Raspberry Pi Pico (RP2040) SD Card Example with MicroPython and C/C++

By ShawnHymel

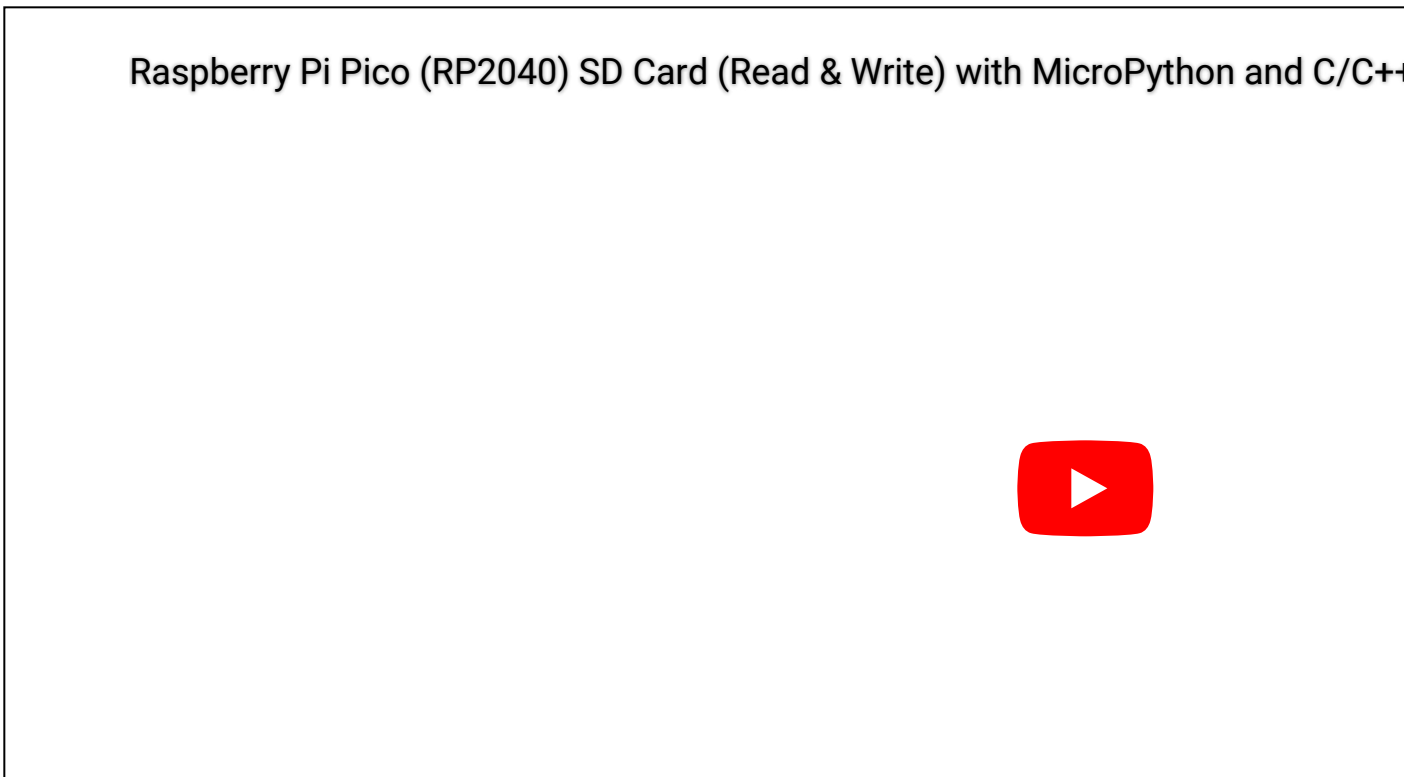
Secure Digital (SD) is a removable flash memory card format originally developed for consumer electronics, such as cameras and MP3 players. The smaller microSD format is perfect for storing event timestamps and sensor logs from a microcontroller project that can be read on a computer.

SD cards support a variety of communication protocols. The most common is the SD bus mode protocol, which uses a 4-bit wide bus to send and receive data. You can read more about SD bus mode here. The most recent version of SD bus mode (UHS-III, not the PCIe lanes of Ex mode) are capable of reaching transfer speeds up to 624 MB/s.

However, we can also use SD cards in a much simpler SPI mode. While transfer rates are slower, we can use fewer pins and rely on easier-to-use SPI peripherals and libraries. SPI mode is great when you're only writing short messages to a file (e.g. event data logging).

In this tutorial, we'll walk you through the process of connecting an SD card to the Raspberry Pi Pico and writing to files using MicroPython and the C/C++ SDK.

You can also view this tutorial in video form:



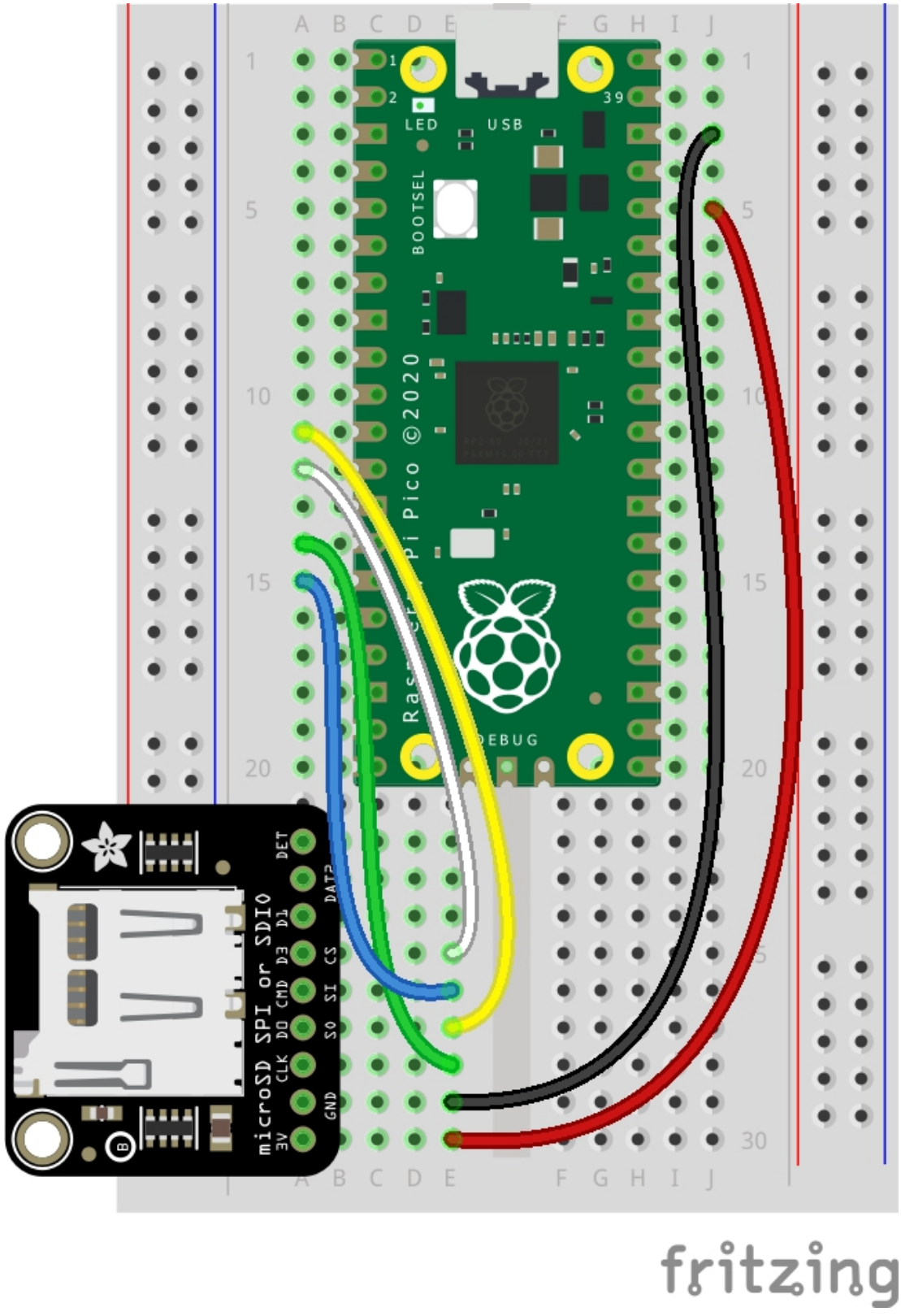
Required Hardware

You will need the following hardware:

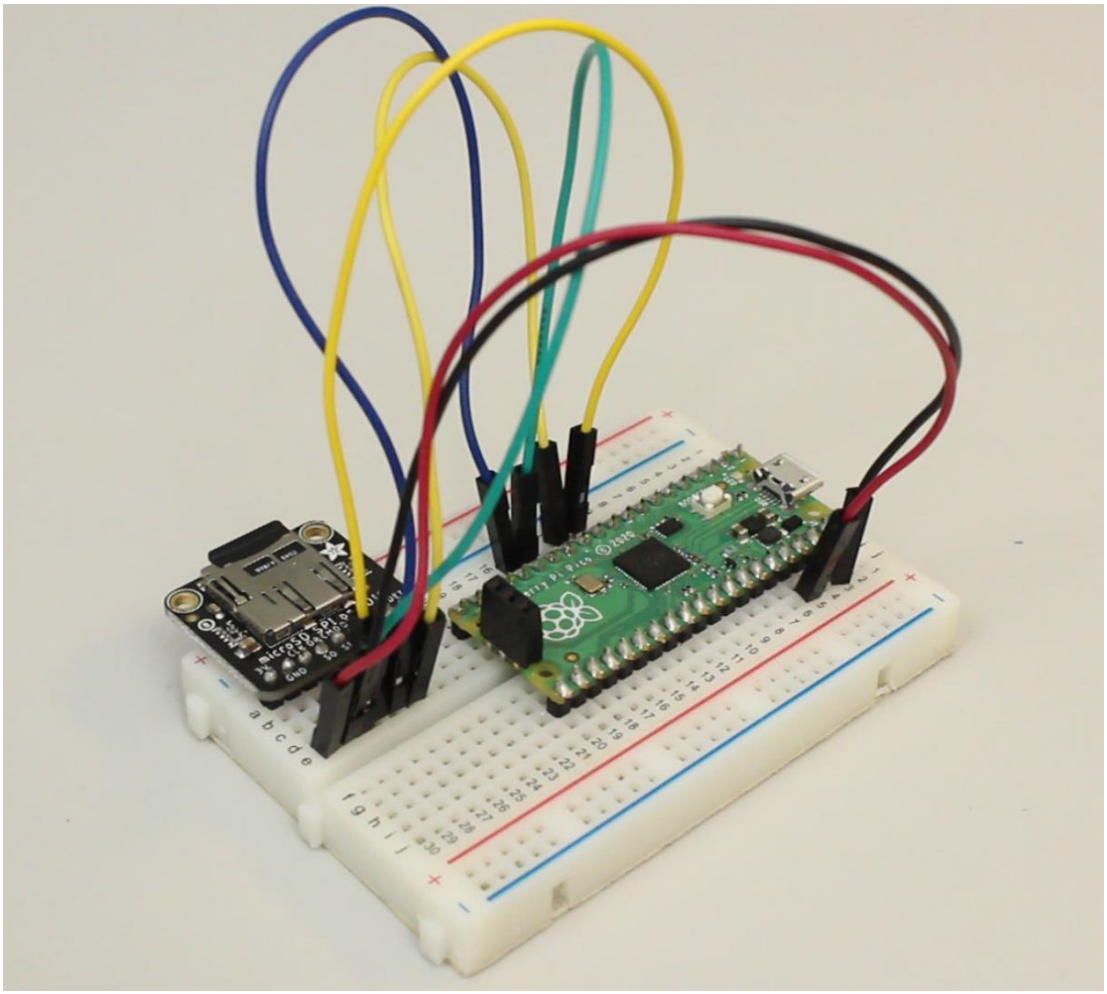
- Raspberry Pi Pico
- MicroSD Card Breakout Board
- MicroSD Card
- Breadboard
- Jumper wires

Hardware Hookup

Connect the sensor to the Pico as follows:

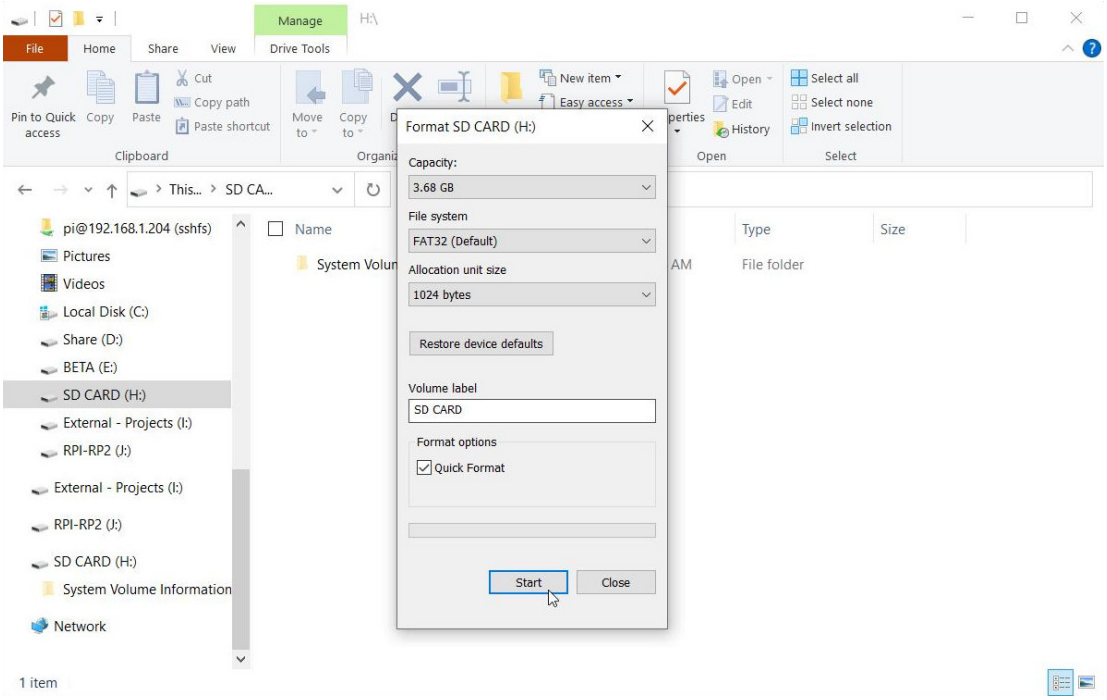


Here is how I connected the microSD card breakout board to the Pico:



Format SD Card

We will need to use the FAT32 filesystem on the SD card in order to read and write files. Insert the SD card into your computer (using an SD port on a laptop or something like a USB SD card reader). I recommend starting with a small block size, such as 512 or 1024 bytes. Feel free to try increasing this value later, once you know everything works.

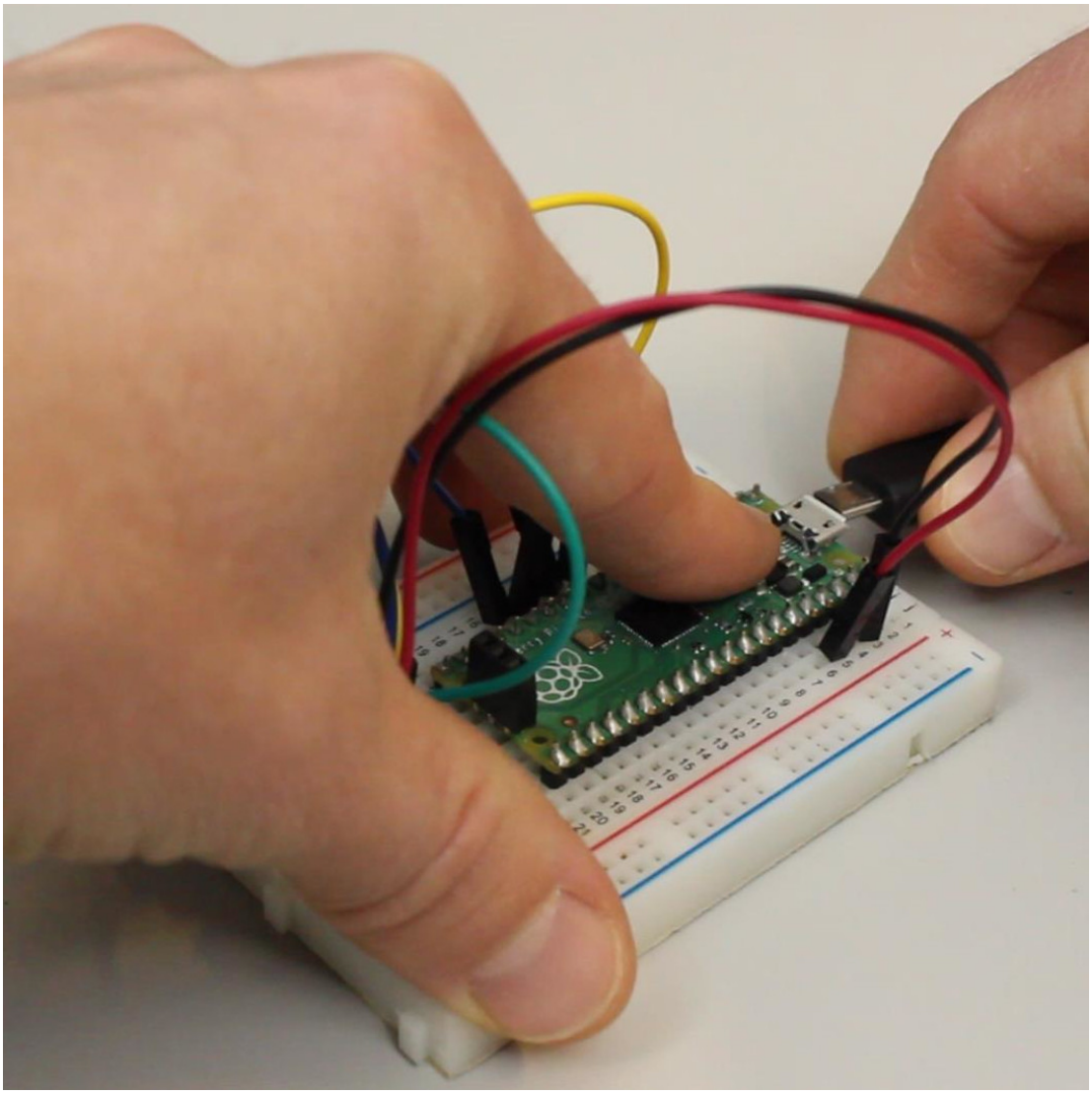


Here is how you can format the SD card on each of the operating systems. Just make sure that you choose “FAT32” as the filesystem type:

- Windows
- Mac
- Linux

Bootloader Mode

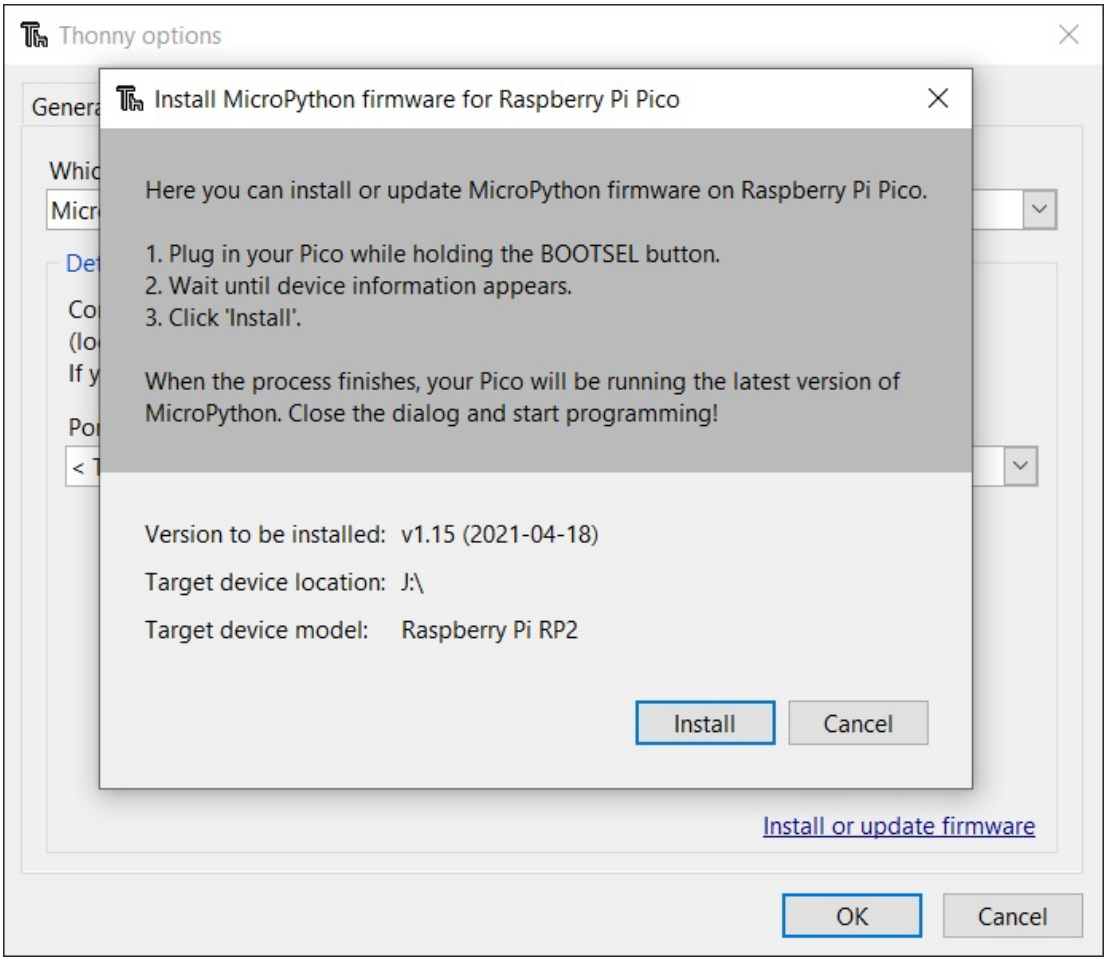
Whenever this guide tells you to put your Pico into “bootloader mode,” you will need to unplug the USB cable. Press and hold the **BOOTSEL** button, and plug the USB cable back in. This will force the Pico to enumerate as a mass storage device on your computer, and you should see a drive appear on your computer with the name “RPI-RP2.”



MicroPython Example

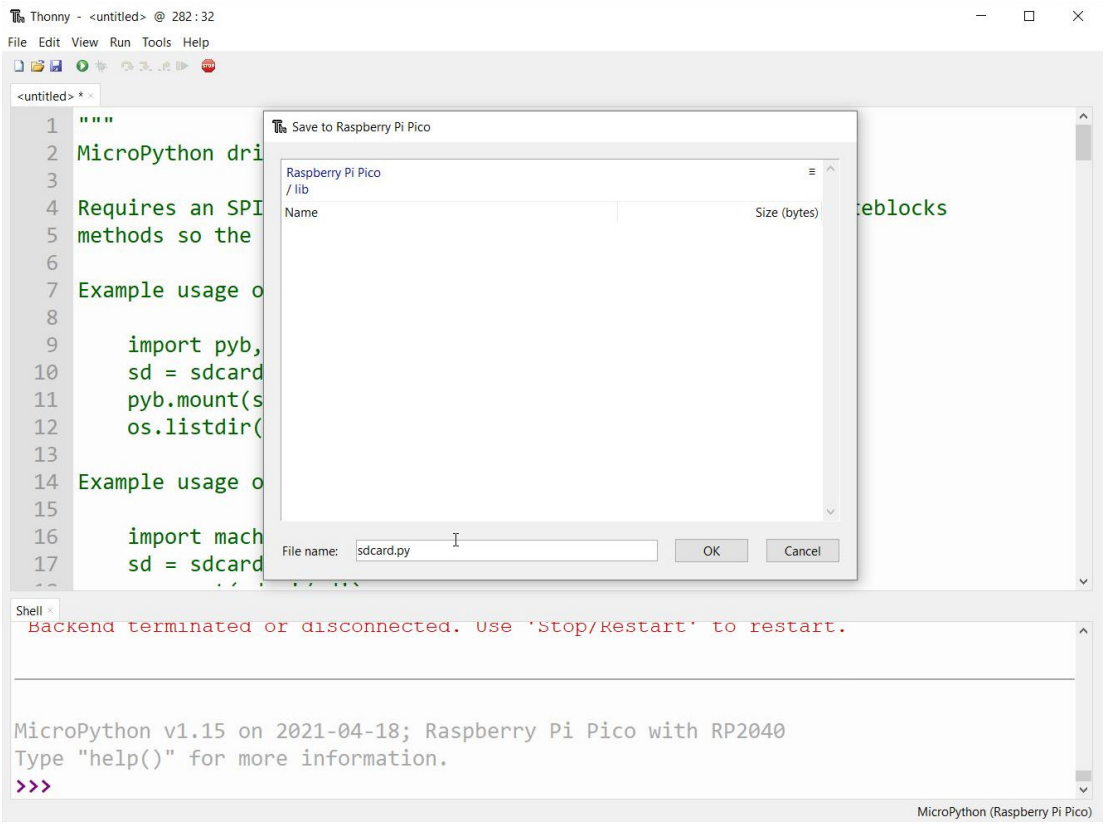
Open Thonny. If you do not already have the MicroPython firmware running on the Pico, click on the bottom-right button and select the Raspberry Pi Pico as your board. Click again and select **Configure Interpreter**. In the pop-up window, select **Install or update firmware**.

Note that you will need MicroPython v1.15 (or later) to support FATFS.



Click **Install** to install the latest MicroPython firmware. Close the pop-up windows when installation is done.

We need to include the sdcard.py driver in order to communicate with the SD card over SPI. Head to the official MicroPython sdcard driver here. Copy the code into a new document in Thonny. Choose to save the file on the Pico device. Create a new folder named *lib*. Save the program as *sdcard.py* in that lib folder.



In a new new document, enter the following code:

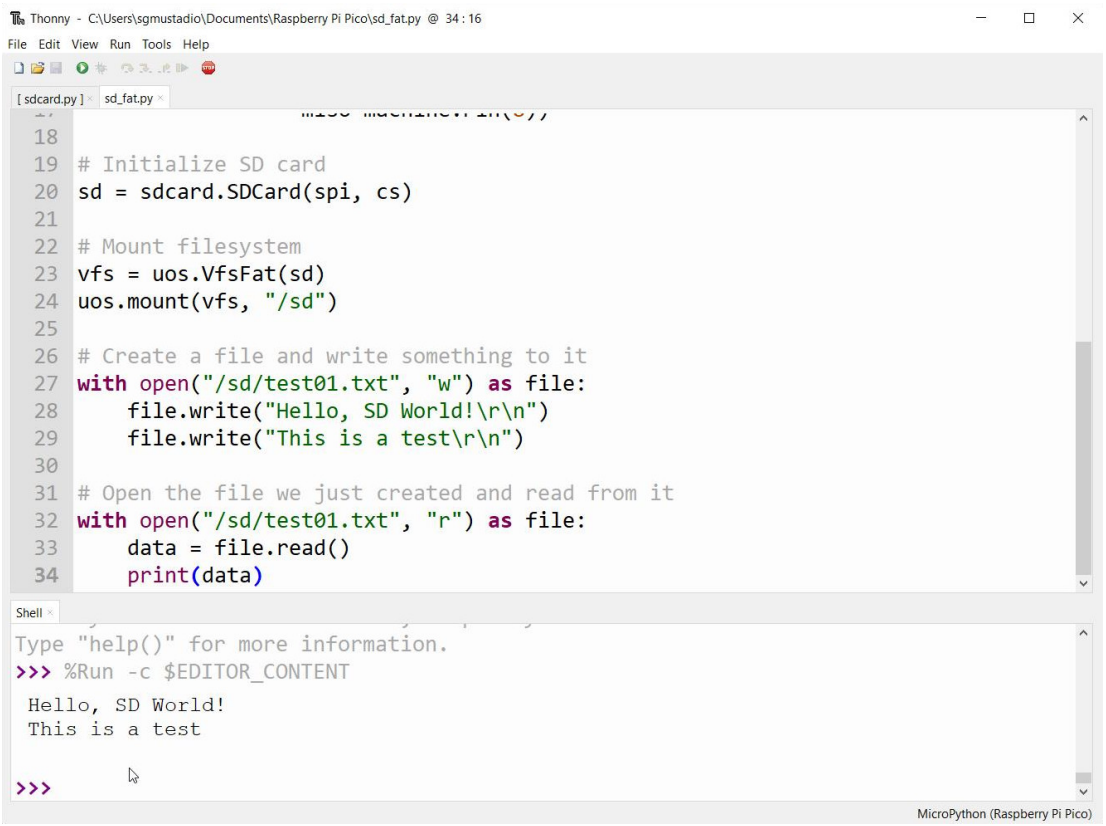
Copy Code

```
import machine
import sdcard
import uos

# Assign chip select (CS) pin (and start it high)
cs = machine.Pin(9, machine.Pin.OUT)

# Intialize SPI peripheral (start with 1 MHz)
spi = machine.SPI(1,
                  baudrate=1000000,
                  polarity=0,
                  phase=0,
                  bits=8,
                  firstbit=machine.SPI.MSB,
                  sck=machine.Pin(10),
                  mosi=machine.Pin(11),
                  miso=machine.Pin(8))

# Initialize SD card
sd = sdcard.SDCard(spi, cs)

# Mount filesystem
vfs = uos.VfsFat(sd)
uos.mount(vfs, "/sd")

# Create a file and write something to it
with open("/sd/test01.txt", "w") as file:
    file.write("Hello, SD World!\r\n")
    file.write("This is a test\r\n")

# Open the file we just created and read from it
with open("/sd/test01.txt", "r") as file:
```

```
data = file.read()
print(data)
```

If you wish, save the program as a file on your computer for safekeeping (e.g. sd_test.py).

Make sure you have the SD card inserted into the breakout board and click the **Run** button. You should see the contents of the file printed out in the shell.



C/C++ Example

If you have not done so already, follow this guide to set up the C/C++ SDK for Pico on your computer and create a Blink program. We will use that Blink program as a template for this project.

Open a file explorer. Create a copy of the *blink* directory you created in the C/C++ setup guide. Rename it to match your project (e.g. *sd_fat_spi*). Delete the *build* directory inside the newly created project folder. Create a new folder named *lib* inside the new project folder.

Open a command prompt with git access (e.g. Git Bash on Windows). Navigate into that lib folder (you might need to change the location depending on where you have your Raspberry Pi Pico projects stored) and clone the no-OS-FatFS-SD-SPI-RPi-Pico project from GitHub user carlk3.

Copy Code

```
cd ~/Documents/Raspberry\ Pi\ Pico/pre_sd_fat_spi/lib/
git clone https://github.com/carlk3/no-OS-FatFS-SD-SPI-RPi-Pico.git
```

This will download carlk3's library that allows us to communicate with an SD card connected through a SPI port.

Open VS Code. Click **File > Open Folder**. Select your newly created project folder. Open **CMakeLists.txt**. Change the project name (e.g. *blink* to *sd_fat_spi*). Add the subdirectory *lib/no-OS-FatFS-SD-SPI-RPi-Pico/FatFs_SPI* with the *add_subdirectory()* function and add the *pico_stdlib* and *FatFs_SPI* libraries to the *target_link_libraries()* function. You may also want to set the USB or UART serial output, depending on if you are using a picoprobe for

debugging (e.g. enable UART serial output for picoprobe, otherwise, use USB serial output). Here is what the CMakeLists.txt file should look like with all these changes:

Copy Code

```
# Set minimum required version of CMake
cmake_minimum_required(VERSION 3.12)

# Include build functions from Pico SDK
include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)

# Set name of project (as PROJECT_NAME) and C/C++ standards
project(sd_fat_spi C CXX ASM)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

# Creates a pico-sdk subdirectory in our project for the libraries
pico_sdk_init()

# Tell CMake where to find the executable source file
add_executable(${PROJECT_NAME}
    main.c
)

# Tell CMake where to find other source code
add_subdirectory(lib/no-OS-FatFS-SD-SPI-RPi-Pico/FatFs_SPI build)

# Create map/bin/hex/uf2 files
pico_add_extra_outputs(${PROJECT_NAME})

# Link to pico_stdlib (gpio, time, etc. functions)
target_link_libraries(${PROJECT_NAME}
    pico_stdlib
    FatFs_SPI
)

# Enable usb output, disable uart output
pico_enable_stdio_usb(${PROJECT_NAME} 1)
pico_enable_stdio_uart(${PROJECT_NAME} 0)
```

In *main.c* replace the code with the following:

Copy Code

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "sd_card.h"
#include "ff.h"

int main() {

    FRESULT fr;
    FATFS fs;
    FIL fil;
    int ret;
    char buf[100];
    char filename[] = "test02.txt";

    // Initialize chosen serial port
    stdio_init_all();

    // Wait for user to press 'enter' to continue
    printf("\r\nSD card test. Press 'enter' to start.\r\n");
    while (true) {
        buf[0] = getchar();
        if ((buf[0] == '\r') || (buf[0] == '\n')) {
            break;
```

```
    }
}

// Initialize SD card
if (!sd_init_driver()) {
    printf("ERROR: Could not initialize SD card\r\n");
    while (true);
}

// Mount drive
fr = f_mount(&fs, "0:", 1);
if (fr != FR_OK) {
    printf("ERROR: Could not mount filesystem (%d)\r\n", fr);
    while (true);
}

// Open file for writing ()
fr = f_open(&fil, filename, FA_WRITE | FA_CREATE_ALWAYS);
if (fr != FR_OK) {
    printf("ERROR: Could not open file (%d)\r\n", fr);
    while (true);
}

// Write something to file
ret = f_printf(&fil, "This is another test\r\n");
if (ret < 0) {
    printf("ERROR: Could not write to file (%d)\r\n", ret);
    f_close(&fil);
    while (true);
}
ret = f_printf(&fil, "of writing to an SD card.\r\n");
if (ret < 0) {
    printf("ERROR: Could not write to file (%d)\r\n", ret);
    f_close(&fil);
    while (true);
}

// Close file
fr = f_close(&fil);
if (fr != FR_OK) {
    printf("ERROR: Could not close file (%d)\r\n", fr);
    while (true);
}

// Open file for reading
fr = f_open(&fil, filename, FA_READ);
if (fr != FR_OK) {
    printf("ERROR: Could not open file (%d)\r\n", fr);
    while (true);
}

// Print every line in file over serial
printf("Reading from file '%s':\r\n", filename);
printf("---\r\n");
while (f_gets(buf, sizeof(buf), &fil)) {
    printf(buf);
}
printf("\r\n---\r\n");

// Close file
fr = f_close(&fil);
if (fr != FR_OK) {
    printf("ERROR: Could not close file (%d)\r\n", fr);
    while (true);
}

// Unmount drive
f_unmount("0:");

// Loop forever doing nothing
```

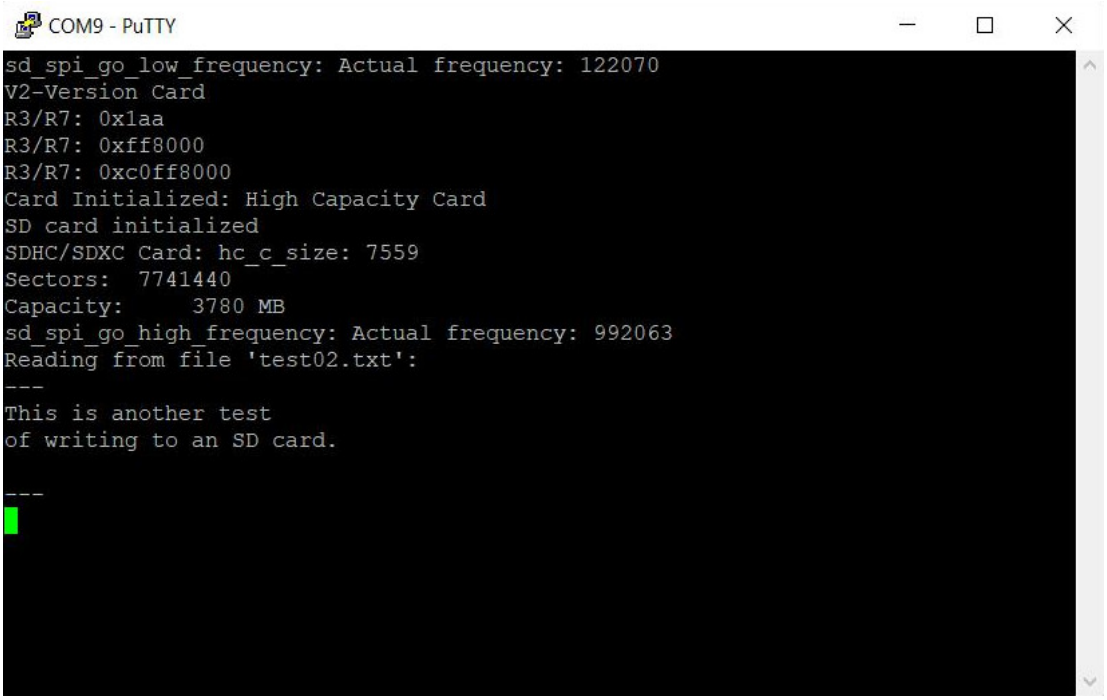
```
        while (true) {
            sleep_ms(1000);
        }
    }
```

Run cmake and make (either from the command line or using the CMake extension in VS Code as outlined in this guide).

If you are using the picoprobe debugger, start debugging to upload the program and click the **Run** button to begin running it.

If you do not have picoprobe set up, put the Pico into bootloader mode and copy *sd_fat_spi.uf2* from the *build* directory to the *RPI-RP2* drive that should have mounted on your computer. Make sure you have the FAT32-formated SD card plugged into the breakout board.

Open your favorite serial terminal program and connect to the Pico with a baud rate of 115200. You might miss the first message being printed to the console, but press 'enter' to start the test. The program will mount the SD card, create a new file, write some lines to it, and then read those lines out to the serial terminal.



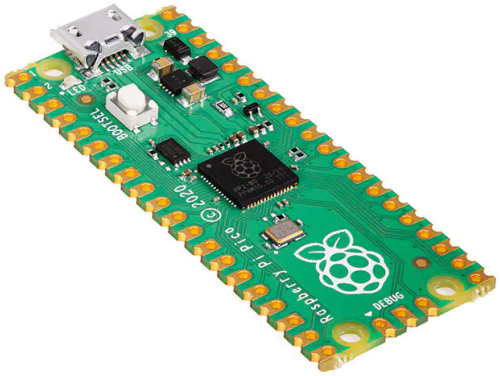
Going Further

I recommend checking out the following documents if you wish to learn more about using SD cards with the Raspberry Pi Pico and RP2040:

- Raspberry Pi Pico Datasheet
- Raspberry Pi Pico MicroPython SDK Guide
- Raspberry Pi Pico C/C++ SDK Guide
- FatFs API Documentation

Key Parts and Components

3 Items

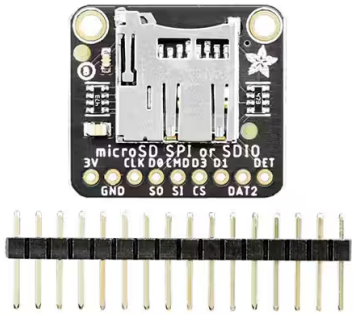


Mfr Part # SC0915

RASPBERRY PI PICO RP2040 BOARD

Raspberry Pi

$4.00 Details



Mfr Part # 4682

MICROSD SPI/SPIO BREAKOUT BOARD

Adafruit Industries LLC

$3.50 Details

Mfr Part # AP8GMCSH4-E

MEMORY CARD MICROSD

Apacer Memory America

$16.02

Add all Digi-Key Parts to Cart

TechForum

Have questions or comments? Continue the conversation on TechForum, Digi-Key's online community and technical resource.

Visit TechForum

Arduino | 3D Printing | Raspberry Pi

Project Details

Development

C

Python

License

Attribution

Get Involved

Like

Save