Getting Started with Xiao ESP32 S3 Sense

Will guide you to get started with Xiao ESP32 S3 Sense.

Difficulty **Easy** | Duration **2 hour(s)** | Categories **Electronics** | Cost **25 USD ($)**

Contents

Introduction

The Xiao esp32 s3 sense is a tiny but powerful development board that integrates the ESP32-S3R8 processor, which supports 2.4GHz Wi-Fi and Bluetooth 5.0 wireless connectivity, 8MB PSRAM, 8MB flash memory, and a rich set of interfaces. The Xiao esp32 s3 sense also comes with a detachable OV2640 camera sensor, a digital microphone, and an SD card slot, which enable various applications in the fields of intelligent voice and vision AI.
The xiao esp32 s3 sense is compatible with the Arduino IDE and MicroPython, which makes it easy to program and use. It also supports low-power consumption modes, battery charging, and multiple themes and information display. The Xiao esp32 s3 sense is suitable for space-limited projects, such as wearable devices, IoT devices, and embedded ML devices.
In this article, I will show you how to get started with the Xiao esp32 s3 sense, how to use the camera and SD card features, and how to customize and optimize the board. Let's begin!

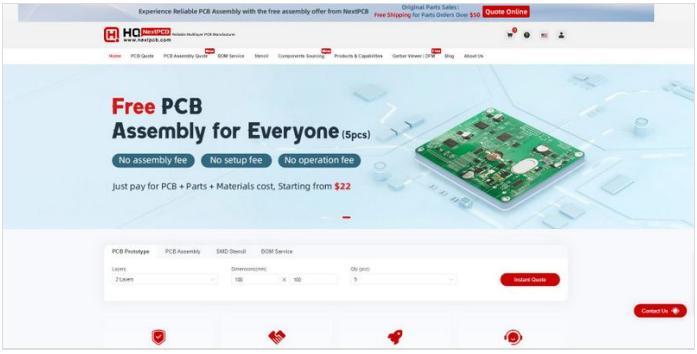
Materials

Hardware components:-
Seeed Studio XIAO ESP32S3 Sense

Software apps and online services:-
Arduino IDE

Tools

Step 1 - Get PCBs for Your Projects Manufactured



This project was successfully completed because of the help and support from NextPCB. Guys if you have a PCB project, please visit their website and get exciting discounts and coupons.
NextPCB offers high-quality, reliable PCB starting at $1.9, and multilayer starting at $6.9. Also, everyone can enjoy free PCB assembly for 5 boards! Also, NextPCB is having a year end sale in which anyone can register through their website and get a $30 Coupon which can be used for ordering PCBs.
You can also try HQDFM free online PCB Gerber viewer to check your PCB design and avoid costly errors.

Step 2 - Capturing and Displaying Images

The Xiao esp32 s3 sense comes with a detachable OV2640 camera sensor, which can capture images up to 2 megapixels. The camera sensor is connected to the board via a 24-pin FPC cable, which can be easily plugged in or unplugged. The camera sensor can be mounted on the board using the provided screws, or placed anywhere you want using the extension cable.
To capture and display images, you need to use the CameraWebServer example sketch from the Arduino IDE. This sketch will create a web server on the Xiao esp32 s3 sense, which will allow you to access the camera stream from any web browser on your local network. You can also take snapshots and save them to the SD card or the flash memory.
To use the CameraWebServer sketch, you need to follow these steps:

- Open the Arduino IDE and select the Xiao esp32 s3 sense board from the Tools menu.
- Go to File > Examples > ESP32 > Camera > CameraWebServer and open the sketch.
- In the sketch, find the line that says `#define CAMERA_MODEL_AI_THINKER` and uncomment it. This will select the correct camera model for the Xiao ESP32 s3 sense.

```
#include "esp_camera.h"
#include <WiFi.h>

#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM

#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid = "ELDRADO";
const char* password = "amazon123";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  while(!Serial);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
```

```
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                     for larger pre-allocated frame buffer.
  if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
```

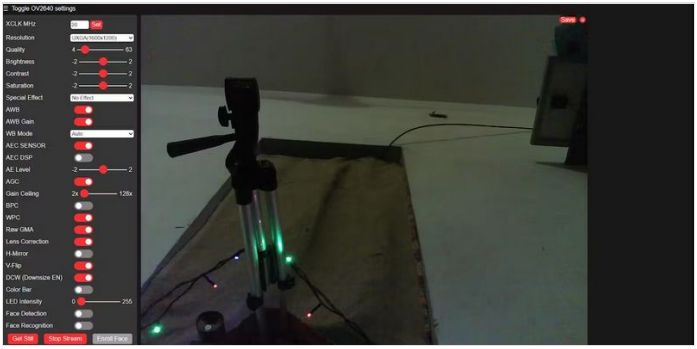
```
  delay(10000);
}
```

- In the sketch, find the line that says `const char* ssid = "your-ssid";` and replace `your-ssid` with the name of your Wi-Fi network. Do the same for the line that says `const char* password = "your-password";` and replace `your-password` with the password of your Wi-Fi network.
- Upload the sketch to the Xiao esp32 s3 sense and open the serial monitor. You should see the IP address of the web server, such as `http://192.168.1.100`.
- Open any web browser on your computer or smartphone and enter the IP address of the web server. You should see the camera stream and some buttons and options on the web page.

Step 3 - Adjusting the Resolution and Orientation

The xiao esp32 s3 sense can capture images at different resolutions, ranging from 160x120 to 1600x1200. You can change the resolution from the web page by selecting one of the options from the drop-down menu. The higher the resolution, the better the image quality, but the slower the frame rate and the more memory usage.

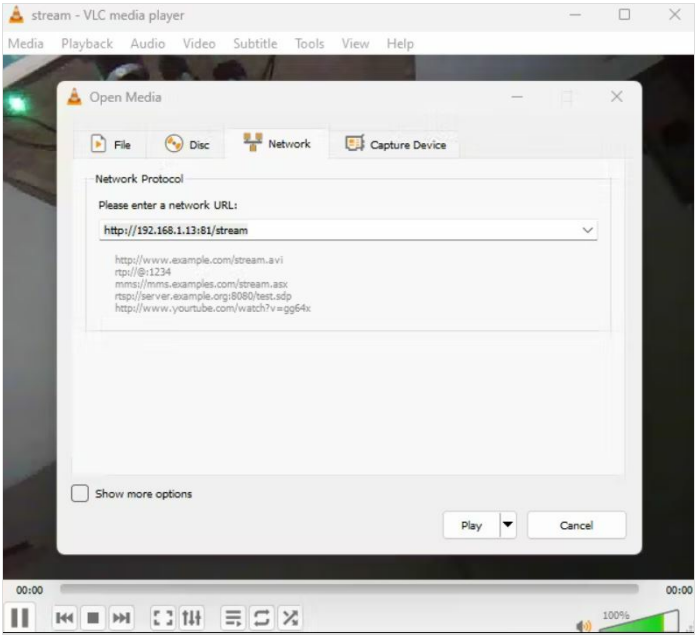
You can also change the orientation of the camera from the web page by clicking on the buttons that say “Flip” or “Rotate”. The flip button will flip the image horizontally, while the rotate button will rotate the image 90 degrees clockwise.



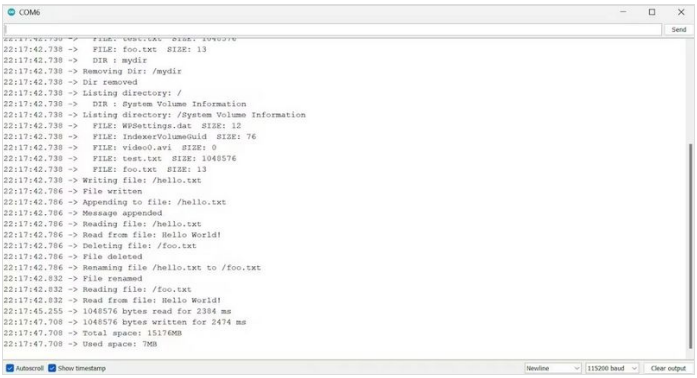
Step 4 - Using the Webcam Web Application

The Xiao esp32 s3 sense can also be used as a webcam for your computer. It allows you to use the camera sensor as a video input device for any software that supports webcams, such as Skype, Zoom, OBS, etc.
To use the webcam web application, you need to follow these steps:

- Install and run the VLC on your computer. You should see a window with a preview of the camera stream and some settings.
- In the settings, enter the IP address of the web server, such as `http://192.168.1.100:81/stream`.
- Click on the “Start” button to start the webcam service.
- Open any software that supports webcams and select the Xiao esp32 s3 sense as the video input device. You should see the camera stream on the software.



Step 5 - Storing and Accessing Data



The Xiao esp32 s3 sense comes with an SD card slot that can support microSD cards up to 32GB. The SD card slot can be used to store and access data, such as images, videos, audio, text, etc. You can also use the SD card as secondary storage for your programs, libraries, or data files.
Before using the SD card on the Xiao ESP32 s3 sense, you need to format the SD card to FAT32 format. This is the most compatible and widely used format for SD cards. You can use any tool or software that can format SD cards to FAT32 format, such as the SD Card Formatter, the Disk Management tool on Windows, the Disk Utility tool on Mac, etc.
After formatting the SD card, you need to insert the SD card into the SD card slot on the Xiao ESP32 s3 sense. Please note the direction of insertion, the side with the gold finger should face inward. The SD card slot has a spring mechanism that will lock the SD card in place. To eject the SD card, you need to press the SD card gently and release it.
To store and access data on the SD card, you need to use the SD library from the Arduino IDE. This library provides functions to create, read, write, delete, and list files and directories on the SD card. You can also use the File object to manipulate the files and directories on the SD card.

```
#include "FS.h"
#include "SD.h"
#include "SPI.h"

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

    File root = fs.open(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.path(), levels -1);
            }
        } else {
            Serial.print("  FILE: ");
            Serial.print(file.name());
            Serial.print("  SIZE: ");
            Serial.println(file.size());
        }
        file = root.openNextFile();
    }
}

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");
    }
}

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");
    }
}

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());
    }
    file.close();
}

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");
    }
```

```
    }
    file.close();
}

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");
    }
    file.close();
}

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");
    }
}

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");
    }
}

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);
    while(!Serial);
    if(!SD.begin(21)){
        Serial.println("Card Mount Failed");
        return;
    }
    uint8_t cardType = SD.cardType();

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

    Serial.print("SD 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.cardSize() / (1024 * 1024);
    Serial.printf("SD Card Size: %lluMB\n", cardSize);

    listDir(SD, "/", 0);
    createDir(SD, "/mydir");
    listDir(SD, "/", 0);
    removeDir(SD, "/mydir");
    listDir(SD, "/", 2);
    writeFile(SD, "/hello.txt", "Hello ");
    appendFile(SD, "/hello.txt", "World!\n");
    readFile(SD, "/hello.txt");
    deleteFile(SD, "/foo.txt");
    renameFile(SD, "/hello.txt", "/foo.txt");
    readFile(SD, "/foo.txt");
    testFileIO(SD, "/test.txt");
    Serial.printf("Total space: %lluMB\n", SD.totalBytes() / (1024 * 1024)
);
    Serial.printf("Used space: %lluMB\n", SD.usedBytes() / (1024 * 1024
));
}

void loop(){

}
```

Step 6 - Wrap-Up

In this article, I have shown how to use the Xiao Esp32 S3 Sense Camera and SD card future. Hope you guys found this helpful. Will see you another one. Bye.