);
    }
    end = millis() - start;
    Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
    file.close();
}

void setup(){
    Serial.begin(115200);
    if(!SD_MMC.begin()){
        Serial.println("Card Mount Failed");
        return;
    }
    uint8_t cardType = SD_MMC.cardType();

    if(cardType == CARD_NONE){
        Serial.println("No SD_MMC card attached");
        return;
    }

    Serial.print("SD_MMC Card Type: ");
    if(cardType == CARD_MMC){
        Serial.println("MMC");
    } else if(cardType == CARD_SD){
        Serial.println("SDSC");
    } else if(cardType == CARD_SDHC){
        Serial.println("SDHC");
    } else {
        Serial.println("UNKNOWN");
    }

    uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
    Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);
    //Print filelist
    listDir(SD_MMC, "/", 0);
    //Create a folder named mydir in the root
   createDir(SD_MMC, "/mydir");
    //Print file list and the folder mydir is available
    listDir(SD_MMC, "/", 0);
    //Delete the folder mydir
    removeDir(SD_MMC, "/mydir");
    //Print file list and the folder mydir be cleared away
    listDir(SD_MMC, "/", 2);
    //Create a file hello.txt and white Hello into the file
    writeFile(SD_MMC, "/hello.txt", "Hello ");
    //Append World!\r to the end of Hello in the hello.txt
    appendFile(SD_MMC, "/hello.txt", "World!\n");
    //Read hello.txt and its content "Hello World!" will be printed to the Se
rial Port
    readFile(SD_MMC, "/hello.txt");
    //Delete foo.txt in the SD card
    deleteFile(SD_MMC, "/foo.txt");
    //Rename hello.txt as foo.txt
    renameFile(SD_MMC, "/hello.txt", "/foo.txt");
    //Read foo.txtand its contentHello World! will be printed to the Serial P
ort
    readFile(SD_MMC, "/foo.txt");
    //Test Read&White speed in SD card
    testFileIO(SD_MMC, "/test.txt");
}


void loop(){


}
```

- Result

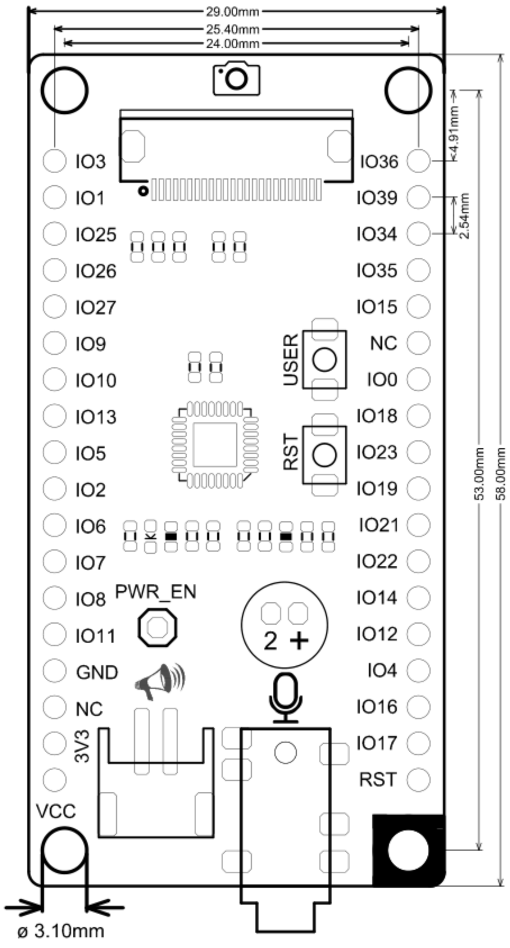


```
SD_MMC Card Type: SDSC
SD_MMC Card Size: 1890MB
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /System Volume Information
  FILE: /test1.wav  SIZE: 39624342
  FILE: /photo1.bmp  SIZE: 153654
Creating Dir: /mydir
Dir created
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /System Volume Information
  DIR : /mydir
  FILE: /test1.wav  SIZE: 39624342
  FILE: /photo1.bmp  SIZE: 153654
Removing Dir: /mydir
Dir removed
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /System Volume Information
Listing directory: /System Volume Information
  FILE: /System Volume Information/WPSettings.dat  SIZE: 12
  FILE: /System Volume Information/IndexerVolumeGuid  SIZE: 76
  FILE: /test1.wav  SIZE: 39624342
  FILE: /photo1.bmp  SIZE: 153654
Writing file: /hello.txt
File written
Appending to file: /hello.txt
Message appended
Reading file: /hello.txt
Read from file: Hello World!
Deleting file: /foo.txt
File deleted
Renaming file /hello.txt to /foo.txt
File renamed
Reading file: /foo.txt
Read from file: Hello World!
1048576 bytes read for 654 ms
1048576 bytes written for 931 ms
```

Compatibility Test

MCU	Support	Not Support
FireBeetle-Board328P		√
FireBeetle-ESP32	√	
FireBeetle-ESP8266		√

Dimension Diagram



Camera&Audio Media Board:Dimension Diagram

More Documents

- Schematic
- 3D Files
- OV7725 Datasheet
- NAU8822LDataSheetRev1.2 Datasheet