Getting Started with Seeed Studio XIAO ESP32C3



Seeed Studio XIAO ESP32C3 is an IoT mini development board based on the Espressif ESP32-C3 WiFi/Bluetooth dual-mode chip. ESP32-C3 is a 32-bit RISC-V CPU, which includes an FPU (Floating Point Unit) for 32-bit single-precision arithmetic with powerful computing power. It has excellent radio frequency performance, supporting IEEE 802.11 b/g/n WiFi, and Bluetooth 5 (LE) protocols. This board comes included with an external antenna to increase the signal strength for your wireless applications. It also has a small and exquisite formfactor combined with a single-sided surface-mountable design. It is equipped with rich interfaces and has 11 digital I/O that can be used as **PWM pins** and **4 analog I/O** that can be used as **ADC pins**. It supports four serial interfaces such as **UART**, **I2C**, **SPI and I2S**. There is also a small **reset button** and a **bootloader mode button** on the board. XIAO ESP32C3 is fully compatible with the <u>Grove Shield for Seeeduino</u> XIAO and <u>Seeeduino XIAO Expansion board</u> except for the Seeeduino XIAO Expansion board, the SWD spring contacts on the board will not be compatible.

With regard to the features highlighted above, XIAO ESP32C3 is positioned as a **high-performance**, **low-power**, **cost-effective IoT mini** development board, suitable for **low-power IoT applications and** wireless wearable applications.

This wiki will show you how you can quickly get started with XIAO ESP32C3!

Get One Now 📜

Features 9

- Powerful CPU: ESP32-C3, 32bit RISC-V singlecore processor that operates at up to 160 MHz
- Complete WiFi subsystem: Complies with IEEE 802.11b/g/n protocol and supports Station mode, SoftAP mode, SoftAP + Station mode, and promiscuous mode

- Bluetooth LE subsystem: Supports features of Bluetooth 5 and Bluetooth mesh
- Ultra-Low Power: Deep sleep power consumption is about 43µA
- Better RF performance: External RF antenna included
- Battery charging chip: Supports lithium battery charge and discharge management
- Rich on-chip resources: 400KB of SRAM, and 4MB of onboard flash memory
- Ultra small size: As small as a thumb(20x17.5mm) XIAO series classic form-factor for wearable devices and small projects
- Reliable security features: Cryptographic hardware accelerators that support AES-128/256, Hash, RSA, HMAC, digital signature and secure boot
- Rich interfaces: 1xI2C, 1xSPI, 1xI2S, 2xUART, 11xGPIO(PWM),
 4xADC, 1xJTAG bonding pad interface
- Single-sided components, surface mounting design

Specifications comparison

Processor	ESP32-C3 32-bit	SAMD21	RP2040 Dual-corenRF52840		nRF52840
	RISC-V @160MHz	MO+@48MHz	MO+@133Mhz	M4F@64MHz	M4F@64MHz
Wireless	WiFi and	N/A	N/A	Bluetooth	Bluetooth
Connectivity	Bluetooth 5 (LE)			5.0/BLE/NFC	5.0/BLE/NFC
Memory	400KB SRAM, 4MB	32KB SRAM	264KB SRAM 2MB	256KB RAM, 1MB	256KB RAM, 1MB
	onboard Flash	256KB FLASH	onboard Flash	Flash 2MB onboard Flash	Flash 2MB onboard Flash
Built-in	N/A	N/A	N/A	N/A	6 DOF IMU
Sensors					(LSM6DS3TR-C),
					PDM Microphone
Interfaces	I2C/UART/SPI/I2S	I2C/UART/SPI	I2C/UART/SPI	I2C/UART/SPI	I2C/UART/SPI
PWM/Analog	11/4	11/11	11/4	11/6	11/6
Pins					
Onboard	Reset/ Boot	N/A	Reset/ Boot	Reset Button	Reset Button
Buttons	Button		Button		
Onboard LEDs	Charge LED	N/A	Full-color RGB/	3-in-one LED/	3-in-one LED/
			3-in-one LED	Charge LED	Charge LED
Battery	Built-in	N/A	N/A	BQ25101	BQ25101
Charge Chip					
Programming	Arduino	Arduino/	Arduino/	Arduino/	Arduino/
Languages		CircuitPython	MicroPython/	MicroPython/	MicroPython/
			CircuitPython	CircuitPython	CircuitPython

Hardware overview

Pinout diagram

GPI02 GPI03 GPI04 GPI05 GPI06 GPI07 GPI021	LRC BCLK DIN SDA SCL TX	A0 D0 A1 D1 A2 D2 A3 D3 D4 D4 D5 D4 D6 D4		Seeed SI Model:XIAO-ES FCC (C =CC:Z4T-XIAOE	SP32C3	5V GND 3V3 D10 D9 D8 D7	MOSI MISO SCK RX	GPI010 GPI09 GPI08 GPI020
Digital	Analog	Pin No.	IIS	ш	UART	SPI	GND	Power

Component overview





Power Pins

- 5V This is 5v out from the USB port. You can also use this as
 a voltage input but you must have some sort of diode
 (schottky, signal, power) between your external power source
 and this pin with anode to battery, cathode to 5V pin.
- 3V3 This is the regulated output from the onboard regulator.
 You can draw 700mA
- GND Power/data/signal ground

Getting started

First, we are going to connect XIAO ESP32C3 to the computer, connect an LED to the board and upload a simple code from Arduino IDE to check whether the board is functioning well by blinking the connected LED.

Hardware setup

You need to prepare the following:

- 1 x Seeed Studio XIAO ESP32C3
- 1 x Computer
- 1 x USB Type-C cable

Тір

Some USB cables can only supply power and cannot transfer data. If you don't have a USB cable or don't know if your USB cable can transmit data, you can check <u>Seeed USB Type-C support USB 3.1</u>.

• Step 1. Connect XIAO ESP32C3 to your computer via a USB

Type-C cable.





Step 2. Connect an LED to D10 pin as follows

Note: Make sure to connect a resistor (about 150Ω) in series to limit the current through the LED and to prevent excess current that can burn out the LED

Software setup

NOTE: Currently Seeed Studio XIAO ESP32C3 board is not available on Arduino Board Manager. The board has already been added to esp32 official board libraries but we need to wait until Arduino release the latest version of the board libraries. The current version is 2.0.3 and the version after this will have the XIAO ESP32C3 board library. Because of this, we will manually add the board library.

• **Step 1.** Download and Install the latest version of Arduino IDE according to your operating system

Download Arduino IDE

- Step 2. Launch the Arduino application
- Step 3. Add ESP32 board package to your Arduino IDE

Navigate to File > Preferences, and fill "Additional Boards Manager

URLs" with the url

below: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-

pages/package_esp32_dev_index.json

Preferences	×
Settings Network	
Sketchbook location:	
C:\Users\user\Documents\Arc	duino Browse
Editor language:	System Default v (requires restart of Arduino)
Editor font size:	27
Interface scale:	✓ Automatic 100 🔷 % (requires restart of Arduino)
Theme:	Default theme \lor (requires restart of Arduino)
Show verbose output during:	compilation upload
Compiler warnings:	None v
Display line numbers	Enable Code Folding
🗹 Verify code after upload	Use external editor
✓ Check for updates on star	tup Save when verifying or uploading
Use accessibility features	
Additional Boards Manager UR	Ls: ubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
More preferences can be edite	za airectiy in the file
C:\Users\user\Documents\Ard	uinoData\preferences.txt
(edit only when Arduino is not	running)
	OK Cancel

Navigate to **Tools > Board > Boards Manager...**, type the keyword

"esp32" in the search box, select the latest version of ****esp32****, and install it.



Bonus steps - add XIAO ESP32C3 manually (This step will be changed

after XIAO ESP32C3 shows up on Arduino Board Manager)

Step 4: Navigate

to C:\Users\<username>\Documents\ArduinoData\packages\esp32\har

dware\esp32\2.0.3\variants and create a new folder

named XIAO_ESP32C3

Step 5: Create a new file called pins_arduino.h and add the following

content

#ifndef Pins_Arduino_h
#define Pins_Arduino_h
#include <stdint.h>
#define EXTERNAL_NUM_INTERRUPTS 22
#define NUM_DIGITAL_PINS 22
#define NUM_ANALOG_INPUTS 6
#define analogInputToDigitalPin(p)
(((p) <NUM_ANALOG_INPUTS)?(analogChannelToDigitalPin(p)):-1)
#define digitalPinToInterrupt(p) (((p) <NUM_DIGITAL_PINS)?(p):-1)</pre>

```
#define digitalPinHasPWM(p) (p < EXTERNAL_NUM_INTERRUPTS)
static const uint8 t TX = 21;
static const uint8 t RX = 20;
static const uint8 t SDA = 6;
static const uint8 t SCL = 7;
static const uint8_t SS = 20;
static const uint8 t MOSI = 10;
static const uint8_t MISO = 9;
static const uint8 t SCK = 8;
static const uint8 t A0 = 2;
static const uint8 t A1 = 3;
static const uint8 t A2 = 4;
static const uint8_t A3 = 5;
static const uint8 t D0 = 2;
static const uint8 t D1 = 3;
static const uint8 t D2 = 4;
static const uint8_t D3 = 5;
static const uint8 t D4 = 6;
static const uint8 t D5 = 7;
static const uint8 t D6 = 21;
static const uint8 t D7 = 20;
static const uint8_t D8 = 8;
static const uint8 t D9 = 9;
static const uint8 t D10 = 10;
#endif /* Pins Arduino h */
```

Step 6: Navigate

to C:\Users\<username>\Documents\ArduinoData\packages\esp32\har

dware\esp32\2.0.3 and open boards.txt

Step 7: Add the following content at the end of the file

```
XIAO_ESP32C3. name=XIAO_ESP32C3
XIAO_ESP32C3. vid. 0=0x2886
XIAO_ESP32C3. pid. 0=0x0047
XIAO_ESP32C3. bootloader. tool=esptool_py
XIAO_ESP32C3. bootloader. tool. default=esptool_py
XIAO_ESP32C3. upload. tool=esptool_py
XIAO_ESP32C3. upload. tool. default=esptool_py
XIAO_ESP32C3. upload. tool. default=esptool_py
XIAO_ESP32C3. upload. tool. network=esp_ota
XIAO_ESP32C3. upload. maximum_size=1310720
XIAO_ESP32C3. upload. maximum_data_size=327680
```

XIAO_ESP32C3. upload. flags= XIAO ESP32C3.upload.extra flags= XIAO ESP32C3.upload.use 1200bps touch=false XIAO ESP32C3.upload.wait for upload port=false XIAO ESP32C3.serial.disableDTR=false XIAO ESP32C3.serial.disableRTS=false XIAO ESP32C3. build. tarch=riscv32 XIAO_ESP32C3. build. target=esp XIAO_ESP32C3. build. mcu=esp32c3 XIAO ESP32C3. build. core=esp32 XIAO ESP32C3. build. variant=XIAO ESP32C3 XIAO ESP32C3. build. board=XIAO ESP32C3 XIAO_ESP32C3.build.bootloader_addr=0x0 XIAO ESP32C3. build. cdc on boot=1 XIA0 ESP32C3. build. f cpu=16000000L XIAO ESP32C3. build. flash size=4MB XIAO_ESP32C3.build.flash_freq=80m XIAO ESP32C3. build. flash mode=qio XIAO ESP32C3. build. boot=qio XIAO ESP32C3. build. partitions=default XIAO ESP32C3. build. defines= XIAO ESP32C3.menu.CDCOnBoot.default=Enabled XIAO ESP32C3. menu. CDCOnBoot. default. build. cdc on boot=1 XIAO ESP32C3. menu. CDCOnBoot. cdc=Disabled XIAO ESP32C3. menu. CDCOnBoot. cdc. build. cdc on boot=0 XIAO ESP32C3. menu. PartitionScheme. default=Default 4MB with spiffs (1. 2MB APP/1. 5MB SPIFFS) XIAO ESP32C3. menu. PartitionScheme. default. build. partitions=default XIAO ESP32C3. menu. PartitionScheme. defaultffat=Default 4MB with ffat (1.2MB APP/1.5MB FATFS)XIAO ESP32C3. menu. PartitionScheme. defaultffat. build. partitions=defaul t ffat XIAO ESP32C3. menu. PartitionScheme. default 8MB=8M Flash (3MB APP/1.5MB FAT) XIAO ESP32C3. menu. PartitionScheme. default 8MB. build. partitions=defaul t 8MB XIAO ESP32C3. menu. PartitionScheme. default 8MB. upload. maximum size=334 2336 XIAO ESP32C3. menu. PartitionScheme. minimal=Minimal (1.3MB APP/700KB SPIFFS) XIAO ESP32C3. menu. PartitionScheme. minimal. build. partitions=minimal XIAO ESP32C3. menu. PartitionScheme. no ota=No OTA (2MB APP/2MB SPIFFS) XIAO ESP32C3. menu. PartitionScheme. no ota. build. partitions=no ota XIAO_ESP32C3.menu.PartitionScheme.no_ota.upload.maximum_size=2097152

XIAO_ESP32C3.menu.PartitionScheme.noota_3g=No OTA (1MB APP/3MB SPIFFS)

XIAO_ESP32C3.menu.PartitionScheme.noota_3g.build.partitions=noota_3g XIAO_ESP32C3.menu.PartitionScheme.noota_3g.upload.maximum_size=104857 6

XIAO_ESP32C3.menu.PartitionScheme.noota_ffat=No OTA (2MB APP/2MB FATFS)

XIAO_ESP32C3.menu.PartitionScheme.noota_ffat.build.partitions=noota_f fat

XIAO_ESP32C3.menu.PartitionScheme.noota_ffat.upload.maximum_size=2097 152

XIAO_ESP32C3.menu.PartitionScheme.noota_3gffat=No OTA (1MB APP/3MB FATFS)

XIAO_ESP32C3.menu.PartitionScheme.noota_3gffat.build.partitions=noota _3gffat

XIAO_ESP32C3.menu.PartitionScheme.noota_3gffat.upload.maximum_size=10 48576

XIAO_ESP32C3.menu.PartitionScheme.huge_app=Huge APP (3MB No OTA/1MB SPIFFS)

XIAO_ESP32C3.menu.PartitionScheme.huge_app.build.partitions=huge_app XIAO_ESP32C3.menu.PartitionScheme.huge_app.upload.maximum_size=314572

XIAO_ESP32C3.menu.PartitionScheme.min_spiffs=Minimal SPIFFS (1.9MB APP with OTA/190KB SPIFFS)

XIAO_ESP32C3.menu.PartitionScheme.min_spiffs.build.partitions=min_spiffs

XIAO_ESP32C3.menu.PartitionScheme.min_spiffs.upload.maximum_size=1966
080

XIAO_ESP32C3.menu.PartitionScheme.fatflash=16M Flash (2MB APP/12.5MB FAT)

XIAO_ESP32C3.menu.PartitionScheme.fatflash.build.partitions=ffat

XIAO_ESP32C3.menu.PartitionScheme.fatflash.upload.maximum_size=209715 2

XIAO_ESP32C3.menu.PartitionScheme.app3M_fat9M_16MB=16M Flash (3MB APP/9MB FATFS)

XIAO_ESP32C3.menu.PartitionScheme.app3M_fat9M_16MB.build.partitions=a pp3M fat9M 16MB

XIAO_ESP32C3.menu.PartitionScheme.app3M_fat9M_16MB.upload.maximum_siz e=3145728

XIAO_ESP32C3.menu.PartitionScheme.rainmaker=RainMaker

XIAO_ESP32C3.menu.PartitionScheme.rainmaker.build.partitions=rainmake r

XIAO_ESP32C3.menu.PartitionScheme.rainmaker.upload.maximum_size=31457 28

XIAO_ESP32C3. menu. CPUFreq. 160=160MHz (WiFi) XIAO ESP32C3. menu. CPUFreq. 160. build. f cpu=160000000L XIAO ESP32C3. menu. CPUFreq. 80=80MHz (WiFi) XIAO ESP32C3. menu. CPUFreq. 80. build. f cpu=80000000L XIAO ESP32C3. menu. CPUFreq. 40=40MHz XIAO ESP32C3. menu. CPUFreq. 40. build. f cpu=40000000L XIAO ESP32C3. menu. CPUFreq. 20=20MHz XIA0_ESP32C3.menu.CPUFreq.20.build.f_cpu=2000000L XIAO ESP32C3. menu. CPUFreq. 10=10MHz XIAO ESP32C3. menu. CPUFreq. 10. build. f cpu=10000000L XIAO ESP32C3. menu. FlashMode. gio=QIO XIAO ESP32C3. menu. FlashMode. gio. build. flash mode=dio XIAO_ESP32C3. menu. FlashMode. qio. build. boot=qio XIAO ESP32C3. menu. FlashMode. dio=DIO XIAO ESP32C3. menu. FlashMode. dio. build. flash mode=dio XIAO ESP32C3. menu. FlashMode. dio. build. boot=dio XIAO_ESP32C3.menu.FlashMode.qout=QOUT XIAO ESP32C3. menu. FlashMode. gout. build. flash mode=dout XIAO ESP32C3. menu. FlashMode. gout. build. boot=gout XIAO ESP32C3. menu. FlashMode. dout=DOUT XIAO ESP32C3.menu.FlashMode.dout.build.flash mode=dout XIAO ESP32C3. menu. FlashFreq. 80=80MHz XIAO ESP32C3. menu. FlashFreq. 80. build. flash freq=80m XIAO_ESP32C3.menu.FlashFreq.40=40MHz XIAO_ESP32C3.menu.FlashFreq.40.build.flash freq=40m XIAO ESP32C3. menu. FlashSize. 4M=4MB (32Mb) XIAO ESP32C3. menu. FlashSize. 4M. build. flash size=4MB XIAO ESP32C3. menu. FlashSize. 8M=8MB (64Mb) XIAO_ESP32C3.menu.FlashSize.8M.build.flash_size=8MB XIAO ESP32C3. menu. FlashSize. 8M. build. partitions=default 8MB XIAO ESP32C3. menu. FlashSize. 2M=2MB (16Mb) XIAO ESP32C3. menu. FlashSize. 2M. build. flash size=2MB XIAO_ESP32C3. menu. FlashSize. 2M. build. partitions=minimal XIAO_ESP32C3.menu.FlashSize.16M=16MB (128Mb) XIAO ESP32C3. menu. FlashSize. 16M. build. flash size=16MB XIAO ESP32C3. menu. UploadSpeed. 921600=921600 XIAO ESP32C3. menu. UploadSpeed. 921600. upload. speed=921600 XIAO_ESP32C3. menu. UploadSpeed. 115200=115200 XIAO ESP32C3. menu. UploadSpeed. 115200. upload. speed=115200 XIAO ESP32C3. menu. UploadSpeed. 256000. windows=256000 XIAO ESP32C3. menu. UploadSpeed. 256000. upload. speed=256000 XIAO ESP32C3. menu. UploadSpeed. 230400. windows. upload. speed=256000 XIA0_ESP32C3. menu. UploadSpeed. 230400=230400 XIAO ESP32C3. menu. UploadSpeed. 230400. upload. speed=230400

XIAO ESP32C3. menu. UploadSpeed. 460800. 1inux=460800 XIAO_ESP32C3.menu.UploadSpeed.460800.macosx=460800 XIAO ESP32C3. menu. UploadSpeed. 460800. upload. speed=460800 XIAO ESP32C3. menu. UploadSpeed. 512000. windows=512000 XIAO ESP32C3. menu. UploadSpeed. 512000. upload. speed=512000 XIAO ESP32C3. menu. DebugLevel. none=None XIAO ESP32C3. menu. DebugLevel. none. build. code debug=0 XIAO ESP32C3. menu. DebugLevel. error=Error XIAO ESP32C3. menu. DebugLevel. error. build. code debug=1 XIAO ESP32C3. menu. DebugLevel. warn=Warn XIAO ESP32C3. menu. DebugLevel. warn. build. code debug=2 XIAO ESP32C3. menu. DebugLevel. info=Info XIAO_ESP32C3. menu. DebugLevel. info. build. code_debug=3 XIAO ESP32C3. menu. DebugLevel. debug=Debug XIAO ESP32C3. menu. DebugLevel. debug. build. code debug=4 XIAO ESP32C3. menu. DebugLevel. verbose=Verbose XIAO_ESP32C3. menu. DebugLevel. verbose. build. code_debug=5

Now we have manually added the XIAO ESP32C3 board.

Step 8. Select your board and port

Board

Navigate to Tools > Board > ESP32 Arduino and select

"XIAO_ESP32C3"

Port

Navigate to Tools > Port and select the serial port name of the

connected XIAO ESP32C3. This is likely to be COM3 or higher

(COM1 and COM2 are usually reserved for hardware serial ports).

Blink the LED

• Step 1. Copy the below code to Arduino IDE

```
// define led according to pin diagram
int led = D10;
void setup() {
    // initialize digital pin led as an output
    pinMode(led, OUTPUT);
    }
void loop() {
    digitalWrite(led, HIGH); // turn the LED on
    delay(1000); // wait for a second
    digitalWrite(led, LOW); // turn the LED off
    delay(1000); // wait for a second
}
```

• Step 2. Click the Upload button to upload the code to the

board



Once uploaded, you will see the connected LED blinking with a 1-second delay between each blink. This means the connection is successful and now you can explore more projects with XIAO ESP32C3!

FAQ<mark>1</mark>

Q1: My Arduino IDE is stuck when uploading code to the board

You can first try to reset the board by clicking the **RESET BUTTON** once

while the board is connected to your PC. If that does not work, hold

the **BOOT BUTTON**, connect the board to your PC while holding the **BOOT** button, and then release it to enter **bootloader mode**.

Q2: My board is not showing up as a serial device on Arduino IDE

Follow the same answer as for **Q1** above.

Q3: I want to reflash the bootloader with factory firmware

You can simply connect the board to a PC via **USB Type-C** and reflash the bootloader with factory firmware by using **ESP RF Test Tool**.

- Step 1. Hold on the BOOT BUTTON and connect XIAO
 ESP32C3 to the PC to enter bootloader mode
- Step 2. After it is connected, release the BOOT BUTTON
- Step 3. Visit <u>this page</u> and download ESP RF Test Tool and Test Guide

🕷 ESPRESSIF	Products	Solutions	Support	Ecosystem	Company	Join Us	Contact L	s	Q	中文	Subscribe
Support > Download > Tool	s										
All SDKs & Demos	Apps T	ools AT									
Q Search keywords	- Found 0										
Filter Clear Product ESD32 S3	Flash	Download	d Tools				Expan	d all +	J Do	wnload	selected
ESP32-S2		Title					Platform	Version	Release D	ate 👻	Download
 ESP32-C3 ESP32 		+ Flash Do	wnload Tool	S		,	Windows PC	V3.9.2	2021.11	.10	₹
ESP8266	Certif	ication and	d Test								
Technology Evaluation Kit		Title					Platform	Version	Release D	ate 👻	Download
		+ ESP RF	Test Tool an	d Test Guide			ZIP	V2.8	2021.11	.10	ٹ
		+ ESP8266	i & ESP32 V	VFA Certificati	on and Test Gu	lide	Windows PC	V1.1	2020.08	.05	ل
									J Do	wnload	selected

• Step 4. Extract the .zip, navigate

to ESP RF Test EN\ESP RF Test EN\EspRFTestTool v2.8 Ma $\,$

nual and open EspRFTestTool_v2.8_Manual.exe

ESP_RF_Test_EN > ESP_RF_Test_EN > EspRFTestTool_v2.8_Manual								
Name	Date modified	Туре	Size					
🛅 Bin	11/10/2021 12:01 PM	File folder						
🚞 combin_bin	11/10/2021 12:11 PM	File folder						
🛅 config	11/10/2021 9:27 AM	File folder						
🗖 help	11/10/2021 9:27 AM	File folder						
Power limit table	11/10/2021 12:01 PM	File folder						
SpRFTestTool_v2.8_Manual	6/29/2021 6:01 PM	Application	47,662 KB					
	0/25/2021 0.01 PW	Application	47,002 KB					

• Step 5. Select ESP32C3 as the ChipType, your COM

port, **115200** as the BaudRate and click **open**

EspRFTe	stTool						_		×
Tool Help									
Mannul Te	st								
ChipType	ESP32C3	🗸 сом	COM3	~	BaudRate	115200	~	open	close
IDI F							RAM 🗸	Selec	t Bin
10 22							0%	Load	Bin
WiFi Test	BT Test	WiFiAdapti	ivity Mar	nual					

You will see the following output

Log
DEBUG:['COM1', 'COM3', 'COM6', 'COM8']
set to com port!
Com3 is open ser0
DEBUG:open com3 sucess

• Step 6. Select Flash and click Select Bin

EspRFTes	tTool					-		×
Tool Help								
Mannul Tes	t							
ChipType	ESP32C3	∨ сом	COM3	✓ BaudRate	115200	~	open	close
					Ŀ	lash \vee	Selec	t Bin
IDLE						0%	Load	Bin
WiFi Test	BT Test	WiFi ∆dant	ivity Manua					

• Step 7. Select the file starting with ESP32-C3 and click Open

	<mark>``</mark> «	ESP_RF_Test_EN > ESP_RF_Test_EN > EspRFTestTool_v2.	8_Manual > Bin > RF_TEST_BIN	~ C			
/ fol	der				E] - 🔳	?
		Name	Date modified	Туре	Size		
	Ľ	ESP32_RFTest_184_20210927.bin	9/27/2021 2:49 PM	BIN File	187	КВ	
	Г	ESP32-C3_RFTest_108_2b9b157_20211014.bin	10/14/2021 12:42 PM	BIN File	148	КВ	
	Ľ	ESP32-S2_RF_TEST_V200_40M_20200714.bin	6/11/2020 12:45 PM	BIN File	106	КВ	
	ľ	ESP32-S3_RF_Test_Bin_V101_20210623.bin	6/23/2021 1:43 PM	BIN File	154	КВ	
		ESP8266&ESP8285_RFTest_151_26m_20201215.bin	12/15/2020 6:08 PM	BIN File	62	КВ	
		ESP8266&ESP8285_RFTest_151_40M_20201215.bin	12/22/2020 11:58 AM	BIN File	62	КВ	
File	<u>n</u> am	e:		~	firmware(*.bin)		
					<u>O</u> pen	Cancel	

• Step 8. Finally click Load Bin

EspRFTestTool		_	
Tool Help			
Mannul Test			
ChipType ESP32C3 \lor COM	COM3 V BaudRate 115200	~	open close
	3_RFTest_108_2b9b157_20211014.bin	Flash 🗸	Select Bin
IDLE		0%	Load Bin

You will see the following output when flashing is successful

EspRFTest	Tool						_		×
Tool Help									
Mannul Test									
ChipType	ESP32C3	~ co	COM3		BaudRate	115200	~	open	close
SUCC	MAC:10:91:	a8:03:5a:1	C 3_RF	Test_108_	2696157_20211	014.bin	Flash \vee	Selec	t Bin
			_				100%	Load	Bin
WiFi Test	BT Test	WiFiAd	aptivity	Manual					
Test Mode:	:	Wi	i Rate:		BandWdith:		Channel:		
TX continu	Jes	× 11	b 1M	\sim	20M	~	1/2412		~
Attenuation	n(0.25dB)	Dut	y Cycle:		Certificatio	n EN	Certificat	ion Code	e
0		de	fault	\sim	0x1fc000		SRRC		\sim
						start		stop	
Log DEBUG:Writing at 0x0000000 (00 %) DEBUG:Writing at 0x00010000 (83 %) DEBUG:Writing at 0x00014000 (100 %) DEBUG:Wrote 150656 bytes (88631 compressed) at 0x00000000 in 2.0 seconds (effective 608.1 kbit/s) WARNING:Hash of data verified. DEBUG:Leaving DEBUG:Leaving DEBUG:Load bin success! set to com port!									
Com3 is op DEBUG:ope	oen ser0 en com3 su	cess						Log (Clear Save

WiFi Usage

Seeed Studio XIAO ESP32C3 supports WiFi connectivity with IEEE

802.11b/g/n. This wiki will introduce the basics of WiFi usage on this board.

Hardware set up

• Step 1. Connect the included WiFi/ Bluetooth antenna to

the IPEX connector on the board



• Step 2. Connect XIAO ESP32C3 to your computer via a USB

Type-C cable



Scan WiFi networks (Station Mode)

In this example, we are going to use XIAO ESP32C3 to scan available WiFi networks around it. Here the board will be configured in Station (STA) Mode.

```
Step 1. Copy and paste the code below into Arduino IDE
       •
#include "WiFi.h"
void setup()
{
Serial.begin(115200);
// Set WiFi to station mode and disconnect from an AP if it was
previously connected
WiFi. mode (WIFI STA);
WiFi.disconnect();
delay(100);
Serial.println("Setup done");
}
void loop()
{
Serial.println("scan start");
```

```
// WiFi.scanNetworks will return the number of networks found
int n = WiFi.scanNetworks();
Serial.println("scan done");
if (n == 0) {
Serial.println("no networks found");
} else {
Serial.print(n);
Serial.println(" networks found");
for (int i = 0; i < n; ++i) {
// Print SSID and RSSI for each network found
Serial. print (i + 1);
Serial.print(": ");
Serial.print(WiFi.SSID(i));
Serial.print(" (");
Serial.print(WiFi.RSSI(i));
Serial.print(")");
Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN)?" ":"*");
delay(10);
}
}
Serial.println("");
// Wait a bit before scanning again
delay(5000);
}
```

Step 2. Upload the codes and open the Serial Monitor to start scanning

for WiFi networks

```
COM3
                                              ×
                                                 Send
16:07:02.401 -> Setup done
16:07:02.401 -> scan start
16:07:02.401 -> scan done
16:07:02.401 -> 4 networks found
16:07:02.401 -> 1: 2nd Floor V1 (-64)*
16:07:02.401 -> 2: 2nd Floor V2 (-71)*
16:07:02.401 -> 3: Dialog 4G 034 (-83)*
16:07:02.401 -> 4: walawedura (-88)*
16:07:02.401 ->
16:07:02.401 -> scan start
16:07:02.401 -> scan done
16:07:02.401 -> 4 networks found
16:07:02.401 -> 1: 2nd Floor V1 (-63)*
16:07:02.401 -> 2: 2nd Floor V2 (-69)*
16:07:02.401 -> 3: Dialog 4G 034 (-85)*
16:07:02.401 -> 4: walawedura (-88)*
🗹 Autoscroll 🔽 Show timestamp
```

Connect to a WiFi network

In this example, we are going to use XIAO ESP32C3 to connect to a WiFI network.

• Step 1. Copy and paste the code below into Arduino IDE

```
#include <WiFi.h>
const char* ssid = "your-ssid";
const char* password = "your-password";
void setup()
{
    Serial.begin(115200);
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.println();
    Serial.println();
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    }
}
```

```
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
void loop()
{
}
```

Step 2. Upload the codes and open the Serial Monitor to check that the

board is connected to the WiFI network

```
• COM3 - • × Send

16:11:08.316 ->

16:11:08.316 ->

16:11:08.316 -> Connecting to 2nd_Floor_V1

16:11:08.316 -> ....

16:11:08.316 -> WiFi connected

16:11:08.316 -> IP address:

16:11:08.316 -> 192.168.2.116
```

Use as an Access Point

In this example, we are going to use XIAO ESP32C3 as a WiFi access point where other devices can be connected to it. This is similar to WiFi hotspot feature on mobile phones.

```
• Step 1. Copy and paste the code below into Arduino IDE
#include "WiFi.h"
void setup()
{
Serial.begin(115200);
WiFi.softAP("ESP_AP", "123456789");
```

```
}
void loop()
Serial.print("Host Name:");
Serial.println(WiFi.softAPgetHostname());
Serial.print("Host IP:");
Serial.println(WiFi.softAPIP());
Serial.print("Host IPV6:");
Serial.println(WiFi.softAPIPv6());
Serial.print("Host SSID:");
Serial.println(WiFi.SSID());
Serial.print("Host Broadcast IP:");
Serial.println(WiFi.softAPBroadcastIP());
Serial.print("Host mac Address:");
Serial.println(WiFi.softAPmacAddress());
Serial.print("Number of Host Connections:");
Serial.println(WiFi.softAPgetStationNum());
Serial.print("Host Network ID:");
Serial.println(WiFi.softAPNetworkID());
Serial.print("Host Status:");
Serial.println(WiFi.status());
delay(1000);
}
```

Step 2. Upload the codes and open the Serial Monitor to check more

details about the WiFI access point

Step 3. Scan for WiFi networks on a PC or mobile phone and you will be able to connect to this newly created network using the password we specified in the code



Now you will see that the Number of Host Connections on serial

monitor has been updated to 1

```
coms - c x
send
16:20:21.488 -> Host Name:espressif
16:20:21.488 -> Host IP:192.168.4.1
16:20:21.488 -> Host IPV6:0000:0000:0000:0000:0000:0000
16:20:21.488 -> Host SSID:
16:20:21.488 -> Host Broadcast IP:192.168.4.255
16:20:21.488 -> Host mac Address:10:91:A8:03:5A:1D
16:20:21.488 -> Host Network ID:192.168.4.0
16:20:21.488 -> Host Status:255
```

Bluetooth Usage

Seeed Studio XIAO ESP32C3 supports Bluetooth 5 (LE) connectivity.

This wiki will introduce the basics of Bluetooth usage on this board.

Hardware set up

• Step 1. Connect the included WiFi/ Bluetooth antenna to

the IPEX connector on the board



• Step 2. Connect XIAO ESP32C3 to your computer via a USB

Type-C cable



Scan Bluetooth devices

In this example, we are going to use XIAO ESP32C3 to scan available

Bluetooth devices around it.

• Step 1. Copy and paste the code below into Arduino IDE

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan. h>
#include <BLEAdvertisedDevice.h>
int scanTime = 5; //In seconds
BLEScan* pBLEScan;
class MyAdvertisedDeviceCallbacks: public
BLEAdvertisedDeviceCallbacks {
void onResult(BLEAdvertisedDevice advertisedDevice) {
Serial.printf("Advertised Device: %s \n",
advertisedDevice.toString().c str());
}
};
void setup() {
Serial. begin (115200);
Serial.println("Scanning...");
```

```
BLEDevice::init("");
pBLEScan = BLEDevice::getScan(); //create new scan
pBLEScan->setAdvertisedDeviceCallbacks(new
MyAdvertisedDeviceCallbacks());
pBLEScan->setActiveScan(true); //active scan uses more power, but get
results faster
pBLEScan->setInterval(100):
pBLEScan->setWindow(99); // less or equal setInterval value
ł
void loop() {
// put your main code here, to run repeatedly:
BLEScanResults foundDevices = pBLEScan->start(scanTime, false);
Serial.print("Devices found: ");
Serial.println(foundDevices.getCount());
Serial.println("Scan done!");
pBLEScan->clearResults(); // delete results fromBLEScan buffer to
release memory
delay(2000);
}
```

Step 2. Upload the codes and open the Serial Monitor to start scanning

for Bluetooth devices

- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
- c x
-

XIAO ESP32C3 as Bluetooth server

In this example, we are going to use XIAO ESP32C3 as a Bluetooth

server. Here we will search for XIAO ESP32C3 board using a

smartphone and send out strings to display on the serial monitor

Step 1. Copy and paste the code below into Arduino IDE

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
// See the following for generating UUIDs:
// https://www.uuidgenerator.net/
#define SERVICE UUID "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
class MyCallbacks: public BLECharacteristicCallbacks {
void onWrite(BLECharacteristic *pCharacteristic) {
std::string value = pCharacteristic->getValue();
if (value.length() > 0) {
Serial.println("*******");
Serial.print("New value: ");
for (int i = 0; i < value. length(); i++)
Serial.print(value[i]);
Serial.println():
Serial. println("********");
}
}
};
void setup() {
Serial. begin (115200);
BLEDevice::init("MyESP32");
BLEServer *pServer = BLEDevice::createServer();
BLEService *pService = pServer->createService(SERVICE UUID);
BLECharacteristic *pCharacteristic = pService->createCharacteristic(
CHARACTERISTIC_UUID,
BLECharacteristic::PROPERTY READ
BLECharacteristic::PROPERTY WRITE
);
pCharacteristic->setCallbacks(new MyCallbacks());
pCharacteristic->setValue("Hello World");
pService->start();
BLEAdvertising *pAdvertising = pServer->getAdvertising();
pAdvertising->start();
}
void loop() {
// put your main code here, to run repeatedly:
delay(2000);
}
```

• Step 2. Upload the codes and open the Serial Monitor

- Step 3. Download and install LightBlue App on your smartphone
- LightBlue App (Android)
- LightBlue App (Apple)
- Step 4. Open Bluetooth on your phone, bring the phone close

to XIAO ESP32C3, scan for devices and connect

with MyESP32 device



• Step 5. Open the LightBlue app and click Bonded tab



• Step 6. Click CONNECT next to MyESP32



• Step 7. Click the section at the very bottom which

says Readable, Writable

10:41 PM	ت <i>ک</i> اللہ کِ &
← MyESP32	:
ADVERTISEMENT DATA	
DEVICE INFORMATION	
Device Address 10:91:A8:03:5A:1E	
GENERIC ATTRIBUTE	
Service Changed	\rightarrow
GENERIC ACCESS	
Device Name Readable	\rightarrow
Appearance Readable	÷
Central Address Resolution	→
4fafc201–1fb5–459e–8fc 914b	cc-c5c9c331
beb5483e-36e1-4688-b7f b26a8 Readable, Writable	5-ea07361 →

• Step 8. Under Data format drop-down menu, select UTF-8

String



• Step 9. Type "Hello" under WRITTEN VALUES and

click WRITE

10:46 PM	\$\$ \$\$.ml \$\$ ■				
← beb54	← beb5483e-36e1-4688-b7f5				
Able to be	e read from				
✓ Writable Able to be	e written to				
Support X Able to be indications	s notifications/indications subscribed to for notifications/ on changes to the characteristic				
Data format	UTF-8 String				
READ/INDIC	CATED VALUES				
READ AGAIN					
No value rea Tap on one of th begin	d recently he buttons above — if available — to				
WRITTEN V	ALUES				
\b, \f, \n, \r, \t, sequences are su	\x00, \u0000 and \000 escape upported.				
<u>Hello</u>	WRITE				
No value wri Input some data begin	tten recently and tap on the "Write" button to				
No dota					
NO GATA No descriptors associated with this characteristic					

You will see the text string "Hello" output on the serial monitor of

Arduino IDE

COM3	_	
		Send
22:51:29.991 -> *******		
22:51:29.991 -> New value: Hello		
22:51:29.991 -> *******		

Pin Multiplexing

Seeed Studio XIAO ESP32C3 has rich interfaces. There are 11 digital

I/O that can be used as PWM pins and 4 analog inputs that can be used

as ADC pins. It supports four serial communication interfaces such

as UART, I2C, SPI and I2S. This wiki will be helpful to learn about these

interfaces and implement them in your next projects!

Digital 9

Connect a pushbutton to Pin D6 and an LED to Pin D10. Then upload the

following code to control the ON/OFF of LED using the pushbutton.

```
const int buttonPin = D6; // pushbutton connected to digital pin 6
const int ledPin = D10; // LED connected to digital pin 10
int buttonState = 0; // variable for reading the pushbutton status
void setup() {
// initialize the LED pin as an output:
pinMode(ledPin, OUTPUT);
// initialize the pushbutton pin as an input:
pinMode(buttonPin, INPUT);
}
void loop() {
// read the state of the pushbutton value:
buttonState = digitalRead(buttonPin);
// check if the pushbutton is pressed. If it is, the buttonState is
HIGH:
if (buttonState == HIGH) {
// turn LED on:
digitalWrite(ledPin, HIGH);
} else {
// turn LED off:
digitalWrite(ledPin, LOW);
}
}
```

Digital as PWM

Connect an LED to Pin D10. Then upload the following code to see the

LED gradually fading.

```
int ledPin = D10; // LED connected to digital pin 10
void setup() {
// declaring LED pin as output
pinMode(led_pin, OUTPUT);
void loop() {
// fade in from min to max in increments of 5 points:
for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {</pre>
// sets the value (range from 0 to 255):
analogWrite(ledPin, fadeValue);
// wait for 30 milliseconds to see the dimming effect
delay(30);
}
// fade out from max to min in increments of 5 points:
for (int fadeValue = 255; fadeValue >= 0; fadeValue -= 5) {
// sets the value (range from 0 to 255):
analogWrite(ledPin, fadeValue);
// wait for 30 milliseconds to see the dimming effect
delay(30);
}
}
```

Analog

Connect a potentiometer to Pin A0 and an LED to Pin D10. Then upload

the following code to control the blinking interval of the LED by rotating

the potentiometer knob.

```
const int sensorPin = A0;
const int ledPin = D10;
void setup() {
  pinMode(sensorPin, INPUT); // declare the sensorPin as an INPUT
  pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT
```

```
}
void loop() {
// read the value from the sensor:
int sensorValue = analogRead(sensorPin);
// turn the ledPin on
digitalWrite(ledPin, HIGH);
// stop the program for <sensorValue> milliseconds:
delay(sensorValue);
// turn the ledPin off:
digitalWrite(ledPin, LOW);
// stop the program for for <sensorValue> milliseconds:
delay(sensorValue);
}
```

Serial 9

There are 2 serial interfaces on this board:

- USB Serial
- UART0 Serial

By default, USB serial is enabled, which means you can connect the board to a PC via USB Type-C and open serial monitor on Arduino IDE to view data sent via serial.

However, if you want to use UART0 as the serial, you need to connect pin D6 as the TX pin and pin D7 as RX pin with a USB-Serial adapter.



Also, you need to set USB CDC On Boot to Disabled from Arduino IDE.

NOTE: Change photo when board shows up on Arduino Board Manager

File Edit Sketch	Tools Help		
	Auto Format Archive Sketch	Ctrl+T	
sketch_may18 Void s	Fix Encoding & Reload Manage Libraries Serial Monitor Serial Plotter	Ctrl+Shift+I Ctrl+Shift+M Ctrl+Shift+L	
) // F	WiFi101 / WiFiNINA Firmware Updater Board: "ESP32C3 Dev Module" Upload Speed: "921600"	2	n once
}	USB CDC On Boot: "Disabled"		Disabled
void 1	CPU Frequency: "160MHz (WiFi)" Flash Frequency: "80MHz" Flash Mode: "QIO" Flash Size: "4MB (32Mb)"	> > >	Enabled
// F	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)" Core Debug Level: "None" Port: "COM3 (ESP32S3 Dev Module)" Get Board Info)))	repea
}	Programmer Burn Bootloader	>	

Upload the following code to Arduino IDE to send the string "Hello

World!" via serial

```
void setup() {
Serial.begin(115200);
while (!Serial);
}
```

```
void loop() {
Serial.println("Hello World!");
delay(1000);
}
```

The output will be as follows on Arduino Serial Monitor

COM3					-		×
							Send
23:16:05.456 -	->	Hello	World!				- 1
23:16:05.456 -	->	Hello	World!				- 1
23:16:05.456 -	->	Hello	World!				- 1
23:16:05.456 -	->	Hello	World!				- 1
23:16:06.477 -	->	Hello	World!				- 1
23:16:07.454 -	->	Hello	World!				- 1
23:16:08.474 -	->	Hello	World!				- 1
23:16:09.495 -	->	Hello	World!				- 1
23:16:10.466 -	->	Hello	World!				- 1
23:16:11.484 -	->	Hello	World!				- 1
23:16:12.458 -	->	Hello	World!				- 1
23:16:13.477 -	->	Hello	World!				- 1
23:16:14.495 -	->	Hello	World!				- 1
23:16:15.468 -	->	Hello	World!				- 1
Autoscroll 🗹 Show timestamp				No line ending \sim	115200 baud \sim	Clea	r output

12C<mark>1</mark>

Hardware connection

Connect a Grove - OLED Yellow&Blue Display 0.96 (SSD1315) to XIAO

ESP32C3 by following the hardware connection as follows.

SCL	SCL
SDA	SDA
VCC	5V
GND	GND



Software setup

• Step 1. Open Arduino IDE, navigate to sketch > Include

Library > Manage Libraries...

• Step 2. Search for u8g2 and install it

e All V Topic All U8g2	
ore info	
g2	
nochrome LCD, OLED and eInk Library. Display controller: SSD1305, S JD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, S (1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1617, UC163 (7586, ST7588, ST75256, ST75320, NT7534, ST7920, IST3020, IST7 N1661, IL3820, MAX7219. Interfaces: I2C, SPI, Parallel. Monochrome splay controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1320, SSD133 (1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1 (1638, UC1701, ST7511, ST7528, ST7565, ST7567, ST7571, ST7586, ST7588 (7032, KS0108, KS0713, HD44102, T7932, SED1520, SBN1661, IL3820, MAX 700 fonts, U8X8 char output.	SD1306, SSD1309, SSD1316, SSD1320, SSD1322, SSD1325, H1122, T6963, RA8835, LC7981, PCD8544, PCF8812, I8, UC1701, ST7511, ST7528, ST7565, ST7567, ST7571, 920, LD7032, KS0108, KS0713, HD44102, T7932, SED1520, LCD, OLED and eInk Library. Successor of U8glib. Supported 22, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, 230, UC1601, UC1604, UC1610, UC1611, UC1617, 8, ST75256, ST75320, NT7534, ST7920, IST3020, IST7920, K7219. Supported interfaces: I2C, SPI, Parallel. Features: UTF8,
ere lefe	Version 2.31.2 VInstall
g2_for_Adafruit_GFX	
/ oliver dd U8g2 fonts to any Adafruit GFX based graphics library. Use our favorit https://github.com/olikraus/u8g2/wiki/fntlistall). ore_info	te Adafruit graphics library together with fonts from U8g2 project

• Step 3. Upload the following code to display text strings on

the OLED Display

//#include <Arduino.h>
#include <U8g21ib.h>
#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>

```
#endif
#ifdef U8X8 HAVE HW I2C
#include <Wire.h>
#endif
U8G2 SSD1306 128X64 NONAME F SW I2C u8g2 (U8G2 R0, /* clock=*/ SCL, /*
data=*/ SDA, /* reset=*/ U8X8 PIN NONE); //Low spped I2C
void setup(void) {
u8g2.begin();
// u8x8.setFlipMode(1); // set number from 1 to 3, the screen word
will rotary 180
}
void loop(void) {
u8g2.clearBuffer(); // clear the internal memory
u8g2.setFont(u8g2 font ncenB08 tr); // choose a suitable font
u8g2.drawStr(0,15, "Hello World!"); // write something to the internal
memory
u8g2.drawStr(0, 30, "Hello World!");
u8g2. drawStr(0, 40, "Hello World!");
u8g2.sendBuffer(); // transfer internal memory to the display
// delay(1000);
}
```

SPI<u></u>

Hardware connection

Connect a Grove - High Precision Barometric Pressure Sensor

(DPS310) to XIAO ESP32C3 by following the hardware connection as

follows.

3V3	3V3
SDI	MOSI
GND	GND
SDO	MISO
CSK	SCK
CS	CS



Software setup

•

Step 1. Download Seeed_Arduino_DPS310 Library as a zip file

geeed-Studio / Seeed_Arduino_DPS310 (Public)			🛇 Edit Pins 👻 💿 Wat
forked from Infineon/DPS310-Pressure-Sensor			
<> Code 11 Pull requests () Actions	🗄 Projects 🖽 Wiki 😲	Security 🗠 Insights 🕸	Settings
ᢪ dps310 → ᢪ 6 branches ◇6 tag		Go to file	Add file - Code -
This branch is 3 commits ahead, 3 commits l	pehind Infineon:dps310.	▶ Clone	0
		HTTPS SSH GitHub CLI	
Pillar1989 Seeed:Arduino: fix travis.yml	write error	gh repo clone Seeed-Studio	/Seeed_Arduino_1
examples	Pretty printed the Arduino code v	Work fast with our official CLI. Lear	
src src	Pretty printed the Arduino code v	단 Open with GitHub Deskto	ор
🗋 .gitlab-ci.yml	Seeed:Arduino: fix travis.yml write		
🗅 .travis.yml	Seeed:Arduino: fix travis.yml write	Download ZIP	

• Step 2. Open Arduino IDE, navigate to Sketch > Include

Library > Add .ZIP Library... and open the downloaded

zip file



• Step 3. Navigate to File > Examples >

 $\texttt{DigitalPressureSensor} > \texttt{spi}_\texttt{background} \ to \ \texttt{open}$

the **spi_background** example

💿 sketch_may31a Arduino 1.8.19	(Windows Store 1.8.57.0)	
File Edit Sketch Tools Help		
New Ctrl+N		
Open Ctrl+O		
Open Recent >		
Sketchbook >		
Examples >	Built-in Examples	
Close Ctrl+W	01.Basics	`
Save Ctrl+S	02.Digital	here, to
Save As Ctrl+Shift+S	03.Analog	>
Page Setup Ctrl+Shift+P	04.Communication	>
Print Ctrl+P	05.Control	>
	06.Sensors	>
Preferences Ctrl+Comma	07.Display	>
Ouit Ctrl+O	08.Strings	>
VOTA TOOP()	09.USB	>
	10.StarterKit_BasicKit	>
// put yo	11.ArduinoISP	re, to r
	Examples for any board	
	Adafruit Circuit Playground	>
}	Bridge	>
J	Esplora	>
	Ethernet	>
	Firmata	>
	GSM	>
	LiquidCrystal	>
	Robot Control	>
	Robot Motor	>
	SD	>
	Servo	>
	SpacebrewYun	>
	Stepper	>
	Temboo	>
	RETIRED	>
	Examples for Arduino Lino	_
	EEPROM	>
	SoftwareSerial	>
	SPI	>
	Wire	>
		-
	Examples from Custom Libraries	
	DigitalPressureSensor	i2c_background
	Grove - Digital Light Sensor	> i2c_command
	Grove - Laser PM2.5 Sensor HM3301	> i2c_interrupt
	Grove - LCD RGB Backlight	spi_background
	Grove 4-Digit Display	spi_command
	Grove SHT31 Temp Humi Sensor	> spi_interrupt

Alternatively you can copy the code from below as well

#include <Dps310.h>
// Dps310 Opject
Dps310 Dps310PressureSensor = Dps310();

void setup() {

//pin number of your slave select line
//XMC2G0

int16_t pin_cs = SS;

//for XMC 1100 Bootkit & XMC4700 Relax Kit uncomment the following
line

//int16_t pin_cs = 10;

Serial. begin (9600);

while (!Serial);

//Call begin to initialize Dps310
//The parameter pin_nr is the number of the CS pin on your
Microcontroller

Dps310PressureSensor.begin(SPI, pin_cs);

//temperature measure rate (value from 0 to 7)
//2^temp mr temperature measurement results per second

intl6 t temp mr = 2;

//temperature oversampling rate (value from 0 to 7)
//2 temp_osr internal temperature measurements per result
//A higher value increases precision

int16_t temp_osr = 2;

//pressure measure rate (value from 0 to 7)
//2^prs_mr pressure measurement results per second

int16_t prs_mr = 2;

//pressure oversampling rate (value from 0 to 7)
//2^prs_osr internal pressure measurements per result
//A higher value increases precision

int16_t prs_osr = 2;

//startMeasureBothCont enables background mode

//temperature and pressure ar measured automatically

//High precision and hgh measure rates at the same time are not available.

//Consult Datasheet (or trial and error) for more information

int16_t ret = Dps310PressureSensor.startMeasureBothCont(temp_mr, temp_osr, prs_mr, prs_osr);

//Use one of the commented lines below instead to measure only temperature or pressure

//int16_t ret = Dps310PressureSensor.startMeasureTempCont(temp_mr, temp_osr);

//int16_t ret = Dps310PressureSensor.startMeasurePressureCont(prs_mr,
prs_osr);

```
if (ret != 0) {
```

```
Serial.print("Init FAILED! ret = ");
Serial.println(ret);
```

```
} else {
```

```
Serial.println("Init complete!");
}
void loop() {
uint8 t pressureCount = 20;
float pressure[pressureCount];
uint8 t temperatureCount = 20;
float temperature[temperatureCount];
//This function writes the results of continuous measurements to the
arrays given as parameters
//The parameters temperatureCount and pressureCount should hold the
sizes of the arrays temperature and pressure when the function is
called
//After the end of the function, temperatureCount and pressureCount
hold the numbers of values written to the arrays
//Note: The Dps310 cannot save more than 32 results. When its result
buffer is full, it won't save any new measurement results
int16 t ret = Dps310PressureSensor.getContResults(temperature,
temperatureCount, pressure, pressureCount);
if (ret != 0) {
Serial.println();
Serial.println();
Serial.print("FAIL! ret = ");
Serial.println(ret);
} else {
Serial.println();
Serial.println();
Serial.print(temperatureCount);
Serial.println(" temperature values found: ");
for (int16 t i = 0: i < temperatureCount: i++) {</pre>
Serial.print(temperature[i]);
Serial.println(" degrees of Celsius");
Serial.println();
Serial.print(pressureCount);
Serial.println(" pressure values found: ");
for (int16 t i = 0; i < pressureCount; i++) {
Serial.print(pressure[i]);
Serial.println(" Pascal");
}
}
//Wait some time, so that the Dps310 can refill its buffer
delay(10000);
```

• Step 4. Upload the codes and open the Serial Monitor

Note: Once you upload the codes, it will not be executed automatically until you click **Serial Monitor** on the upper right corner of the Arduino window.

	/dev/cu.usbmodem21401
[
14:26:45.430 -> 16 temperature values found:	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.71 degrees of Celsius	
14:26:45.430 -> 26.71 degrees of Celsius	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.73 degrees of Celsius	
14:26:45.430 -> 26.70 degrees of Celsius	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.71 degrees of Celsius	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.71 degrees of Celsius	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.72 degrees of Celsius	
14:26:45.430 -> 26.73 degrees of Celsius	
14:26:45.430 -> 26.71 degrees of Celsius	
14:26:45.430 -> 26.71 degrees of Celsius	
14:26:45.430 ->	
14:26:45.430 -> 16 pressure values found:	
14:26:45.430 -> 99844.84 Pascal	
14:26:45.430 -> 99844.92 Pascal	
14:26:45.430 -> 99844.00 Pascal	
14:26:45.430 -> 99844.27 Pascal	
14:26:45.430 -> 99844.16 Pascal	
14:26:45.430 -> 99845.08 Pascal	
14:26:45.430 -> 99844.75 Pascal	
14:26:45.430 -> 99843.55 Pascal	
14:26:45.430 -> 99844.14 Pascal	
14:26:45.430 -> 99843.22 Pascal	
14:26:45.430 -> 99844.31 Pascal	
14:26:45.430 -> 99844.49 Pascal	
14:26:45.430 -> 99844.09 Pascal	
14:26:45.430 -> 99845.19 Pascal	
14:26:45.430 -> 99844.47 Pascal	
14:26:45.430 -> 99844.07 Pascal	

🗹 Autoscroll 🗹 Show timestamp

Now you will see the temperature and pressure data displayed one after

the other on the serial monitor as above!

Federal Communication Commission (FCC) Radiation Exposure Statement

This equipment complies with FCC RF radiation exposure limits set forth for an uncontrolled environment. When using the product, maintain a distance of 20cm from the body to ensure compliance with RF exposure requirements.

This device complies with part 15 of the FCC rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation. NOTE: The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications or changes to this equipment. Such modifications or changes could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.

- Increase the separation between the equipment and receiver.

-Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

-Consult the dealer or an experienced radio/TV technician for help.

FCC Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

ORIGINAL EQUIPMENT MANUFACTURER (OEM) NOTES

The OEM must certify the final end product to comply with unintentional radiators (FCC Sections 15.107 and 15.109) before declaring compliance of the final product to Part 15 of the FCC rules and regulations. Integration into devices that are directly or indirectly connected to AC lines must add with Class II Permissive Change.

The OEM must comply with the FCC labeling requirements. If the module' s label is not visible when installed, then an additional permanent label must be applied on the outside of the finished product which states: "Contains transmitter module FCC ID: Z4T-XIAOESP32C3.

Additionally, the following statement should be included on the label and in the final product's user manual: "This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interferences, and (2) this device must accept any interference received, including interference that may cause

(2) this device must accept any interference received, including interference that may ca undesired operation."

The module is allowed to be installed in mobile and portable applications

A module or modules can only be used without additional authorizations if they have been tested and granted under the same intended end ⁻ use operational conditions, including simultaneous transmission operations. When they have not been tested and granted in this manner, additional testing and/or FCC application filing may be required. The most straightforward approach to address additional testing conditions is to have the grantee responsible for the certification of at least one of the modules submit a permissive change application. When having a module grantee file a permissive change is not practical or feasible, the following guidance provides some additional options for host manufacturers. Integrations using modules where additional testing and/or FCC application filing(s) may be required are: (A) a module used in devices requiring additional RF exposure compliance information (e.g., MPE evaluation or SAR testing); (B) limited and/or split modules not meeting all of the module requirements; and (C) simultaneous transmissions for independent collocated transmitters not previously granted together.

This Module is full modular approval, it is limited to OEM installation ONLY.

Integration into devices that are directly or indirectly connected to AC lines must add with Class II Permissive Change. (OEM) Integrator has to assure compliance of the entire end product include the integrated Module. Additional measurements (15B) and/or equipment authorizations (e.g. Verification) may need to be addressed depending on co-location or simultaneous transmission issues if applicable. (OEM) Integrator is reminded to assure that these installation instructions will not be made available to the end user