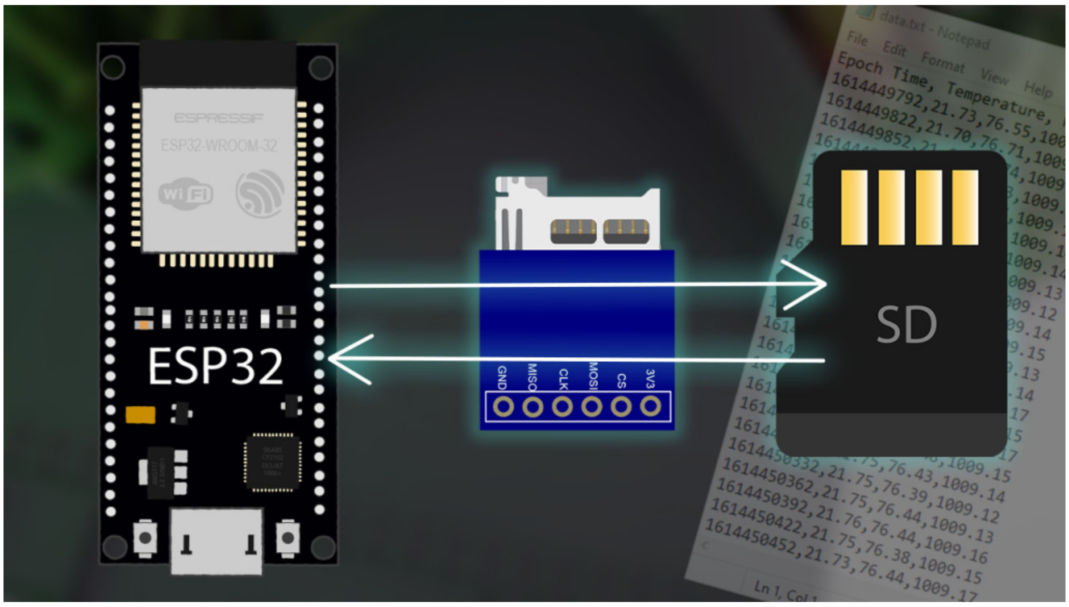
ESP32: Guide for MicroSD Card Module using Arduino IDE

This guide shows how to use a microSD card with the ESP32: you'll learn how to read and write files to the microSD card. To interface the microSD card with the ESP32 board, we'll use a microSD card module (SPI communication protocol). Using a microSD card with the ESP32 is especially useful for data logging or storing files that don't fit in the filesystem (SPIFFS). The ESP32 will be programmed using the Arduino core.

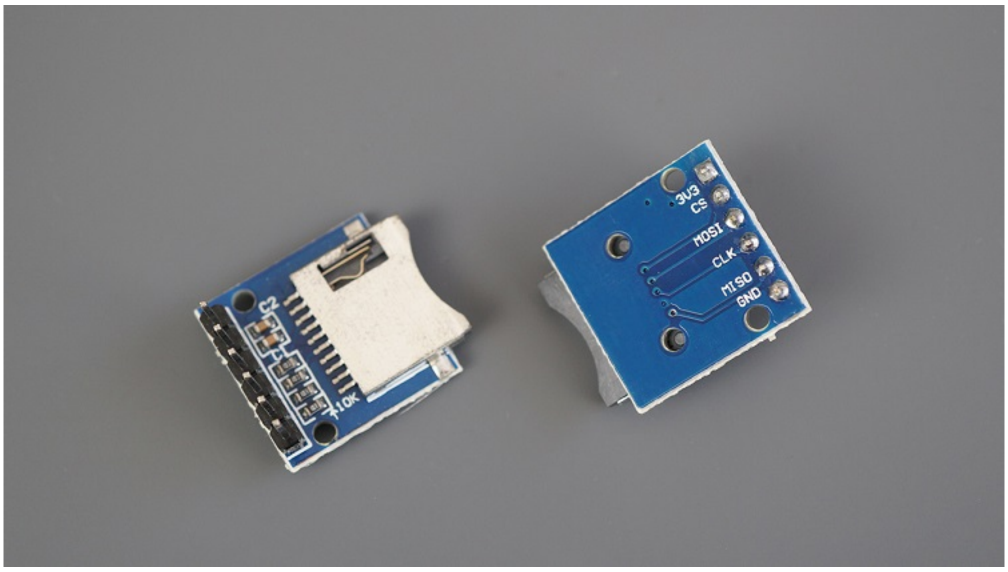


In this tutorial, we'll cover the following topics:

- Introducing the MicroSD Card Module and Pinout
- Interface ESP32 with microSD Card Module
- Testing the microSD Card Module: read, write and handle files
- Use custom SPI pins with the microSD card
- ESP32 Data Logging to microSD Card (sensor readings)
- ESP32 Web Server with Files from microSD Card

MicroSD Card Module

There are different microSD card modules compatible with the ESP32. We're using the microSD card module sown in the following figure – it communicates using SPI communication protocol. You can use any other microSD card module with an SPI interface.



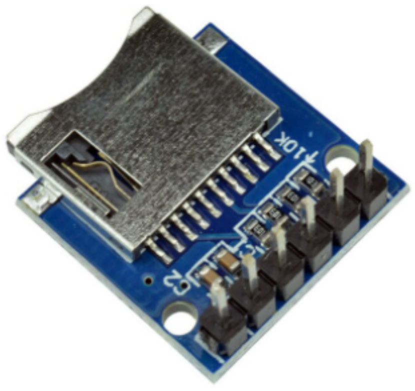
This microSD card module is also compatible with other microcontrollers like the Arduino and the ESP8266 NodeMCU boards. To learn how to use the microSD card module with the Arduino, you can follow the next tutorial:

- Guide to SD Card Module with Arduino

Where to Buy?

You can click the link below to check different stores where you can get the microSD card module:

- MicroSD card module

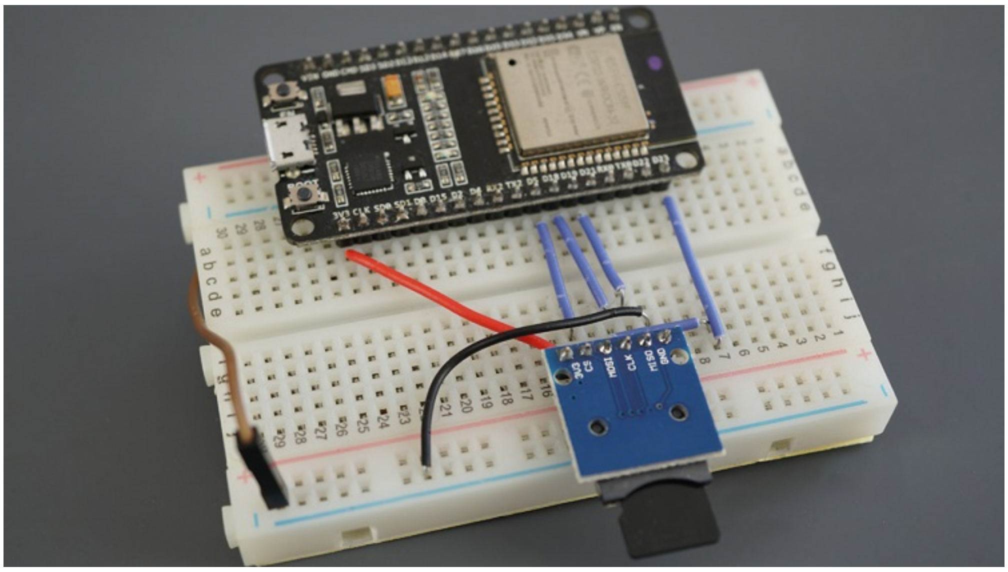


MicroSD Card Module Pinout – SPI

The microSD card module communicates using SPI communication protocol. You can connect it to the ESP32 using the default SPI pins.

MicroSD card module	**ESP32**
3V3	3.3V
CS	GPIO 5
MOSI	GPIO 23
CLK	GPIO 18
MISO	GPIO 19
GND	GND

Parts Required



For this tutorial, you need the following parts:

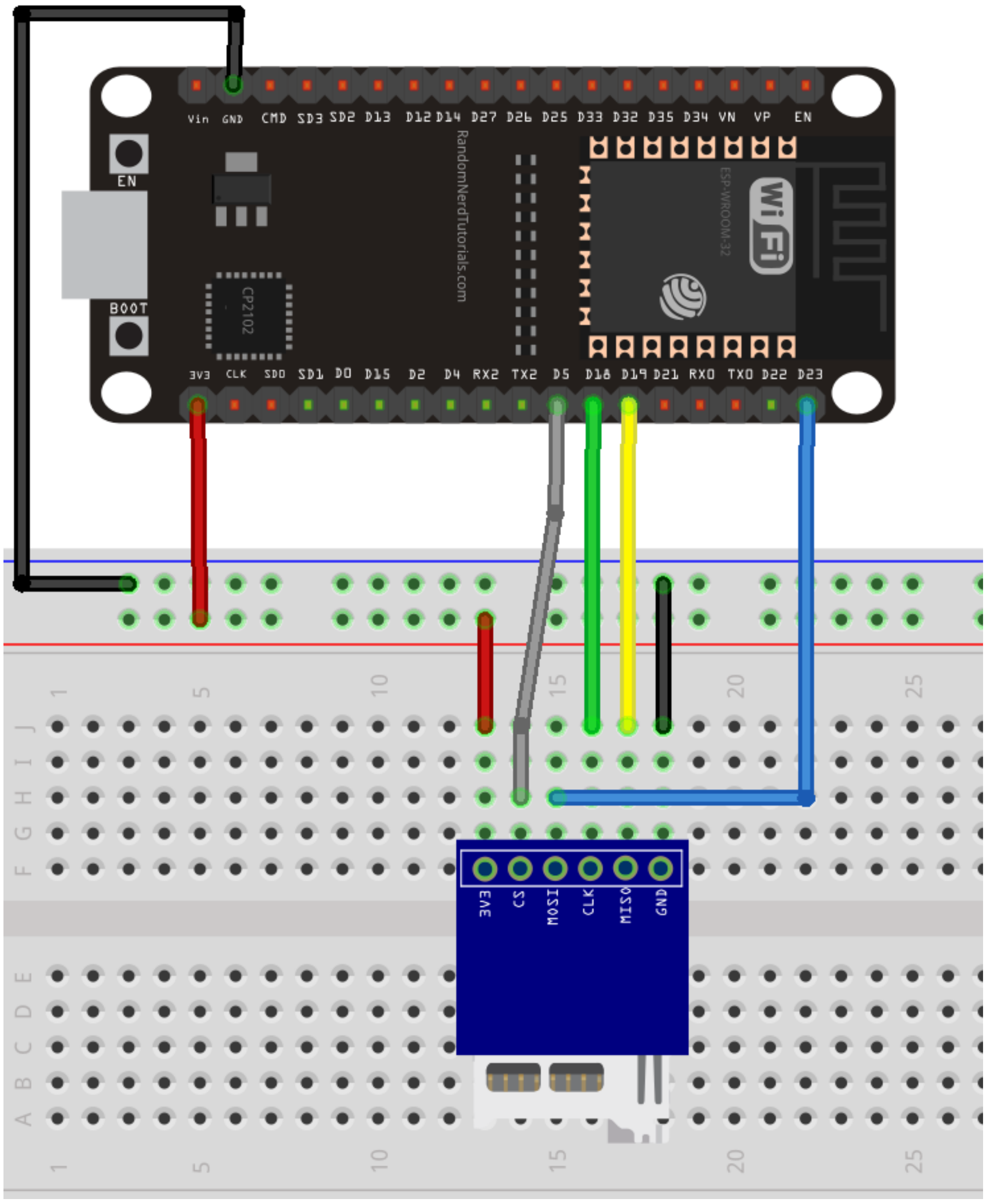
- ESP32 development board (read: Best ESP32 development boards)
- MicroSD Card Module
- MicroSD Card
- Jumper Wires
- Breadboard

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

MAKER ADVISOR

ESP32 with microSD Card Module – Schematic Diagram

To wire the microSD card module to the ESP32 board, you can follow the next schematic diagram (for the default ESP32 SPI pins):

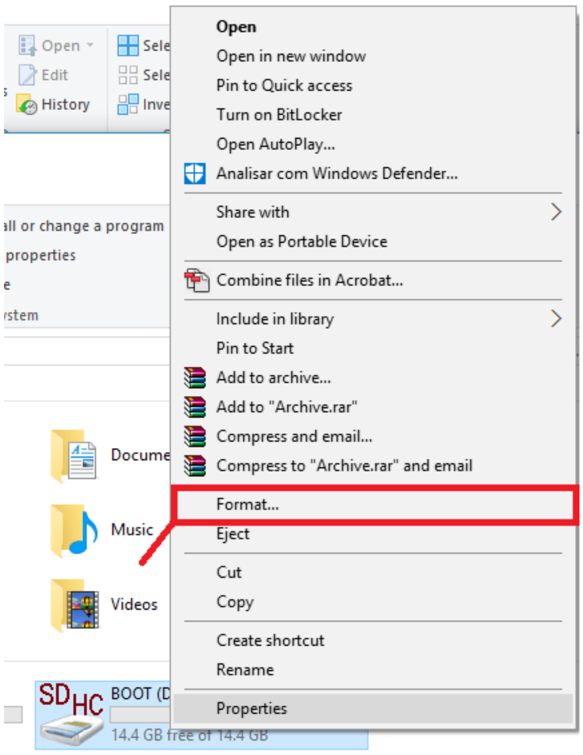


Recommended reading: ESP32 Pinout Reference: Which GPIO pins should you use?

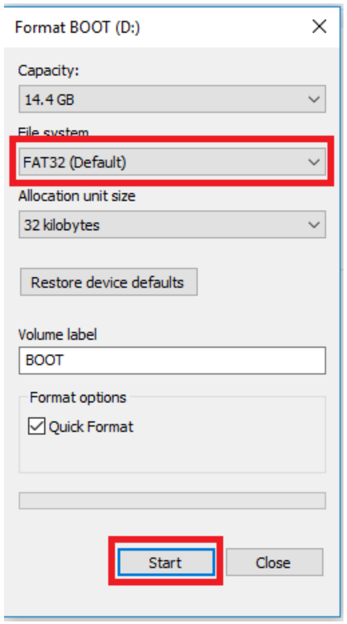
Preparing the microSD Card

Before proceeding with the tutorial, make sure you **format your microSD card as FAT32**. Follow the next instructions to format your microSD card or use a software tool like SD Card Formater (compatible with Windows and Mac OS).

1. Insert the microSD card into your computer. Go to **My Computer** and right-click on the SD card. Select **Format** as shown in the figure below.



2. A new window pops up. Select **FAT32**, press **Start** to initialize the formatting process and follow the onscreen instructions.


Format BOOT (D:)

Capacity:
14.4 GB

File system
FAT32 (Default)

Allocation unit size
32 kilobytes

Restore device defaults

Volume label
BOOT

Format options
☑ Quick Format

Start Close

Preparing Arduino IDE

We'll program the ESP32 board using Arduino IDE. So, make sure you have the ESP32 add-on installed. Follow the next tutorial:

- Install the ESP32 Board in Arduino IDE

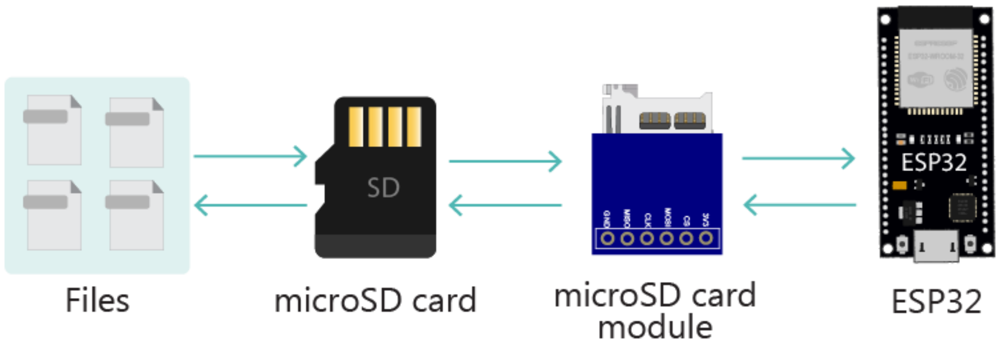
If you prefer using VSCode + PlatformIO, follow the next tutorial instead:

- Getting Started with VS Code and PlatformIO IDE for ESP32 and ESP8266

ESP32 Handling Files with a MicroSD Card Module

There are two different libraries for the ESP32 (included in the Arduino core for the ESP32): the SD library and the SDD_MMC.h library.

If you use the SD library, you're using the SPI controller. If you use the SDD_MMC library you're using the ESP32 SD/SDIO/MMC controller. You can learn more about the ESP32 SD/SDIO/MMC driver.



There are several examples in Arduino IDE that show how to handle files on the microSD card using the ESP32. In the Arduino IDE, go to **File** > **Examples** > **SD(esp32)** > **SD_Test**, or copy the following code.

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-microsd-card-arduino/

  This sketch can be found at: Examples > SD(esp32) > SD_Test
*/

#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);
  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()){
```

View raw code

This example shows how to do almost any task you may need with the microSD card:

- List a directory;
- Create a directory;
- Remove a directory;
- Read a file content;
- Write content to a file;
- Append content to file;
- Rename a file;
- Delete a file;

- Initialize microSD card;
- Get microSD card type;
- Get microSD card size;

Alternatively, you can use the **SD_MMC** examples – these are similar to the SD examples, but use the SDMMC driver. For the SDMMC driver, you need a compatible microSD card module. The module we're using in this tutorial doesn't support SDMMC.

How the Code Works

First, you need to include the following libraries: `FS.h` to handle files, `SD.h` to interface with the microSD card and `SPI.h` to use SPI communication protocol.

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

The example provides several functions to handle files on the microSD card.

List a directory

The `listDir()` function lists the directories on the SD card. This function accepts as arguments the filesystem (`SD`), the main directory's name, and the levels to go into the directory.

```
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();
  }
}
```

Here's an example of how to call this function. The `/` corresponds to the microSD card root directory.

```
listDir(SD, "/", 0);
```

Create a Directory

The `createDir()` function creates a new directory. Pass as an argument the `SD` filesystem and the directory name path.

```
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");
  }
}
```

For example, the following command creates a new directory on the root called `mydir`.

```
createDir(SD, "/mydir");
```

Remove a Directory

To remove a directory from the microSD card, use the `removeDir()` function and pass as an argument the SD filesystem and the directory name path.

```
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");
  }
}
```

Here is an example:

```
removeDir(SD, "/mydir");
```

Read File Content

The `readFile()` function reads the content of a file and prints the content in the Serial Monitor. As with previous functions, pass as an argument the `SD` filesystem and the file path.

```
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();
  }
```

For example, the following line reads the content of the `hello.txt` file.

```
  readFile(SD, "/hello.txt")
```

Write Content to a File

To write content to a file, you can use the `writeFile()` function. Pass as an argument, the `SD` filesystem, the file path and the message

```
  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();
  }
```

The following line writes `Hello` in the `hello.txt` file.

```
  writeFile(SD, "/hello.txt", "Hello ");
```

Append Content to a File

Similarly, you can append content to a file (without overwriting previous content) using the `appendFile()` function.

```
  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();
  }
```

The following line appends the message `World!\n` in the `hello.txt` file. The `\n` means that the next time you write something to the file, it will be written in a new line.

```
  appendFile(SD, "/hello.txt", "World!\n");
```

Rename a File

You can rename a file using the `renameFile()` function. Pass as arguments the SD filesystem, the original filename, and the new filename.

```
  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");
    }
  }
```

The following line renames the `hello.txt` file to `foo.txt`.

```
  renameFile(SD, "/hello.txt", "/foo.txt");
```

Delete a File

Use the `deleteFile()` function to delete a file. Pass as an argument the `SD` filesystem and the file path of the file you want to delete.

```
  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");
    }
  }
```

The following line deletes the `foo.txt` file from the microSD card.

```
  deleteFile(SD, "/foo.txt");
```

Test a File

The `testFileIO()` functions shows how long it takes to read the content of a file.

```
  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();
  }
```

The following function tests the `test.txt` file.

```
  testFileIO(SD, "/test.txt");
```

Initialize the microSD Card

In the `setup()`, the following lines initialize the microSD card with `SD.begin()`.

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

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

If you don't pass any argument to the `begin()` function, it will try to initialize SPI communication with the microSD card on the default chip select (CS) pin. If you want to use another CS pin, you can pass it as an argument to the `begin()` function. For example, if you wanted to use GPIO 17 as a CS pin, you should use the following lines of code:

```
Serial.begin(115200);
if(!SD.begin(17)){
  Serial.println("Card Mount Failed");
  return;
}
uint8_t cardType = SD.cardType();
```

If you want to use custom SPI pins with the microSD card, go to this section.

Get microSD Card Type

The following lines print the microSD card type on the Serial Monitor.

```
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");
}
```

Get microSD Card Size

You can get the microSD card size by calling the `cardSize()` method:

```
uint64_t cardSize = SD.cardSize() / (1024 * 1024);
Serial.printf("SD Card Size: %lluMB\n", cardSize);
```

Testing MicroSD Card Functions

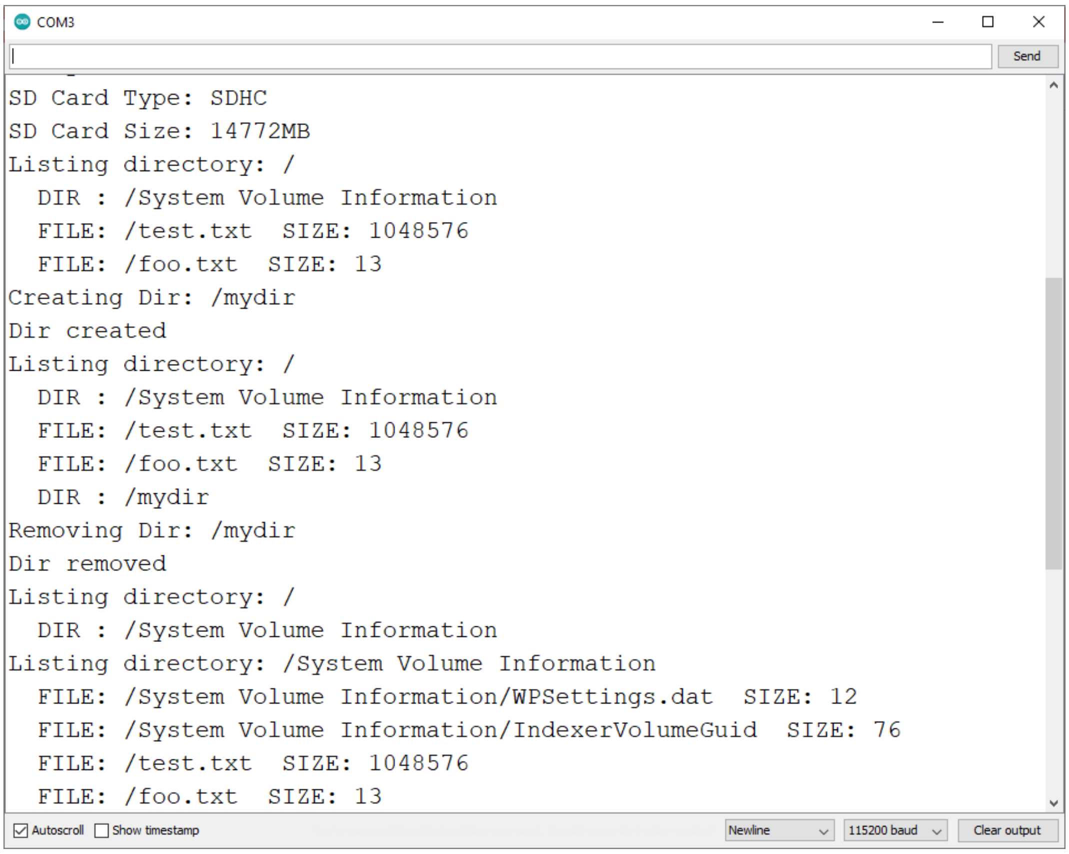
The following lines call the functions we've seen previously.

```
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");
```

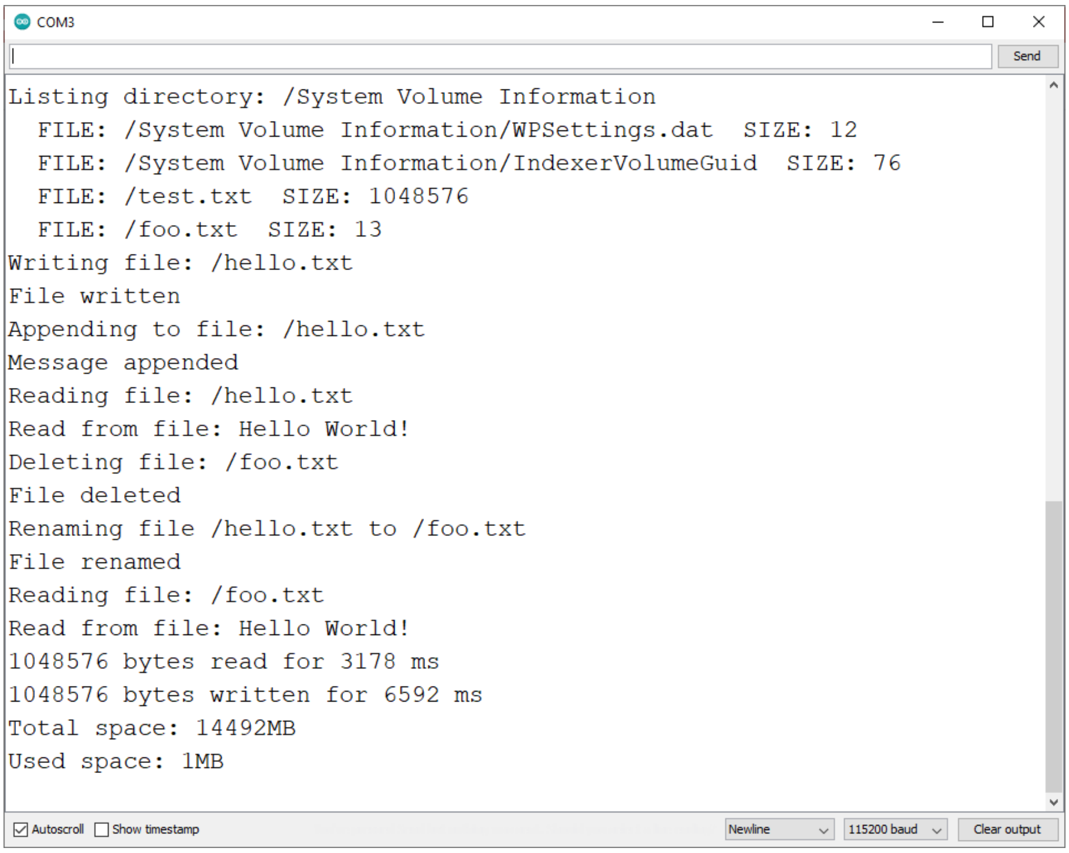
```
  renameFile(SD, "/hello.txt", "/foo.txt");
  readFile(SD, "/foo.txt");
  testFileIO(SD, "/test.txt");
  Serial.printf("Total space: %lluMB\n", SD.totalBytes() / (1024 * 1024));
  Serial.printf("Used space: %lluMB\n", SD.usedBytes() / (1024 * 1024));
```

Demonstration

Upload the previous sketch to your ESP32 board. After that, open the Serial Monitor and press the ESP32 on-board RST button. If the initialization succeeds, you'll get similar messages on the Serial Monitor.



```
SD Card Type: SDHC
SD Card Size: 14772MB
Listing directory: /
  DIR : /System Volume Information
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
Creating Dir: /mydir
Dir created
Listing directory: /
  DIR : /System Volume Information
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /mydir
Removing Dir: /mydir
Dir removed
Listing directory: /
  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: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
```



```
Listing directory: /System Volume Information
  FILE: /System Volume Information/WPSettings.dat  SIZE: 12
  FILE: /System Volume Information/IndexerVolumeGuid  SIZE: 76
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
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 3178 ms
1048576 bytes written for 6592 ms
Total space: 14492MB
Used space: 1MB
```

Newline
115200 baud

Use Custom SPI Pins with the MicroSD Card

The `SD.h` and `SD_MMC.h` libraries use the VSPI SPI pins (23, 19, 18, 5) by default. You can set other pins as SPI pins. The ESP32 features two SPI interfaces: HSPI and VSPI on the following pins:

SPI	**MOSI**	**MISO**	**CLK**	**CS**
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Recommended reading: ESP32 Pinout Reference: Which GPIO pins should you use?

To use other SPI pins, you can proceed as follows:

At the beginning of your code, declare the pins you want to use, for example:

```
#define SCK  17
#define MISO  19
#define MOSI  23
#define CS  5
```

Create a new SPI class on HSPI or VSPI. We're using VSPI. Both will work fine.

```
SPIClass spi = SPIClass(VSPI);
```

In the `setup()`, initialize SPI communication protocol on the pins defined previously:

```
spi.begin(SCK, MISO, MOSI, CS);
```

Finally, initialize the microSD card with the `begin()` method. Pass as argument the CS pin, the SPI instance you want to use, and the bus frequency.

```
if (!SD.begin(CS,spi,80000000)) {
  Serial.println("Card Mount Failed");
  return;
}
```

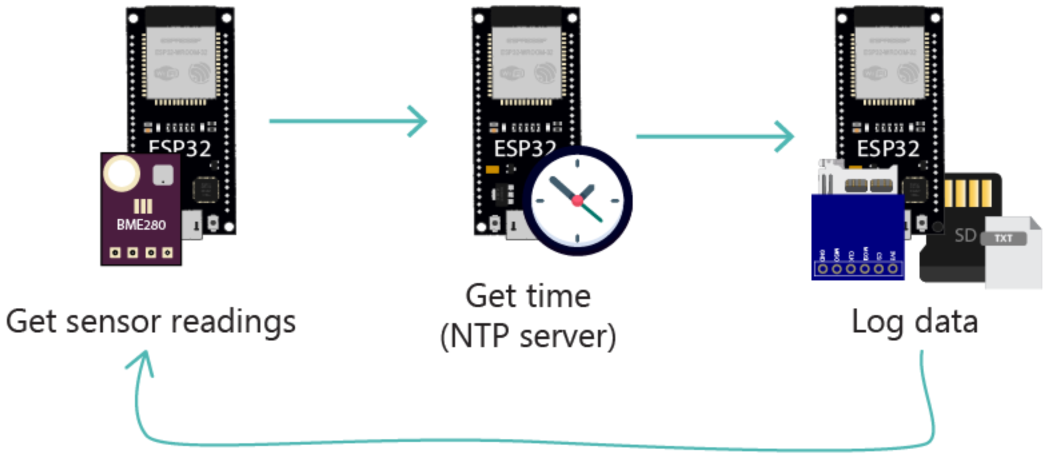
Here is the sample code modified to use custom SPI pins:

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-microsd-card-arduino/

  This sketch was mofidied from: Examples > SD(esp32) > SD_Test
*/

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

#define SCK  17
#define MISO  19
#define MOSI  23
#define CS  5

SPIClass spi = SPIClass(VSPI);

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()){
```

View raw code

Example: ESP32 Data Logging to microSD Card

Using a microSD card is especially useful for data logging projects. As an example, we'll show you how to save sensor readings from a BME280 sensor with timestamps (epoch time).



Prerequisites

For this example, make sure you have the following libraries installed:

- Adafruit BME280 Library
- Adafruit Unified Sensor Driver

You can install these libraries using the Arduino library manager. In your Arduino IDE, go to **Sketch** > **Include Library** > **Manage Libraries…** Then, search for the library names and install them.

If you're using VS Code with PlatformIO, copy the following lines to the *platformio.ini* file to include all the necessary libraries.

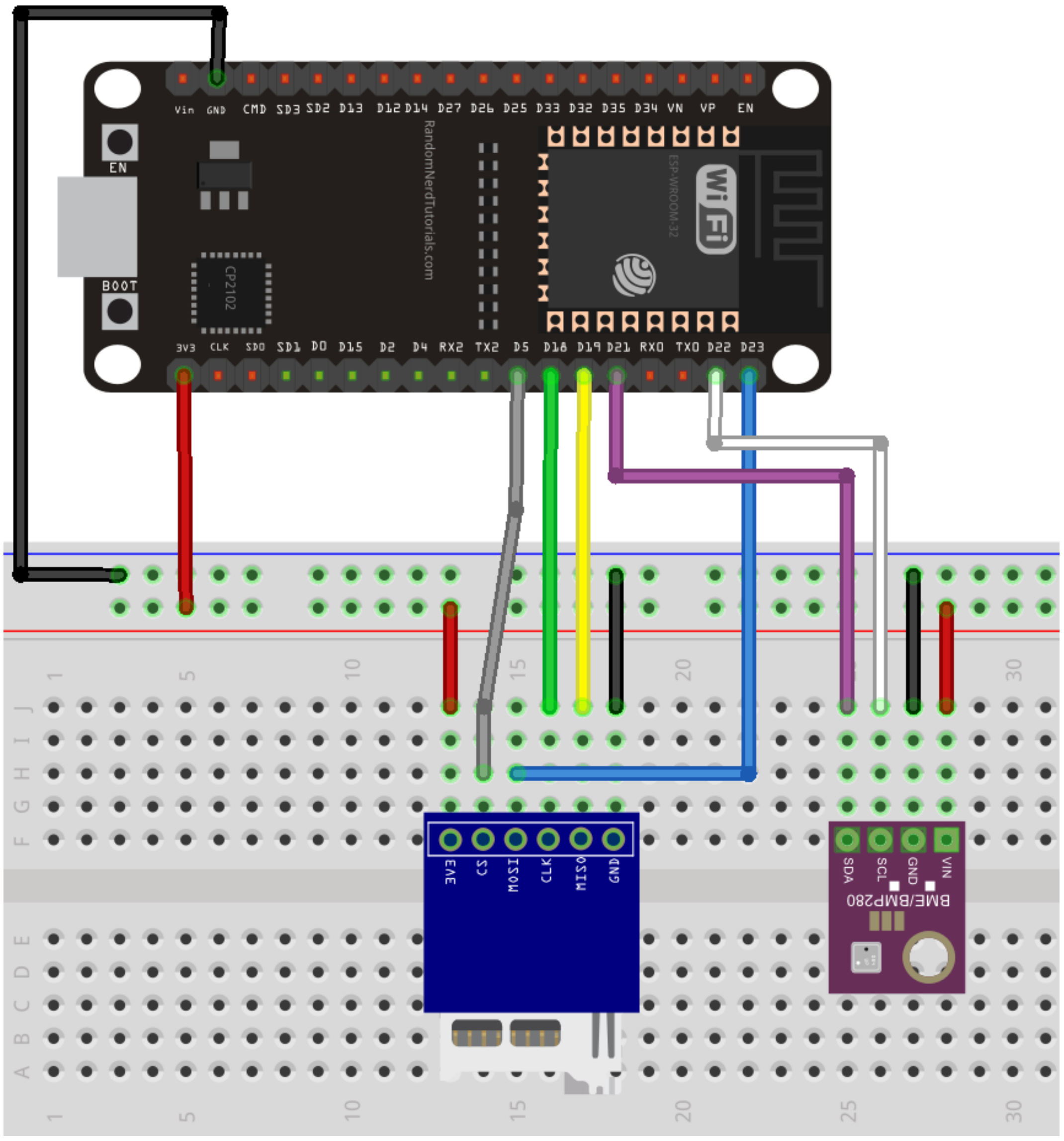
```
lib_deps = adafruit/Adafruit BME280 Library @ ^2.1.0
  adafruit/Adafruit Unified Sensor @ ^1.1.4
```

Schematic Diagram

For this example, you need to wire the microSD card module and the BME280 sensor to the ESP32. Here's a list of the parts required:

- ESP32 development board (read: Best ESP32 development boards)
- BME280 sensor
- MicroSD Card Module
- MicroSD Card
- Jumper Wires
- Breadboard

Wire the circuit by following the next schematic diagram.



You can also take a look at the following tables:

BME280	**ESP32**
VIN	3V3
GND	GND
SCL	GPIO 22
SDA	GPIO 21

microSD card module	**ESP32**
3V3	3.3V
CS	GPIO 5
MOSI	GPIO 23
CLK	GPIO 18
MISO	GPIO 19
GND	GND

Code

Copy the following code to your Arduino IDE. This sketch gets BME280 sensor readings (temperature, humidity, and pressure) and logs them in a file on the microSD card every 30 seconds. It also logs the timestamp (epoch time requested to an NTP server).

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-microsd-card-arduino/

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*/

// Libraries for SD card
#include "FS.h"
#include "SD.h"
#include <SPI.h>

//Libraries for BME280 sensor
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

// Libraries to get time from NTP Server
#include <WiFi.h>
#include "time.h"

// Replace with your network credentials
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

View raw code

Insert your network credentials in the following variables and the code will work straight away:

```
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

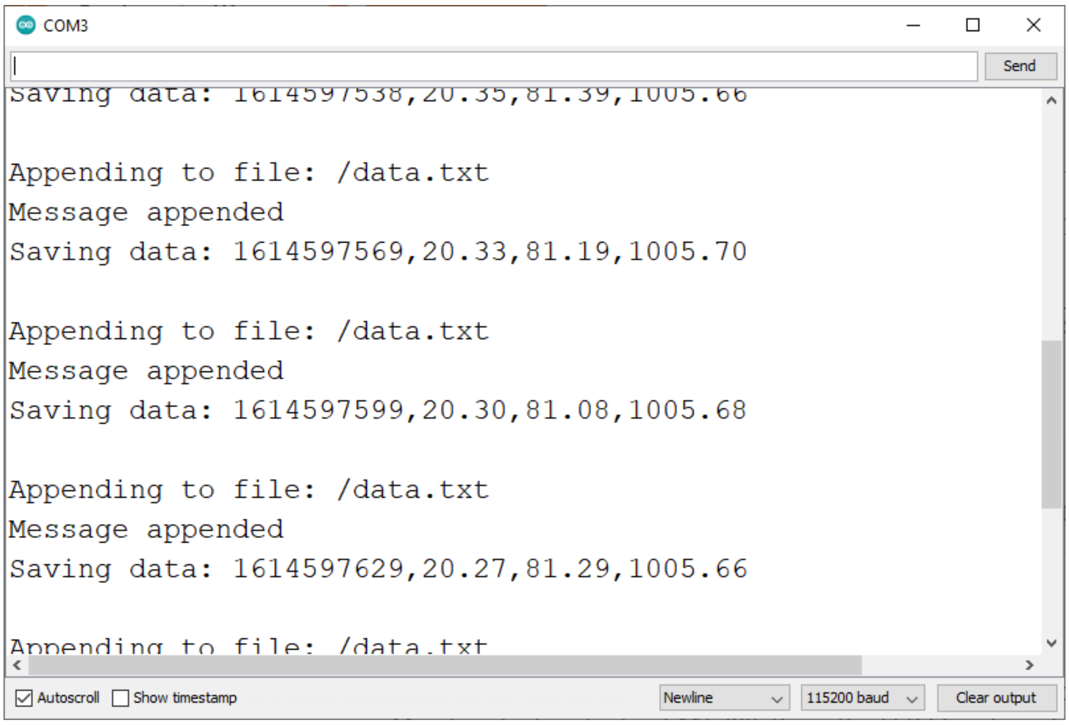
This example uses the functions we've seen previously to write and append data to the microSD card (`writeFile()` and `appendFile()` functions).

To better understand how this example works, we recommend taking a look at the following tutorials:

- Get Epoch/Unix Time with the ESP32 (Arduino)
- ESP32 with BME280 Sensor using Arduino IDE (Pressure, Temperature, Humidity)

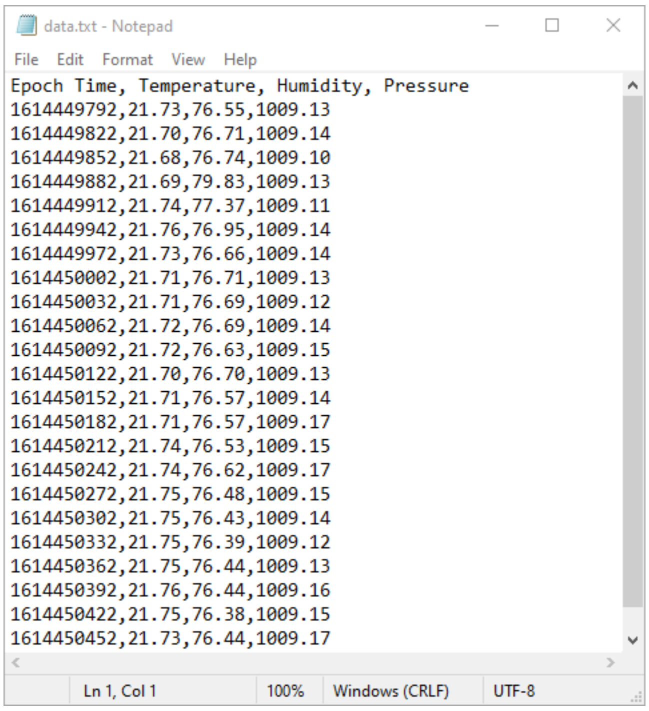
Demonstration

Upload the code to your board. You can check on the Serial Monitor if everything is working as expected.



```
Saving data: 1614597538,20.35,81.39,1005.66

Appending to file: /data.txt
Message appended
Saving data: 1614597569,20.33,81.19,1005.70

Appending to file: /data.txt
Message appended
Saving data: 1614597599,20.30,81.08,1005.68

Appending to file: /data.txt
Message appended
Saving data: 1614597629,20.27,81.29,1005.66

Appending to file: /data.txt
```

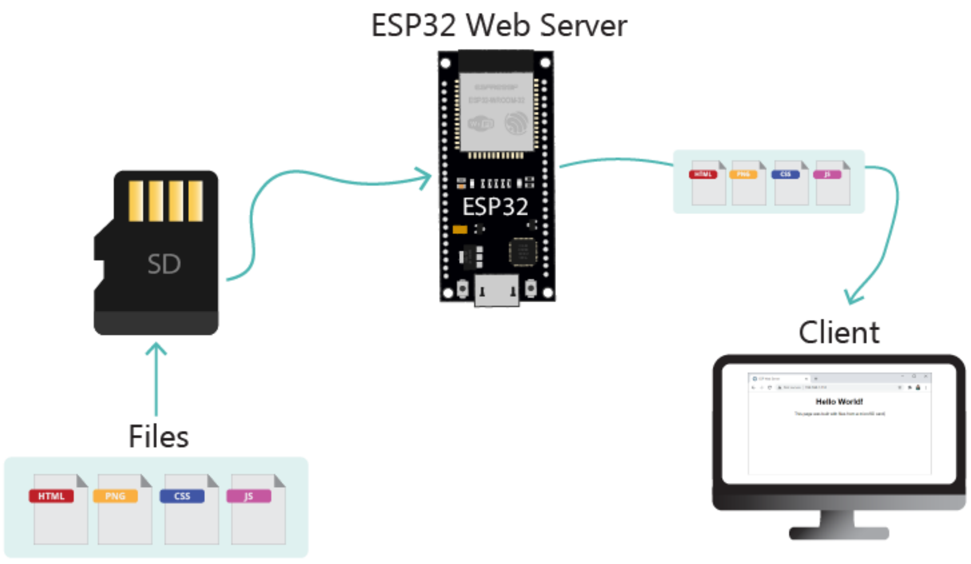
Let the project run for a while to gather some readings. Then, insert the microSD card on your computer, and you should have a file called `data.txt` with the sensor readings.



```
Epoch Time, Temperature, Humidity, Pressure
1614449792,21.73,76.55,1009.13
1614449822,21.70,76.71,1009.14
1614449852,21.68,76.74,1009.10
1614449882,21.69,79.83,1009.13
1614449912,21.74,77.37,1009.11
1614449942,21.76,76.95,1009.14
1614449972,21.73,76.66,1009.14
1614450002,21.71,76.71,1009.13
1614450032,21.71,76.69,1009.12
1614450062,21.72,76.69,1009.14
1614450092,21.72,76.63,1009.15
1614450122,21.70,76.70,1009.13
1614450152,21.71,76.57,1009.14
1614450182,21.71,76.57,1009.17
1614450212,21.74,76.53,1009.15
1614450242,21.74,76.62,1009.17
1614450272,21.75,76.48,1009.15
1614450302,21.75,76.43,1009.14
1614450332,21.75,76.39,1009.12
1614450362,21.75,76.44,1009.13
1614450392,21.76,76.44,1009.16
1614450422,21.75,76.38,1009.15
1614450452,21.73,76.44,1009.17
```

Example: ESP32 Web Server with Files from microSD Card

You can save the files to build a web server with the ESP32 on a microSD card (HTML, CSS, JavaScript, and image files). This can be useful if the files are too big to fit on the ESP32 filesystem, or it can also be more convenient depending on your project.



To show you how to do this, we'll create a simple web server that serves a HTML, a CSS and a PNG file to build a web page and display a favicon.

Move the following files to your microSD card (click on the links to download the files):

- HTML file
- CSS file
- PNG file

Note: move only the files to the microSD card, not the folders.

Then, upload the following code to your Arduino IDE.

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-web-server-microsd-card

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*/

#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "FS.h"
#include "SD.h"
#include "SPI.h"

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

void initSDCard(){
```

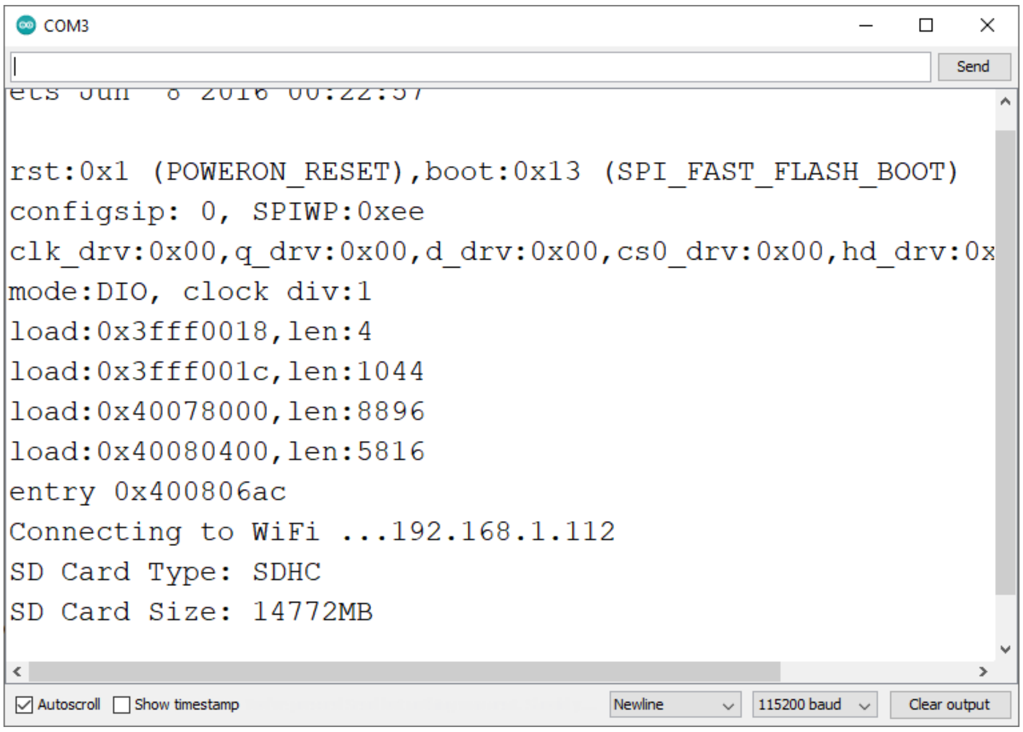
View raw code

Insert your network credentials in the following variables, and the code should work straight away:

```
// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

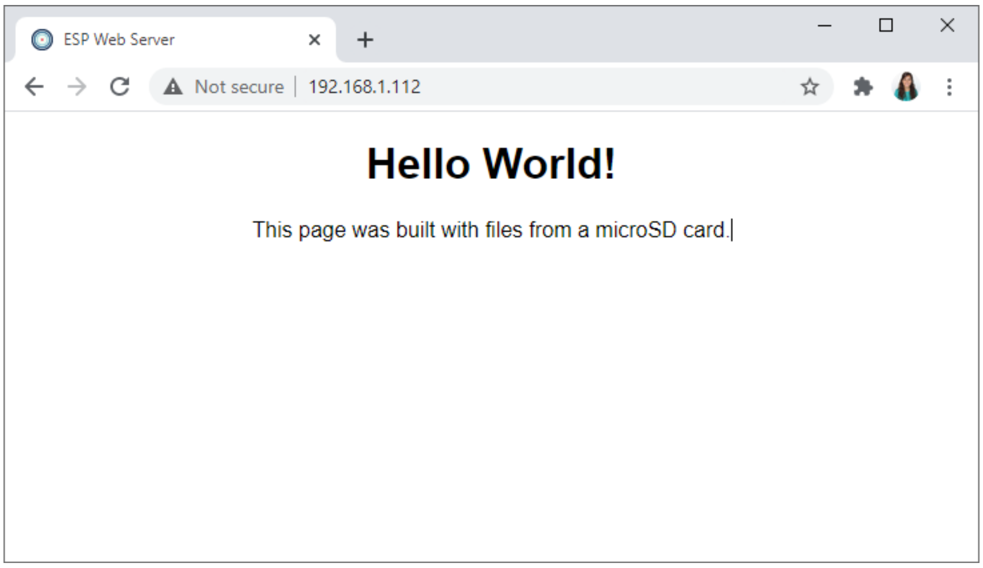
Demonstration

After uploading the files to the microSD card and the sketch to your ESP32 board, open the Serial Monitor at a baud rate of 115200. Press the ESP32 on-board RST button. The ESP32 IP address will be displayed on the Serial Monitor.



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

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
Connecting to WiFi ...192.168.1.112
SD Card Type: SDHC
SD Card Size: 14772MB
```

On your local network, open a web browser and type the ESP32 IP address. You should get access to the following web page built with the files stored on the microSD card.



Hello World!

This page was built with files from a microSD card.

For a detailed explanation of this project, refer to the following tutorial:

- ESP32 Web Server with Files from microSD Card

Wrapping Up

In this tutorial, you've learned how to interface a microSD card with the ESP32 and read and write files. You've also learned how to use it for data logging projects or storing files to serve in your web server projects.

For more in-depth projects with the microSD card, we recommend taking a look at the following:

- ESP32 Data Logging Temperature to MicroSD Card
- ESP32 Web Server with Files from microSD Card

We hope you've found this tutorial useful. Learn more about the ESP32 with our resources:

- Learn ESP32 with Arduino IDE
- Build Web Servers with ESP32 and ESP8266
- More ESP32 Projects and Tutorials…