RAK11200 WisBlock WiFi Module Datasheet

Overview

Description

RAK11200 is a **WisBlock Core** module for RAK **WisBlock** based on Espressif ESP32-WROVER. It is a powerful, generic WiFi-BLE MCU module that targets a wide variety of applications. There are two CPU cores that can be individually controlled and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The low-power deep-sleep current consumption of the ESP32-WROVER is about 10 uA. This makes the **RAK11200** an ultra-low-power communication solution. **RAK11200** can be comfortably programmed with the Arduino™ IDE or PlatformIO.

Features

- Two low-power Xtensa® 32-bit LX6 microprocessors
- Up to 240 MHz CPU clock
- Built-in PCB antenna
- 4 MB External SPI Flash, 520 KB RAM
- 8 MB SPI Pseudo static RAM (PSRAM)
- WiFi 802.11 b/g/n (802.11n up to 150 Mbps)
- Bluetooth v4.2 BR/EDR and BLE specification
- Rich set of peripherals: RTC, UART, I2C, SPI, SD card interface
- low-power deep-sleep mode

Specifications

Overview

Board Overview

The RAK11200 WisBlock WiFi Module back view and front view (top) can be seen in Figure 1.

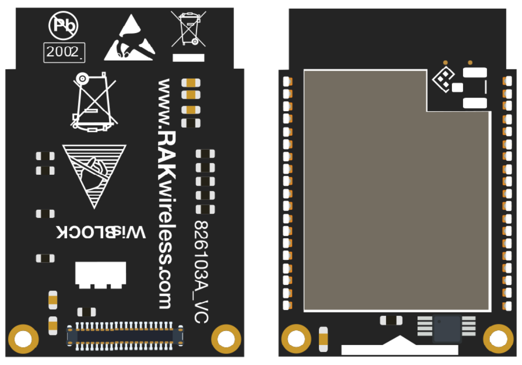


Figure 1: RAK11200 WiFi Module Overview

Mounting Sketch

Figure 2 shows RAK11200 module mounting sketch with the WisBase RAK5005-O board.

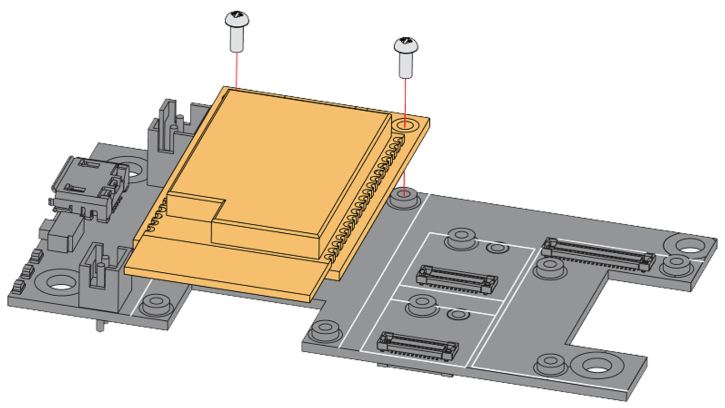


Figure 2: RAK11200 WiFi Module Mounting Sketch

Hardware

The hardware specification is categorized into four parts. It discuses the interfacing of the module and its corresponding functions and diagrams. It also covers the electrical and mechanical parameters that include the tabular data of the functionalities and standard values of the RAK11200 WisBlock WiFi Module.

⚠️ **WARNING**

- Different from other ESP32 boards, the RAK11200 needs to be put ***manually*** into **download mode**. If you do not force the RAK11200 into **download mode**, you cannot upload your sketch from Arduino IDE (or PlatformIO) to the board.
- To force the RAK11200 into **download mode**, you need to connect the pin ***BOOT0*** on the WisBlock Base RAK5005-O to ***GND*** and push the reset button.
- The ***BOOT0*** pin is on the J10 pin header, the ***GND*** pin is next to it.

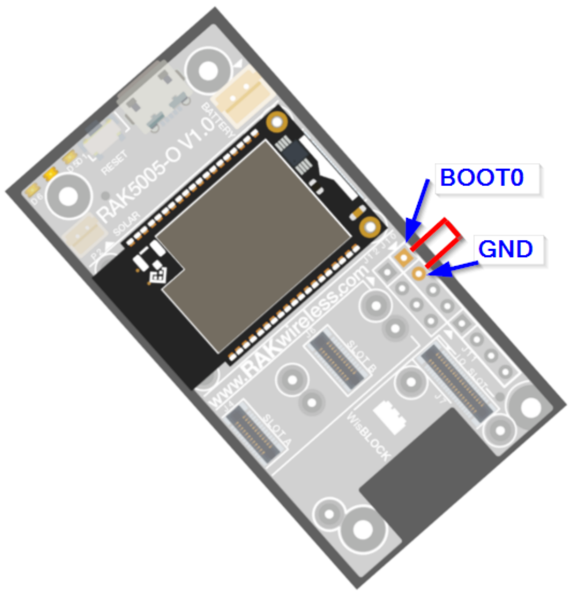


Figure 3: Force ESP32 Download mode

Interfaces

UART Interface

The RAK11200 module provides two UART interfaces: UART0 and UART1. The UART0 can be used for firmware upgrades or to access console output through the WisBlock baseboard USB interface. The UART1 is the main communication interface with WisIO or WisSensor modules.

UART0 Programming Port

To support USB, the RAK11200 has a USB-to-UART converter onboard to connect the ESP32's UART0 to the USB connector. Figure 4 shows the RAK11200 module UART programming circuit.

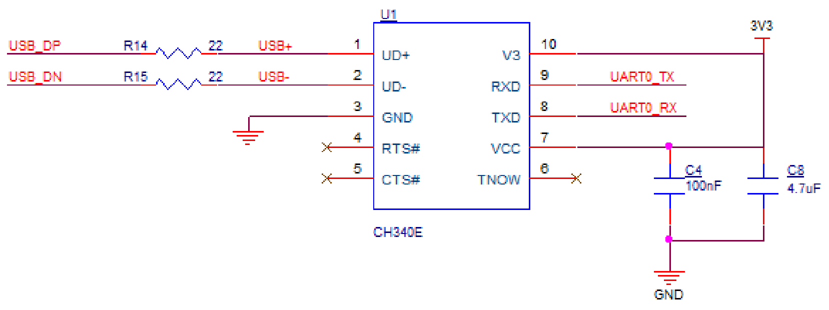


Figure 4: RAK11200 USB to UART schematic

SPI Interface

The RAK11200 supports one single SPI Interface in full-duplex or half-duplex communication modes. The SPI interface supports the following features:

- Both master and slave modes;
- Configurable SPI frequency;
- Four SPI transfer modes, which is defined by the polarity (CPOL) and the phase (CPHA) of the SPI clock;
- An internal FIFO buffer of 64-byte.

I2C Interface

The RAK11200 module provides two I2C bus interfaces. The module allows you to access directly the registers to control I2C interfaces, which adds more flexibility to the design of the final product. Depending on your configuration, it can serve as an I2C master mode. The I2C interface supports:

- Standard mode (100 Kbit/s) and Fast mode (400 Kbit/s);
- Up to 5 MHz, constrained by the SDA pull-up strength;
- 7-bit/10-bit addressing mode.

Pin Definition

The RAK11200 module has an ESP32-WROVER module at its core. Figure 5 shows the core module pins and connection information.

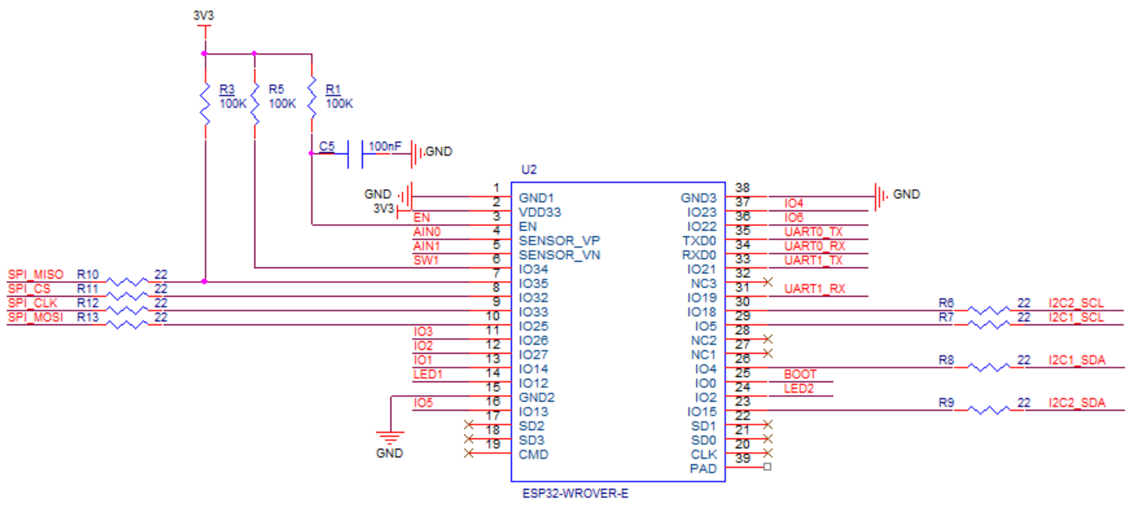


Figure 5: RAK11200 Core module pin connection

WisBlock Core RAK11200 Pin Assignment

Pin number WisBlock	Function	Pin name	Pin number ESP32
1	VBAT	VBAT	--
2	VBAT	VBAT	--
3	GND	GND	1, 15, 38
4	GND	GND	1, 15, 38
5	3V3	3V3	2
6	3V3	3V3	2
7	USB_DP	USB_DP	--
8	USB_DN	USB_DN	--
9	NC	NC	--
10	SW1	GPIO34	6
11	UART0_TX	GPIO1	35
12	UART0_RX	GPIO3	34
13	EN	EN	3
14	LED1	GPIO12	14
15	LED2	GPIO2	24
16	NC	NC	--
17	3V3	3V3	2
18	3V3	3V3	2
19	I2C1_SDA	GPIO4	26
20	I2C1_SCL	GPIO5	29
21	AIN0	GPIO36	4
22	AIN1	GPIO39	5
23	BOOT	GPIO0	25
24	NC	NC	--

Pin number WisBlock	Function	Pin name	Pin number ESP32
25	SPI_CS	GPIO32	8
26	SPI_CLK	GPIO33	9
27	SPI_MISO	GPIO35	7
28	SPI_MOSI	GPIO25	10
29	IO1	GPIO14	13
30	IO2	GPIO27	12
31	IO3	GPIO26	11
32	IO4	GPIO23	37
33	UART1_TX	GPIO21	33
34	UART1_RX	GPIO19	31
35	I2C2_SDA	GPIO15	23
36	I2C2_SCL	GPIO18	30
37	IO5	GPIO13	16
38	IO6	GPIO22	36
39	GND	GND	1, 15, 38
40	GND	GND	1, 15, 38

RF Specifications

BLE Radio

Receiver

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @30.8% PER	-	-94	-93	-92	dBm
Maximum received signal @30.8% PER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dBm
Intermodulation	-	-36	-	-	dBm

Transmitter

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	0	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	-12	-	+9	dBm
Drift rate	-	-	0.7	-	kHz/50us
Drift	-	-	2	-	kHz

WiFi Radio

Parameter	Condition	Min	Typ	Max	Unit
Operating frequency range	-	2412	-	2484	MHz
TX power	11b mode	17.5	18.5	20	dBm
TX power	11n MCS7	12	13	14	dBm
Sensitivity	11b, 1 Mbps	-	-97	-	dBm

Electrical Characteristics

Absolute Maximum Ratings

Symbol	Description	Min.	Typical	Max.	Unit
V_{BAT}	Power supply for the module	0.5	-	4.2	V
V_{DD}	Power supply for ESP32 module	2.3	3.3	3.6	V
I_{out}	Step down IC output current	-	-	700	mA

Recommended Operating Conditions

Symbol	Description	Min.	Typical	Max.	Unit
V_{BAT}	Power supply for the module	3.1	-	4.2	V
V_{DD}	Power supply for ESP32 module	3.0	3.3	3.6	V
T_{OPR}	Operation Temperature	-40	-	85	℃

Mechanical Characteristics

Board Dimensions

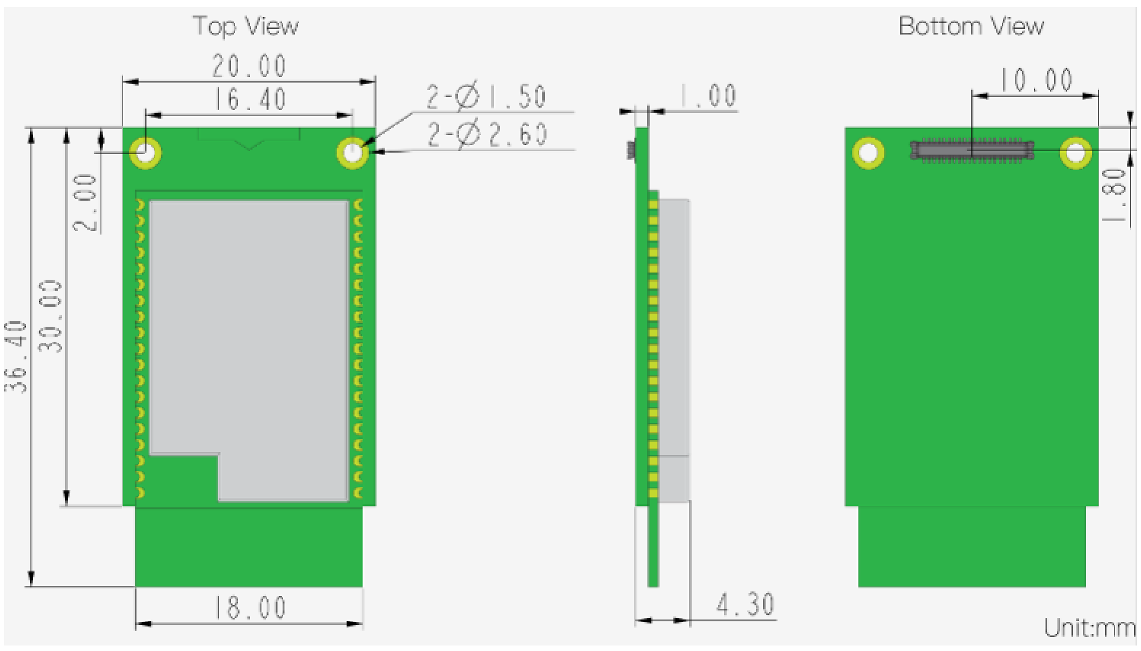
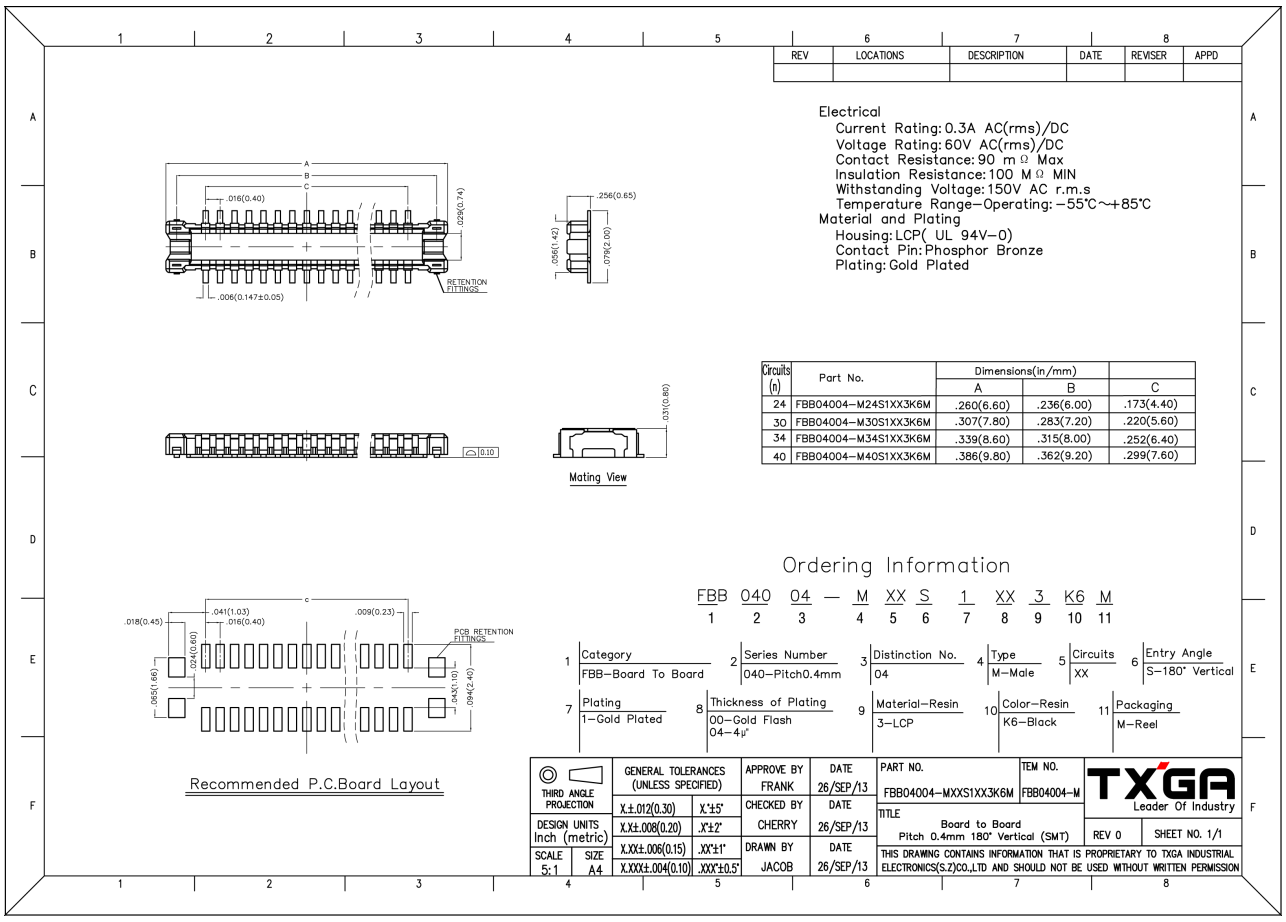


Figure 6: RAK11200 Board Dimensions

WisConnector PCB Layout



REV	LOCATIONS	DESCRIPTION	DATE	REVISER	APPD

Electrical
- Current Rating: 0.3A AC(rms)/DC
- Voltage Rating: 60V AC(rms)/DC
- Contact Resistance: 90 mΩ Max
- Insulation Resistance: 100 MΩ MIN
- Withstanding Voltage: 150V AC r.m.s
- Temperature Range—Operating: −55℃~+85℃

Material and Plating
- Housing: LCP(UL 94V—0)
- Contact Pin: Phosphor Bronze
- Plating: Gold Plated

Circuits (n)	Part No.	Dimensions(in/mm)		
		A	B	C
24	FBB04004—M24S1XX3K6M	.260(6.60)	.236(6.00)	.173(4.40)
30	FBB04004—M30S1XX3K6M	.307(7.80)	.283(7.20)	.220(5.60)
34	FBB04004—M34S1XX3K6M	.339(8.60)	.315(8.00)	.252(6.40)
40	FBB04004—M40S1XX3K6M	.386(9.80)	.362(9.20)	.299(7.60)

Ordering Information

FBB 040 04 — M XX S 1 XX 3 K6 M
1 2 3 4 5 6 7 8 9 10 11

1	Category	FBB—Board To Board
2	Series Number	040—Pitch0.4mm
3	Distinction No.	04
4	Type	M—Male
5	Circuits	XX
6	Entry Angle	S—180° Vertical
7	Plating	1—Gold Plated
8	Thickness of Plating	00—Gold Flash 04—4μ"
9	Material—Resin	3—LCP
10	Color—Resin	K6—Black
11	Packaging	M—Reel

THIRD ANGLE PROJECTION	GENERAL TOLERANCES (UNLESS SPECIFIED)		APPROVE BY FRANK	DATE 26/SEP/13	PART NO. FBB04004—MXXS1XX3K6M	TEM NO. FBB04004—M
DESIGN UNITS Inch (metric)	X.±.012(0.30)	X.°±5°	CHECKED BY CHERRY	DATE 26/SEP/13	TITLE Board to Board Pitch 0.4mm 180° Vertical (SMT)	TXGA Leader Of Industry
	X.X±.008(0.20)	.X°±2°				REV 0 SHEET NO. 1/1
SCALE 5:1 SIZE A4	X.XX±.006(0.15)	.XX°±1°	DRAWN BY JACOB	DATE 26/SEP/13	THIS DRAWING CONTAINS INFORMATION THAT IS PROPRIETARY TO TXGA INDUSTRIAL ELECTRONICS(S.Z)CO.,LTD AND SHOULD NOT BE USED WITHOUT WRITTEN PERMISSION	
	X.XXX±.004(0.10)	.XXX°±0.5°				

Figure 7: WisConnector PCB footprint and recommendations

Schematic Diagram

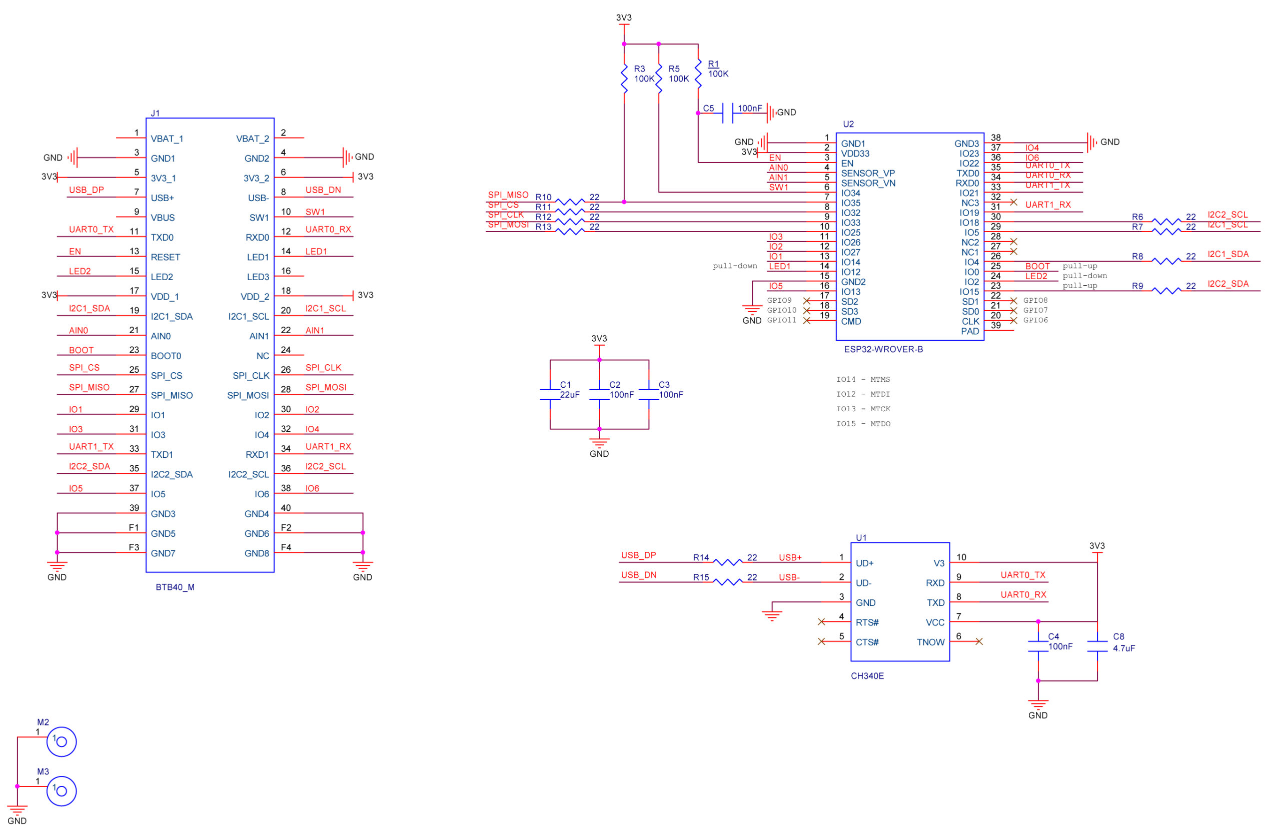


Figure 8: RAK11200 Schematic Diagram

RAK11200 is a **WisBlock Core** module for RAK **WisBlock** based on Espressif ESP32-WROVER. It is a powerful, generic WiFi-BLE MCU module that targets a wide variety of applications. There are two CPU cores that can be individually controlled and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The low-power deep-sleep current consumption of the ESP32-WROVER is about 10 uA. This makes the **RAK11200** an ultra-low-power communication solution. **RAK11200** can be comfortably programmed with the Arduino™ IDE or PlatformIO.

Last Updated: 11/4/2021, 5:44:19 AM

RAK11200 Quick Start Guide

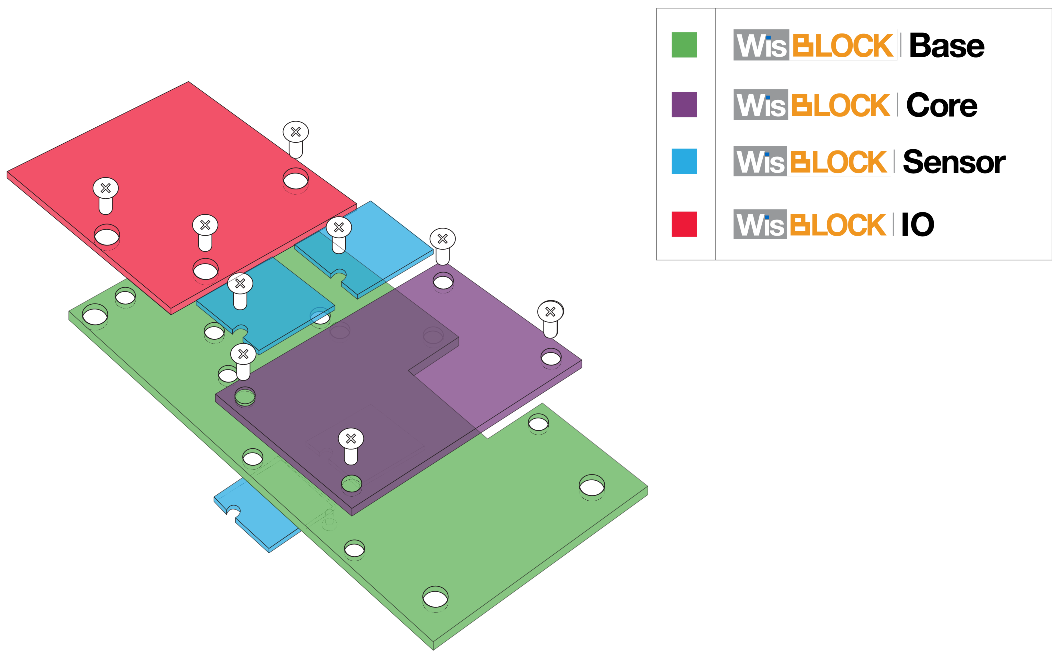


Figure 1: WisBlock-Assembly

Content

- Introduction
- Safety information
- Hardware Setup
- Arduino IDE BSP Installation
- PlatformIO Installation

Introduction

WisBlock is an amazing product built by RAK company for the IoT industry. It can build circuits like building blocks quickly to realize your idea, and through high-speed connectors and fasteners interconnection, it can directly compose reliable industrial products.

WisBlock consists of WisBlock Base, WisBlock Core, WisBlock Sensor, and WisBlock IO.

RAK5005-O is the WisBlock Base board which can be connected with WisBlock Core and WisBlock IO through the connector of the board and provides direct data bus interconnection. WisBlock Base module also integrates the power supply circuit to realize low power battery power supply. In order to facilitate users, WisBlock Base has reserved USB ports, indicator lights, keys, and extended IO interfaces.

RAK11200 is the WisBlock Core board which consists of ESP32 WROVER. It supports WiFi and BLE functions, and supply a rich resource MCU so that you can program it if you want.

WisBlock is not only a functional test capable product in the product development verification stage but also industrial products oriented to mass production. It uses a high-speed connector to ensure the integrity of the signal. At the same time, it is equipped with a fastening screw, which can be used in a vibration environment. And WisBlock can be used reliably in various civil and industrial scenarios through rigorous reliability tests.

WisBlock uses a compact stacked hardware design, which integrates various computing, connecting, and sensor circuits in the size of 60*30 mm. The compact size makes it easy for users to build in various customized housings to achieve complete products. RAK also has a series of housings for WisBlock modules, which can meet the requirements of various protection levels.

Safety Information

Read the following items carefully so that WisBlock can be used safely.

Hardware

- Use WisBlock according to its hardware specification, including the power supply, the temperature of use, the battery, and so on.
- Don't submerge WisBlock in liquids, and don't place WisBlock where water can reach.
- Don't power WisBlock using other power sources which RAK hasn't suggested.
- Some WisBlock modules require higher current that can't be provided by USB port alone. In this case, it is recommended to connect a battery to WisBlock Base Board.

⚠️ **WARNING**

- Battery can cause harm if not handled properly.
- Only 3.7-4.2 V Rechargeable LiPo batteries are supported. It is highly recommended not to use other types of batteries with the system unless you know what you are doing.
- If a non-rechargeable battery is used, it has to be unplugged first before connecting the USB cable to the USB port of the board to configure the device. Not doing so might damage the battery or cause a fire.x
- Make sure the battery wires match the polarity on the RAK WisBlock Base Board. Not all batteries have the same wiring.
- Only 5 V solar panels are supported. Do not use 12 V solar panels. It will destroy the charging unit and eventually other electronic parts.

- There is already a bootloader in every WisBlock core board MCU when you receive the device so that you don't need to flash the bootloader again. Normally, you only need to use it directly or upload new code into it through Arduino IDE. If you accidentally erase the bootloader, contact RAK forum.

- Don't unplug any hardware connector when you are uploading code into it, otherwise WisBlock may become unresponsive.

Hardware Setup

The RAK5005-O board offers several GPIO's on solder pads or the WisBlock Sensor or WisBlock IO modules. These GPIO's are named IO1 to IO6 and SW1. These GPIO's are connected to GPIO's of the RAK11200 module.

The GPIO assignments are defined in the RAK11200 variant.h file of the Arduino BSP.

RAK5005-O GPIO mapping to RAK11200 GPIO ports

- RAK5005-O <-> ESP32
- IO1 <-> Arduino GPIO number 14
- IO2 <-> Arduino GPIO number 27
- IO3 <-> Arduino GPIO number 26
- IO4 <-> Arduino GPIO number 23
- IO5 <-> Arduino GPIO number 13

- IO6 <-> Arduino GPIO number 22
- SW1 <-> Arduino GPIO number 34
- A0 <-> Arduino GPIO number 36
- A1 <-> Arduino GPIO number 39
- SPI_CS <-> Arduino GPIO number 32
- LED1 <-> Arduino GPIO number 12
- LED2 <-> Arduino GPIO number 2

Defined names from variant.h

```
#define WB_IO1 14
#define WB_IO2 27
#define WB_IO3 26
#define WB_IO4 23
#define WB_IO5 13
#define WB_IO6 22
#define WB_SW1 34
#define WB_A0 36
#define WB_A1 39
#define WB_CS 32
#define WB_LED1 12
#define WB_LED2 2
```

Software Setup

Getting started with RAK11200 is simple and straightforward. The first thing you need is to set up your software development environment. We have made detailed tutorials on how to set up Arduino™ IDE and the PlatformIO extension to be ready to use the WisBlock 11200.

Arduino IDE BSP Installation

Install RAKWireless ESP32 BSP on Arduino Boards Manager

1. To add board support for RAK11200 on Arduino, start Arduino IDE and open the Preferences window (**File** > **Preferences**).

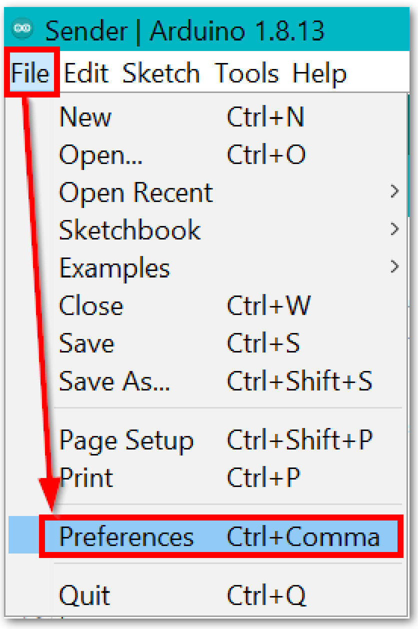
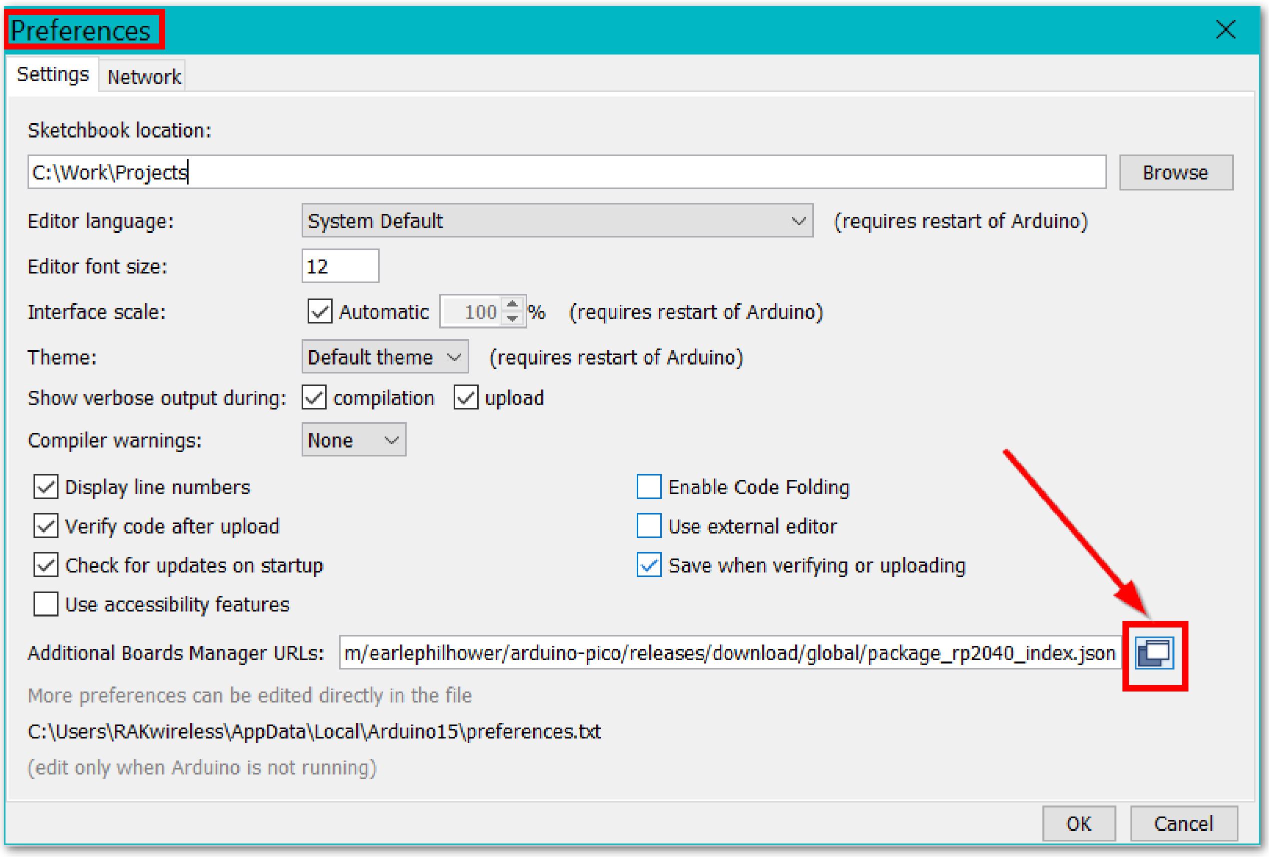


Figure 2: Arduino File Preferences Window

2. In the **Preferences** window, look for **Additional Boards Manager URLs** and click the icon on the right side.



Preferences

Settings | Network

Sketchbook location:
C:\Work\Projects

Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: ✓ Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ✓ compilation ✓ upload
Compiler warnings: None

✓ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
✓ Save when verifying or uploading

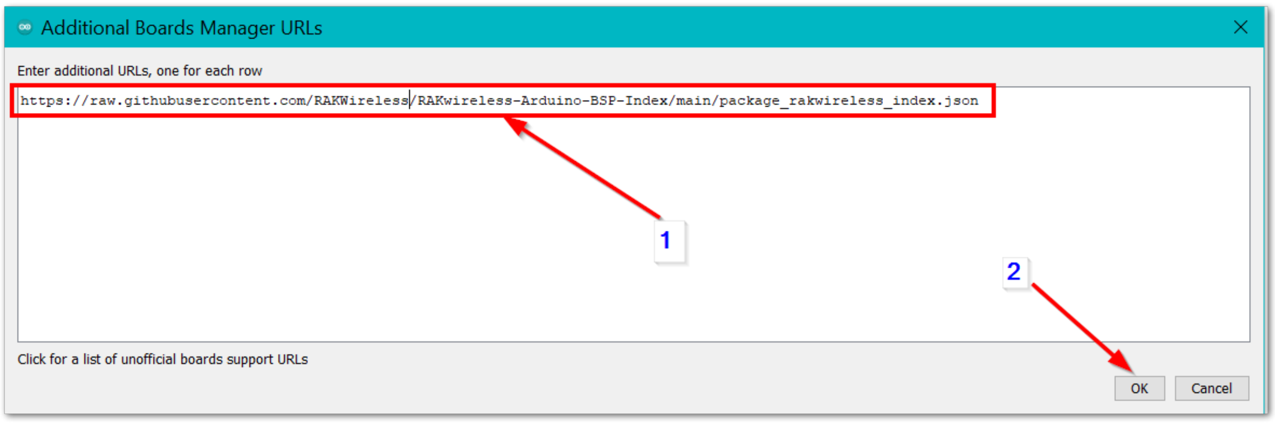
Additional Boards Manager URLs: m/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

More preferences can be edited directly in the file
C:\Users\RAKwireless\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Figure 3: Arduino Preferences

3. Copy `https://raw.githubusercontent.com/RAKwireless/RAKwireless-Arduino-BSP-Index/main/package_rakwireless_index.json` and paste it into the new window.

- If there is already an URL from another manufacturer in that field, paste the above URL into a new line. Then press the **OK** button.



Additional Boards Manager URLs

Enter additional URLs, one for each row

https://raw.githubusercontent.com/RAKWireless/RAKwireless-Arduino-BSP-Index/main/package_rakwireless_index.json

Click for a list of unofficial boards support URLs

Figure 4: Arduino Additional Boards Manager URLs

4. Next, open the **Boards Manager** in the menu **Tools**.

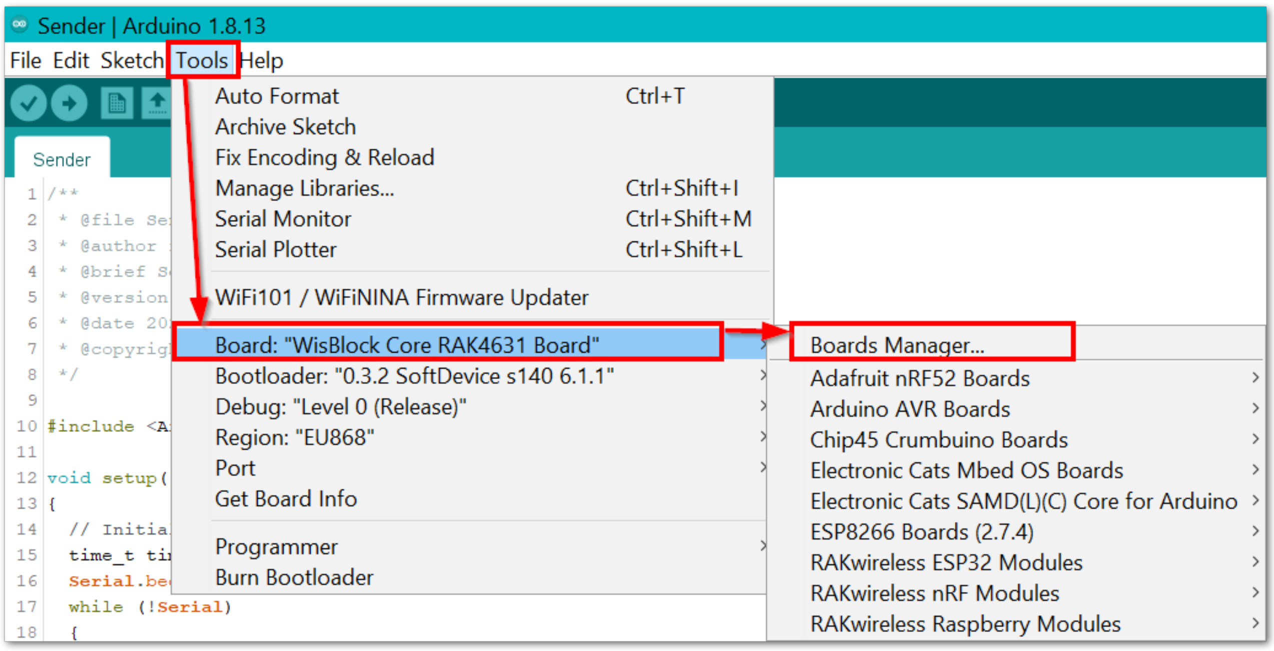
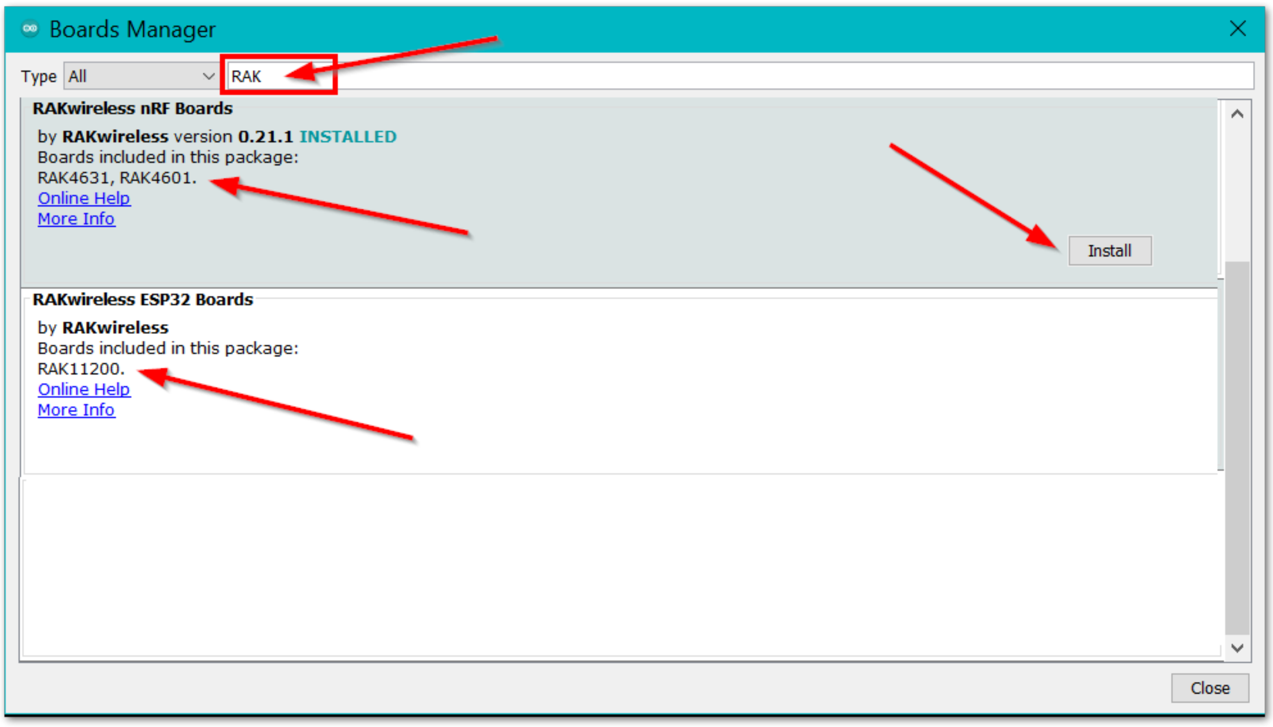

Sender | Arduino 1.8.13
File Edit Sketch Tools Help
Sender
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "WisBlock Core RAK4631 Board"
Bootloader: "0.3.2 SoftDevice s140 6.1.1"
Debug: "Level 0 (Release)"
Region: "EU868"
Port
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Adafruit nRF52 Boards
Arduino AVR Boards
Chip45 Crumbuino Boards
Electronic Cats Mbed OS Boards
Electronic Cats SAMD(L)(C) Core for Arduino
ESP8266 Boards (2.7.4)
RAKwireless ESP32 Modules
RAKwireless nRF Modules
RAKwireless Raspberry Modules

Figure 5: Arduino Boards Manager

5. Type **RAK** in the search bar. The RAKwireless WisBlock Core modules will be shown in the window.



RAKwireless nRF Boards
by **RAKwireless** version **0.21.1** **INSTALLED**
Boards included in this package:
RAK4631, RAK4601.
Online Help
More Info

RAKwireless ESP32 Boards
by **RAKwireless**
Boards included in this package:
RAK11200.
Online Help
More Info

Figure 6: Arduino Tools Boards Manager

6. Select RAKwireless ESP32 Boards and click on **Install** button.

Depending on your connection speed, the installation can take some time. Just be patient.

Compiling a Project

1. The compiling process is very easy, what you need to do is just to click the Verify/Compile button on Arduino IDE.

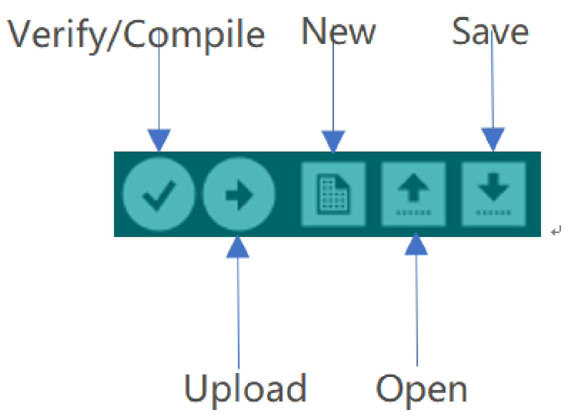
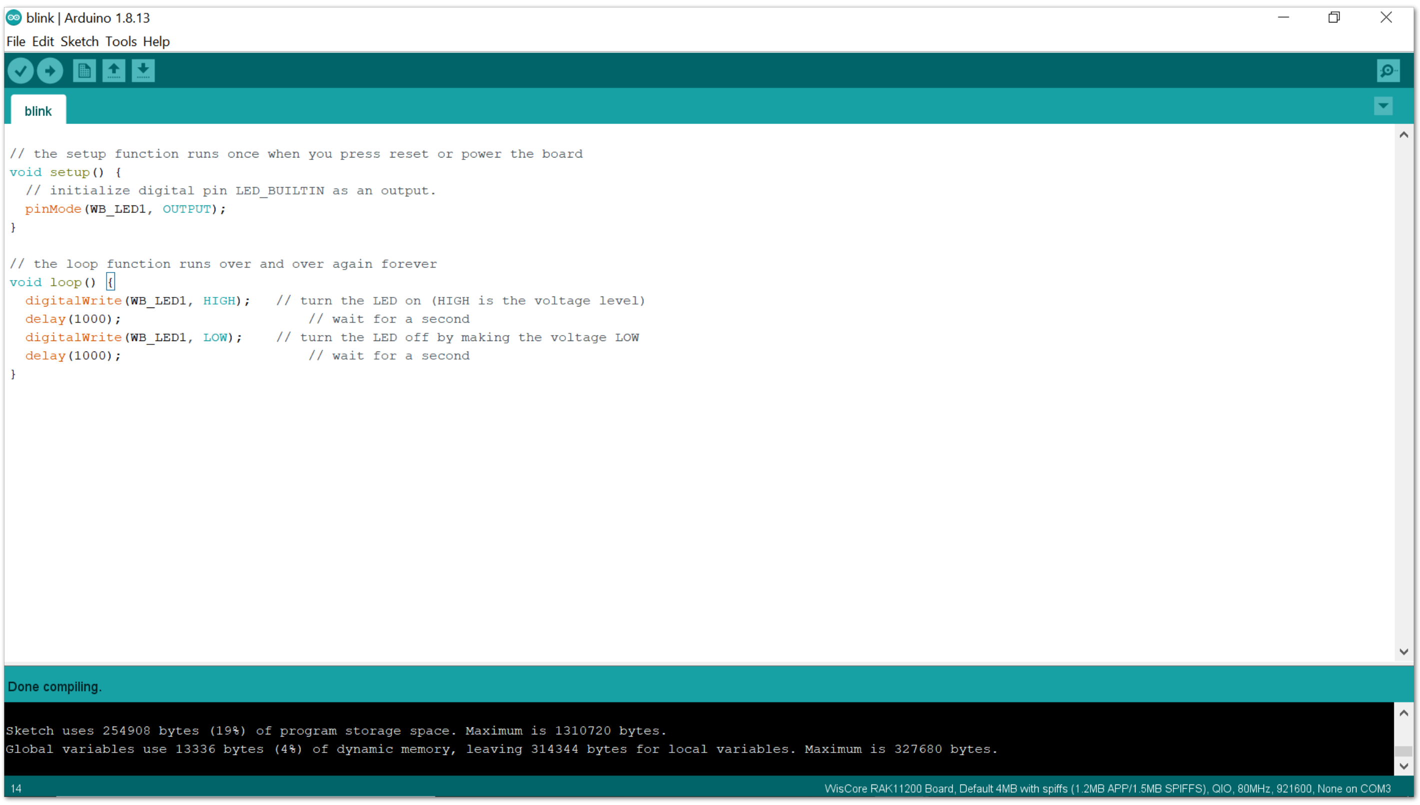


Figure 7: Arduino Verify/Compile

2. After compiling successfully, you can see some information in the output message area, and the state is "Done compiling.":



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(WB_LED1, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(WB_LED1, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(WB_LED1, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done compiling.

```
Sketch uses 254908 bytes (19%) of program storage space. Maximum is 1310720 bytes.
Global variables use 13336 bytes (4%) of dynamic memory, leaving 314344 bytes for local variables. Maximum is 327680 bytes.
```

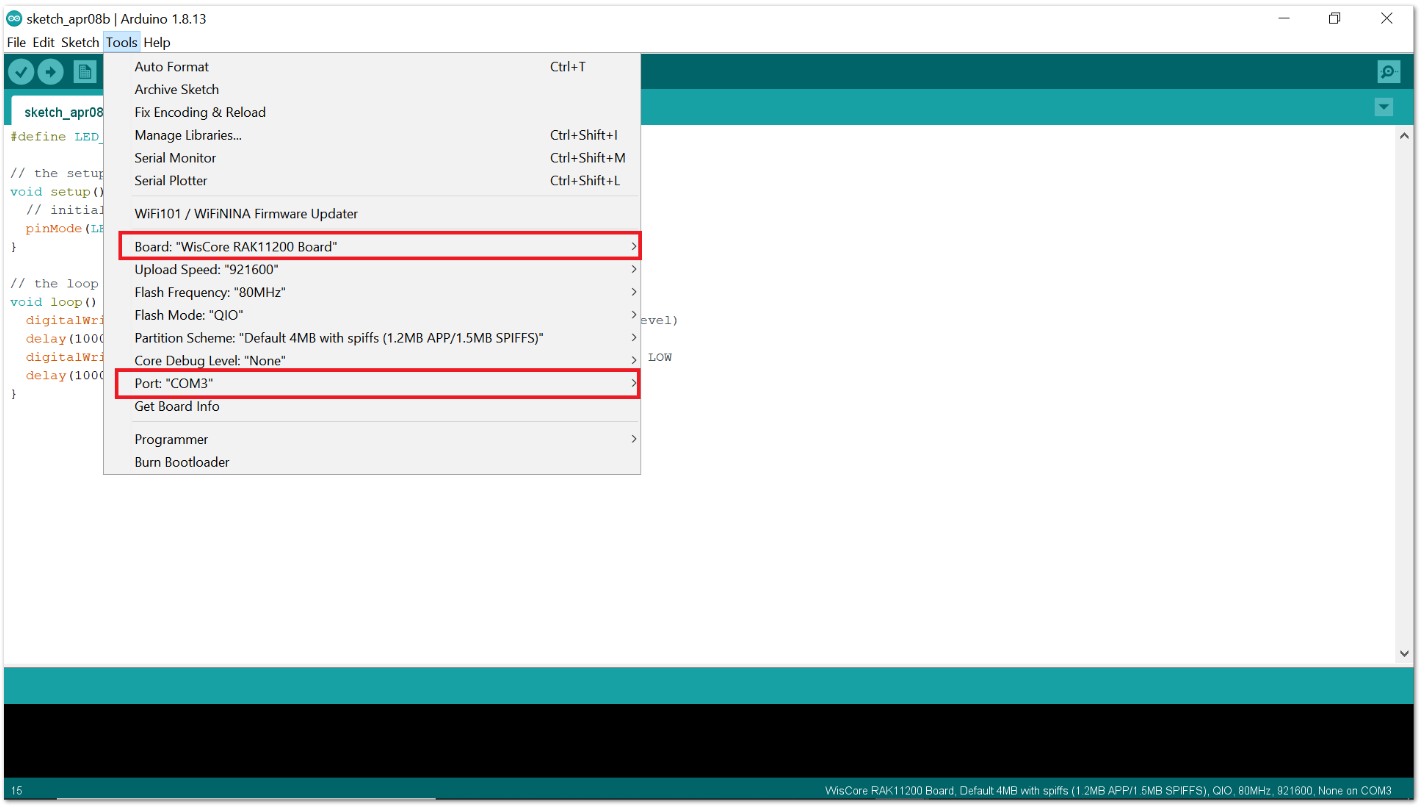
14   WisCore RAK11200 Board, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on COM3

Figure 8: Arduino Done compiling

Now, you can connect your WisBlock hardware with your PC, and upload the code into it.

Uploading to WisBlock

Make sure that your WisBlock hardware has been connected with your PC correctly, and your PC has recognized WisBlock hardware successfully. If it is, you can select the board and port now, as shown in Figure 9:


sketch_apr08b | Arduino 1.8.13
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Programmer
Burn Bootloader
Get Board Info

Board: "WisCore RAK11200 Board"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
Port: "COM3"

WisCore RAK11200 Board, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on COM3

Figure 9: Arduino Tools Configuration

1. Before uploading your sketch, short circuit BOOT0 and GND pin and press the reset button. Then click the Upload button using the configuration below.

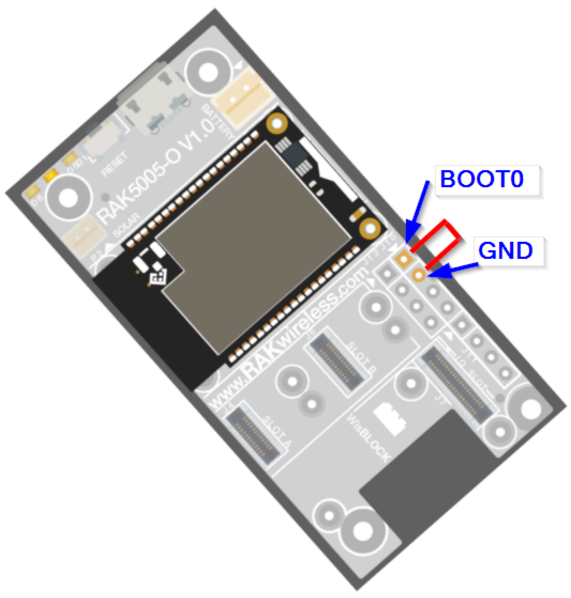
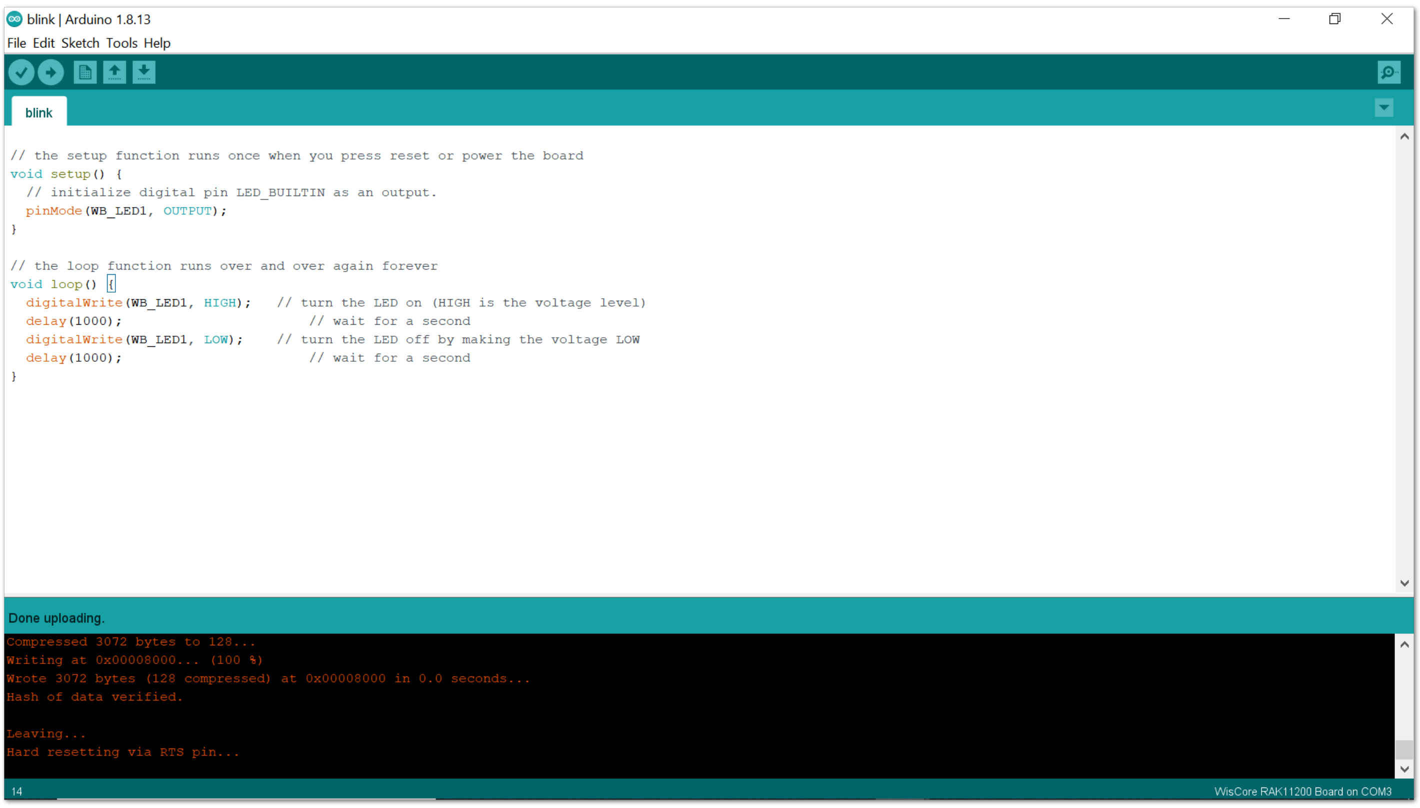


Figure 10: Force ESP32 Download mode

```
Board:"Wiscore RAK11200 board"
Upload Speed:"921600"
Flash Frequency:"80MHZ"
Flash Mode:"QIO"
Partition Scheme:"Default 4MB with spiffs(1.2MB APP/1.5MB SPIFFS)"
Core Debug Level:"No"
```

2. After uploading successfully, push the reset button. Then you can see some information, as shown in Figure 11 in the output message area. That means you've uploaded the code into RAK11200 successfully.



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(WB_LED1, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(WB_LED1, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(WB_LED1, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done uploading.

```
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 11: Arduino Done uploading

📝 **NOTE**

In case of upload error, the **Upload Speed** must be reduced.

Library Management on Arduino

In the Arduino IDE, the Library Manager and the libraries installed are available for every Arduino sketch.

Reserved GPIO Pins

It is not recommended to use the reserved GPIO pins. Some GPIO's cannot be used freely, as they are already assigned to module peripherals or have special functions during booting.

Bootstraping Pins

- GPIO0 pin is used as a bootstrapping pin and should be low to enter UART download mode. Make sure it is not pulled low by a peripheral device during boot or the firmware will not start.
- GPIO2 pin is used as a bootstrapping pin, and should be low to enter UART download mode. Make sure it is not pulled high by a peripheral device during boot or you will not be able to flash a firmware to the module.
- GPIO12 is used as a bootstrapping pin to select the output voltage of an internal regulator which powers the flash chip (VDD_SDIO). This pin has an internal pulldown so if left unconnected it will read low at reset (selecting default 3.3 V operation). Make sure it is not pulled high by a peripheral device during boot or the module might not be able to start.
- GPIO15 can be used to stop debug output on Serial during boot. If pulled low, there will be no output on the Serial port during the boot process. This can be helpful in battery-powered applications where you do not want to use the Serial port at all to reduce power consumption.

SPI Flash Integrated Pins

GPIO06 to GPIO11 are connected to the integrated SPI flash on the ESP32-WROVER chip and are not recommended for other uses.

Input Only Pins

GPIO34-39 can only be set as input mode and do not have software pullup or pulldown functions.

PSRAM Pins

GPIO16 and 17 are used for the RAK11200 Pseudo static RAM (PSRAM).

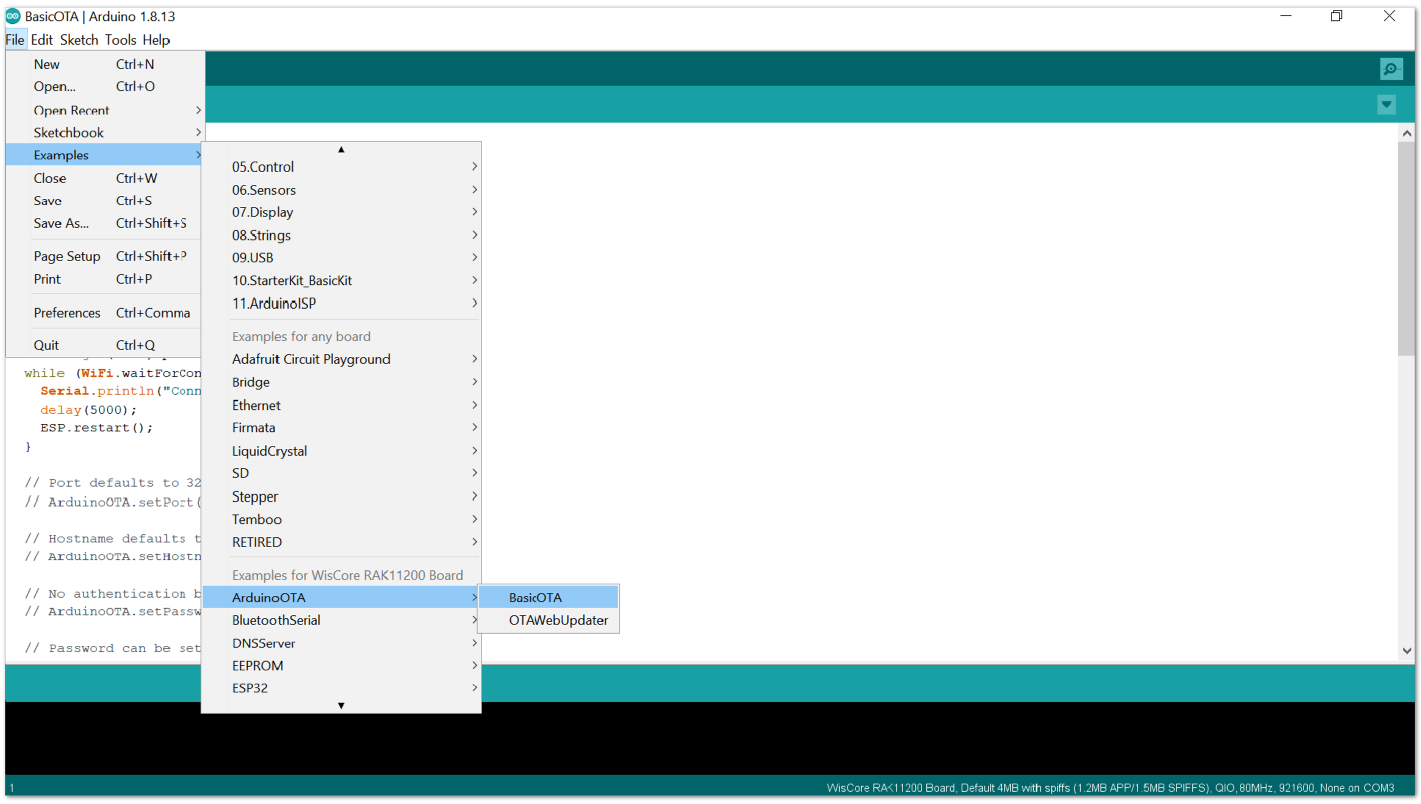
ESP32 Basic Over The Air (OTA)

OTA stands for Over-The-Air. This feature allows uploading a new program to RAK11200 using WiFi instead of requiring the user to connect the RAK11200 to a computer via USB to perform the update. See the detailed instructions on the link below.

- ESP32 Basic OTA in Arduino IDE

If you already installed the RAKwireless ESP32, then the BasicOTA sketch has also been installed.

1. Open the Arduino IDE -> File -> Examples-> ArduinoOTA-> BasicOTA.



```
while (WiFi.waitForCon
  Serial.println("Conn
  delay(5000);
  ESP.restart();
}

// Port defaults to 32
// ArduinoOTA.setPort(

// Hostname defaults t
// ArduinoOTA.setHostn

// No authentication b
// ArduinoOTA.setPassw

// Password can be set
```

WisCore RAK11200 Board, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on COM3

Figure 12: Arduino OTA Sketch

2. Modify the following two variables with your network credentials, so that RAK11200 can establish a WiFi connection with the existing network. Then save and upload the BasicOTA sketch.

```
const char* ssid = "..........";
const char* password = "..........";
```

3. Now, upload a new sketch using over the air port.

4. Copy the blink LED sketch below to your Arduino IDE.

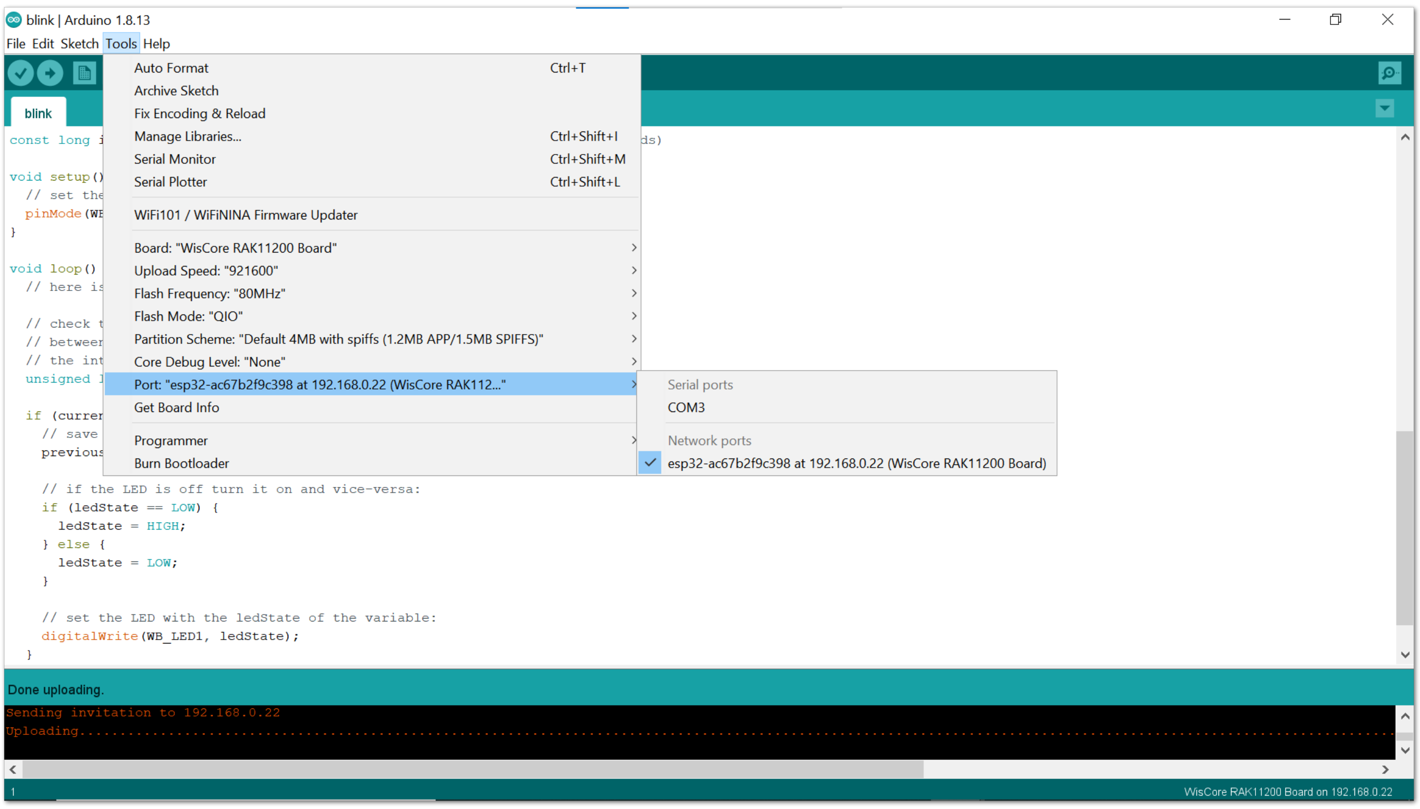
```
unsigned long previousMillis = 0;  // will store last time LED was updated
const long interval = 1000;  // interval at which to blink (milliseconds)
int ledState = LOW;  // ledState used to set the LED

void setup()
{
  pinMode(WB_LED1, OUTPUT);
}
void loop()
{

  //loop to blink without delay
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    // if the LED is off turn it on and vice-versa:
    ledState = not(ledState);
    // set the LED with the ledState of the variable:
    digitalWrite(WB_LED1,  ledState);
  }
}
```

The delay() for blinking builtin LED is not used because RAK11200 ESP32-WROVER pauses your program during the delay() routine. If the next OTA request is generated while the program is paused waiting for the delay() to pass, the OTA request will be lost.

5. Open Arduino IDE and select Tools->Port->esp32-xxxx, as shown in Figure 13.


blink | Arduino 1.8.13
File Edit Sketch Tools Help
blink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM3
Network ports

Board: "WisCore RAK11200 Board"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
Port: "esp32-ac67b2f9c398 at 192.168.0.22 (WisCore RAK112..."
✓ esp32-ac67b2f9c398 at 192.168.0.22 (WisCore RAK11200 Board)

```
const long i                                                     ds)

void setup()
  // set the
  pinMode(WE
}

void loop()
  // here is

  // check t
  // betweer
  // the int
  unsigned l

  if (currer
    // save
    previous

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(WB_LED1, ledState);
  }
```

```
Done uploading.
Sending invitation to 192.168.0.22
Uploading.........................................................................................
```

1   WisCore RAK11200 Board on 192.168.0.22

Figure 13: Arduino OTA WiFi Port

6. Finally, click on Upload button.

Within a few seconds, the new sketch will be uploaded using over the air port.

ESP32 Deep Sleep

To achieve maximum power saving during deep sleep, it is necessary to switch off Bluetooth and WiFi before calling esp_deep_sleep_start().

```
#include <esp_wifi.h>
#include <esp_bt.h>
...
  esp_wifi_stop();
  esp_bt_controller_disable();
  esp_deep_sleep_start();
```

ESP32 Wiki Content

- Arduino core for ESP32 Wiki content

Installation of BSP in PlatformIO

Install PlatformIO

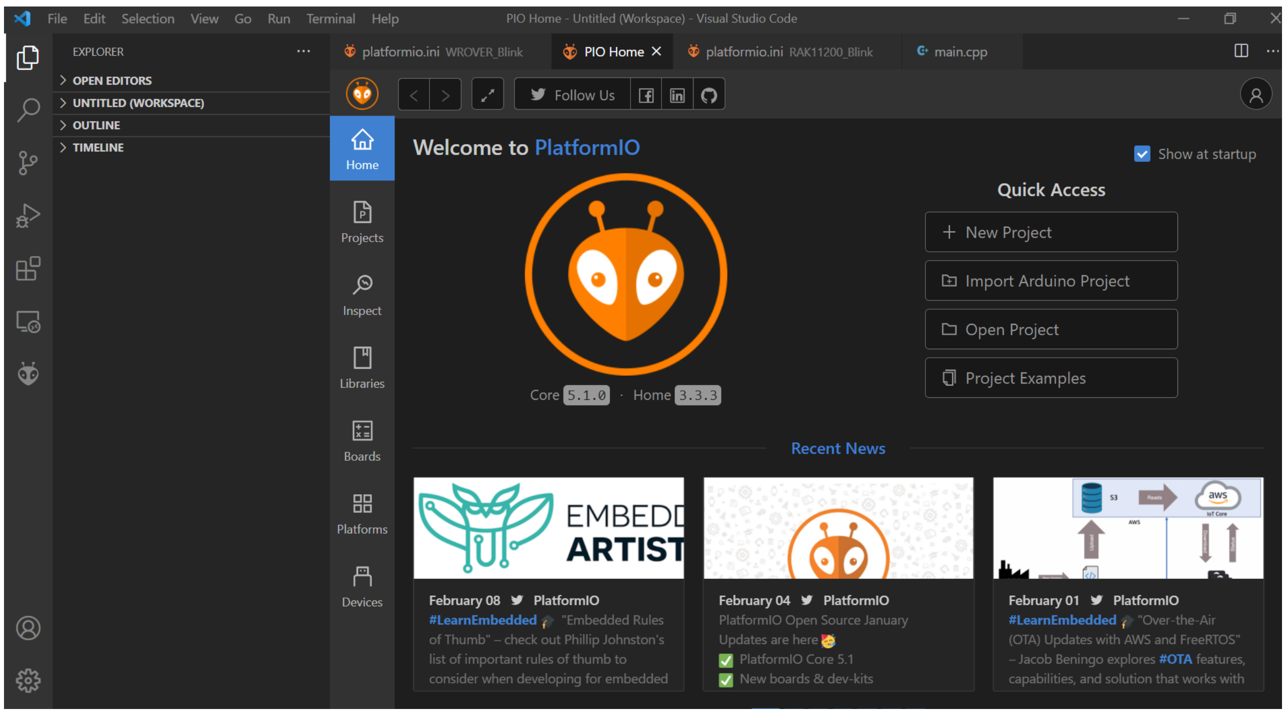
Download and install the Visual Studio Code which is a great and open source tool.

- Visual Studio Code

After installing the Visual Studio Code, you can search for PlatformIO and install it in the Extensions item.

Install Espressif 32 Arduino Framework

1. After installing PlatformIO, you can see the PlatformIO icon and click open.



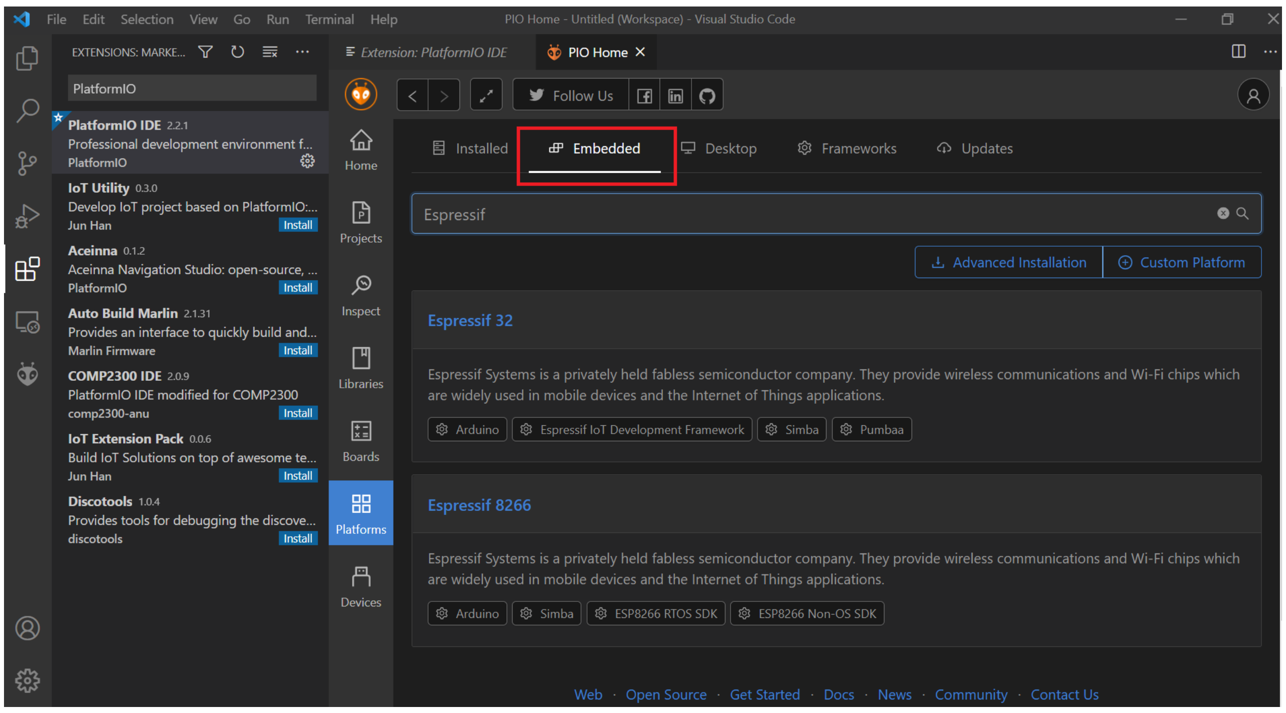
February 08 PlatformIO
#LearnEmbedded "Embedded Rules of Thumb" – check out Phillip Johnston's list of important rules of thumb to consider when developing for embedded

February 04 PlatformIO
PlatformIO Open Source January Updates are here
✓ PlatformIO Core 5.1
✓ New boards & dev-kits

February 01 PlatformIO
#LearnEmbedded "Over-the-Air (OTA) Updates with AWS and FreeRTOS" – Jacob Beningo explores #OTA features, capabilities, and solution that works with

Figure 14: Visual Studio Code PlatformIO extension

2. Open "Platforms" in PlatformIO and search for "Espressif" on Embedded tab.



Espressif 32

Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Arduino · Espressif IoT Development Framework · Simba · Pumbaa

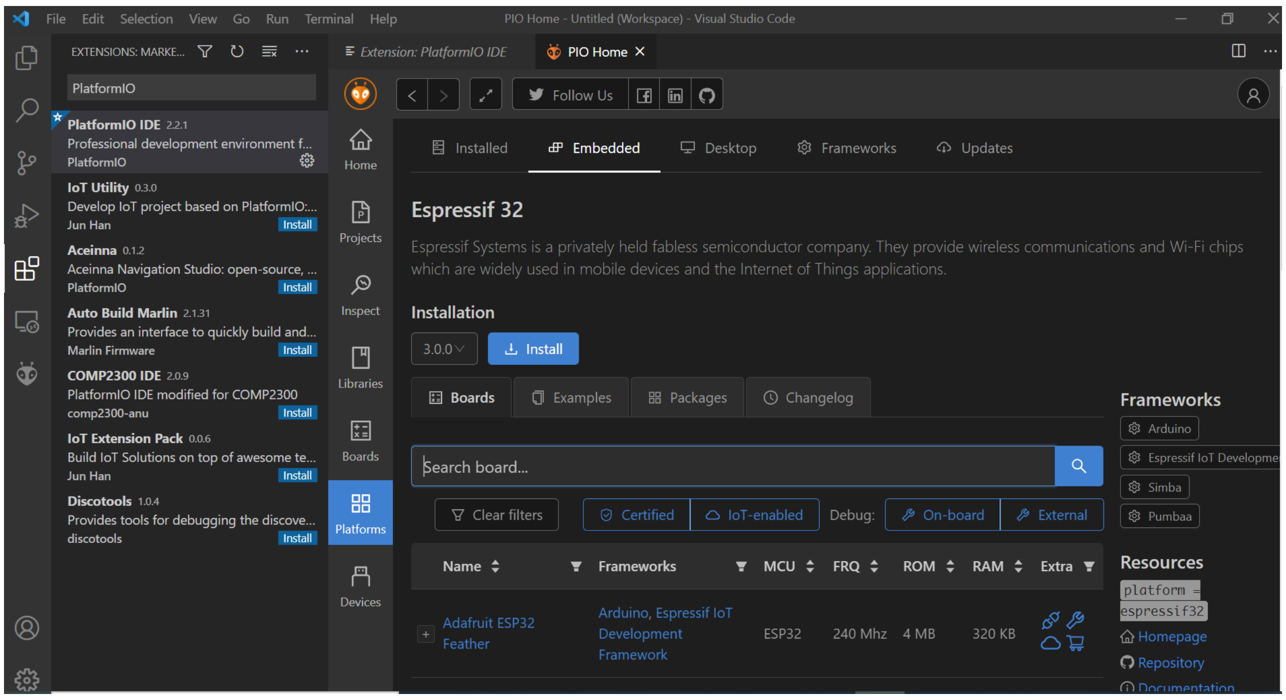
Espressif 8266

Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Arduino · Simba · ESP8266 RTOS SDK · ESP8266 Non-OS SDK

Figure 15: Espressif Platform

3. You can see there are several items, just click "Espressif 32" item and then "Install".



Espressif 32

Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Name	Frameworks	MCU	FRQ	ROM	RAM	Extra
Adafruit ESP32 Feather	Arduino, Espressif IoT Development Framework	ESP32	240 Mhz	4 MB	320 KB	

Figure 16: Espressif Framework

4. Before running the first RAK11200 project on the PlatformIO, you need to ensure that the framework-arduinoespressif32 is installed. Then import a minimal project named **arduino-blink**. On PIO Home, click on "Project Examples".

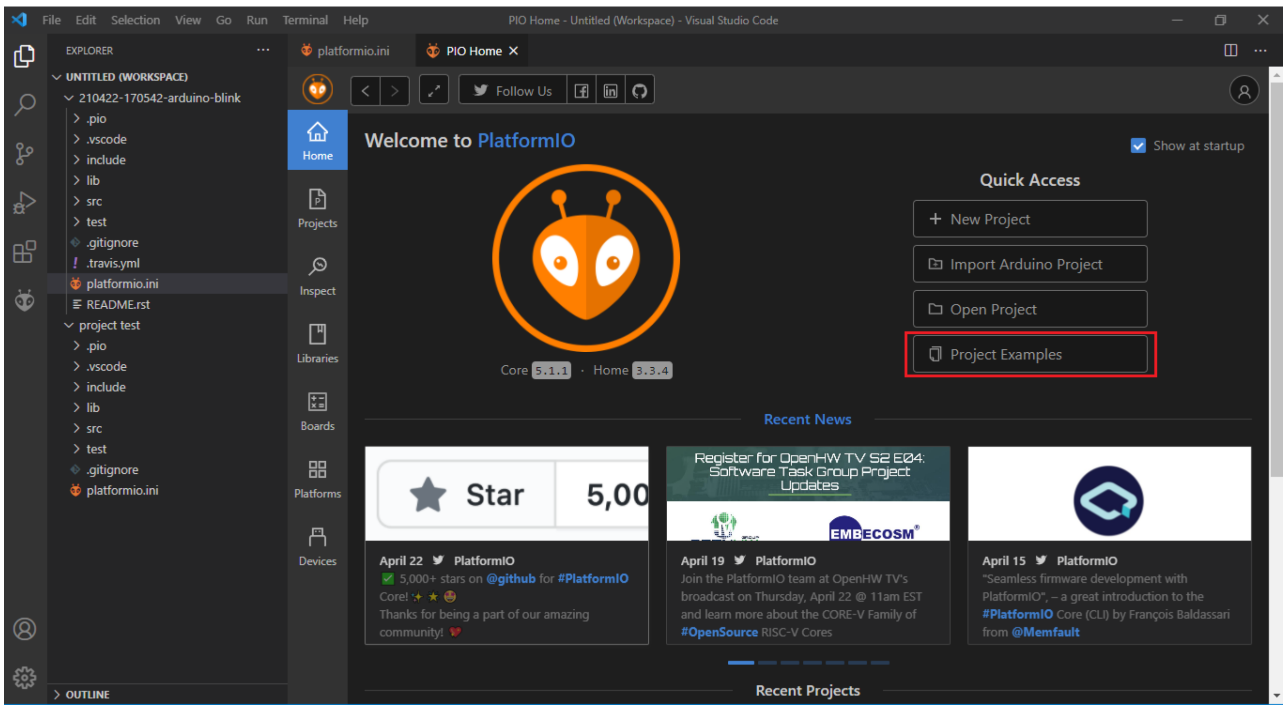
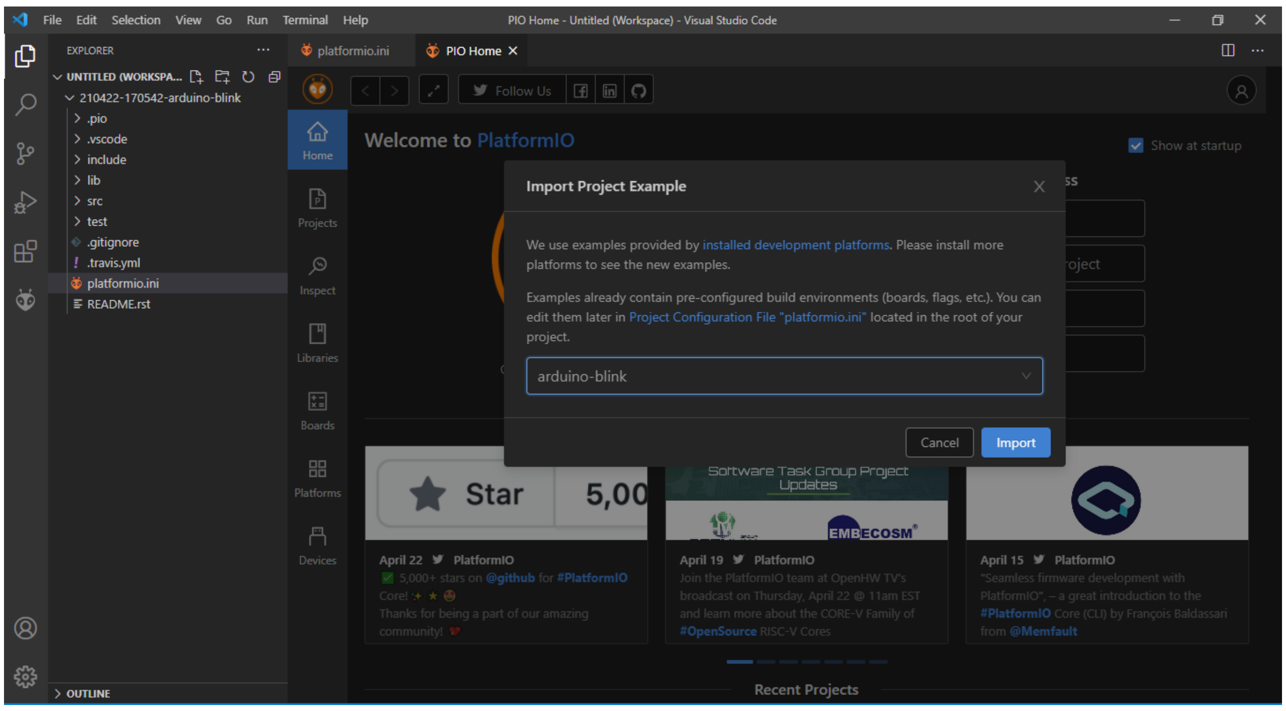


Figure 17: PIO Project Examples

5. On "Import Project Example" window type **arduino-blink** and then click on Import button.



We use examples provided by installed development platforms. Please install more platforms to see the new examples.

Examples already contain pre-configured build environments (boards, flags, etc.). You can edit them later in Project Configuration File "platformio.ini" located in the root of your project.

arduino-blink

Figure 18: PIO arduino-blink

Add WisBlock Core RAK11200 to the Platform

1. Clone WisBlock repository .

```
git clone https://github.com/RAKWireless/WisBlock.git
```

2. Copy the file **wiscore_rak11200.json** located on folder **<cloned_dir>\WisBlock\PlatformIO\RAK11200** to espressif32 platform folder.

- The platform folder path is similar to the following:
  - Windows: **%userprofile%.platformio\platforms\espressif32\boards**
  - Linux (Ubuntu): **$HOME/.platformio/platforms/espressif32/boards**
  - Mac OS: **/Users/{Your_User_id}/.platformio/platforms/espressif32/boards**

Add WisBlock Core RAK11200 to the Framework

Copy the folder **WisCore_RAK11200_Board** located on **<cloned_dir>\WisBlock\PlatformIO\RAK11200** to the variants folder inside the espressif32 package folder.

- The espressif32 package folder path is similar to the following:
    - Windows: **%userprofile%.platformio\packages\framework-arduinoespressif32\variants**
    - Linux (Ubuntu): **$HOME/.platformio/packages/framework-arduinoespressif32/variants**
    - Mac OS: **/Users/{Your_User_id}/.platformio/packages/framework-arduinoespressif32**

Library Management on PlatformIO

The PlatformIO libraries are managed on a per-project basis. You install a library for a specific project and not for the entire IDE. Thus it is possible to have the same library working with different versions in two different projects.

Last Updated: 1/10/2022, 2:07:11 AM