ESP32 개발환경과 GPS Application

care-lab.kr

강사 소개

- 안양 국제유통단지에 본사 캐어랩 2021년 설립 https://care-lab.kr/
- 마라톤 풀코스를 38번 완주한 마라토너로 전자 공학, 코딩과 하드웨어 교육에 진심인 임베디드 개발자.
- IT, 메이커와 달리기에 관한 글 11,480(15년)개를 발행한 블로그를 운영하고 있다.
- https://fishpoint.tistory.com/
- 현재 캐어랩 대표로 아두이노, 라즈베리파이, 통신, ESP32, STM32를 활용한 프로젝트와 디지털 제품 개발, 마케팅, 출판에 관한 콘텐츠를 개발하고 있다.

- LG정밀 방산 연구소
- Ebooknet, 아이보호닷컴 창업
- 휴인스 수석연구원
- 한양대학교 지능형로봇사업단
- 한양대 ERICA 어드벤처디자인 강의
- 현 디지털 창작집단 대표
- 현 메이커블 출판사 대표
- 현 과천 정보과학도서관 과학육성실무협의회

- 저서: 당장 써먹는 AI 프롬프트 사전

@kimbongzo

@kimbongzo

@kimbongzo

@kimbongzo

@kimbongzo

자료와 소스 필요할 때

- 상세 튜토리얼은 블로그를 참고하세요
- https://fishpoint.tistory.com/

- 예제 코드는 이곳을 참고하세요.
- https://github.com/kimbongzo

- 기술 문서는 이곳에서 내려받으세요.(Raspberry pi, Arduino, ESP32, OpenCV)
- https://kimbongzo.gumroad.com/

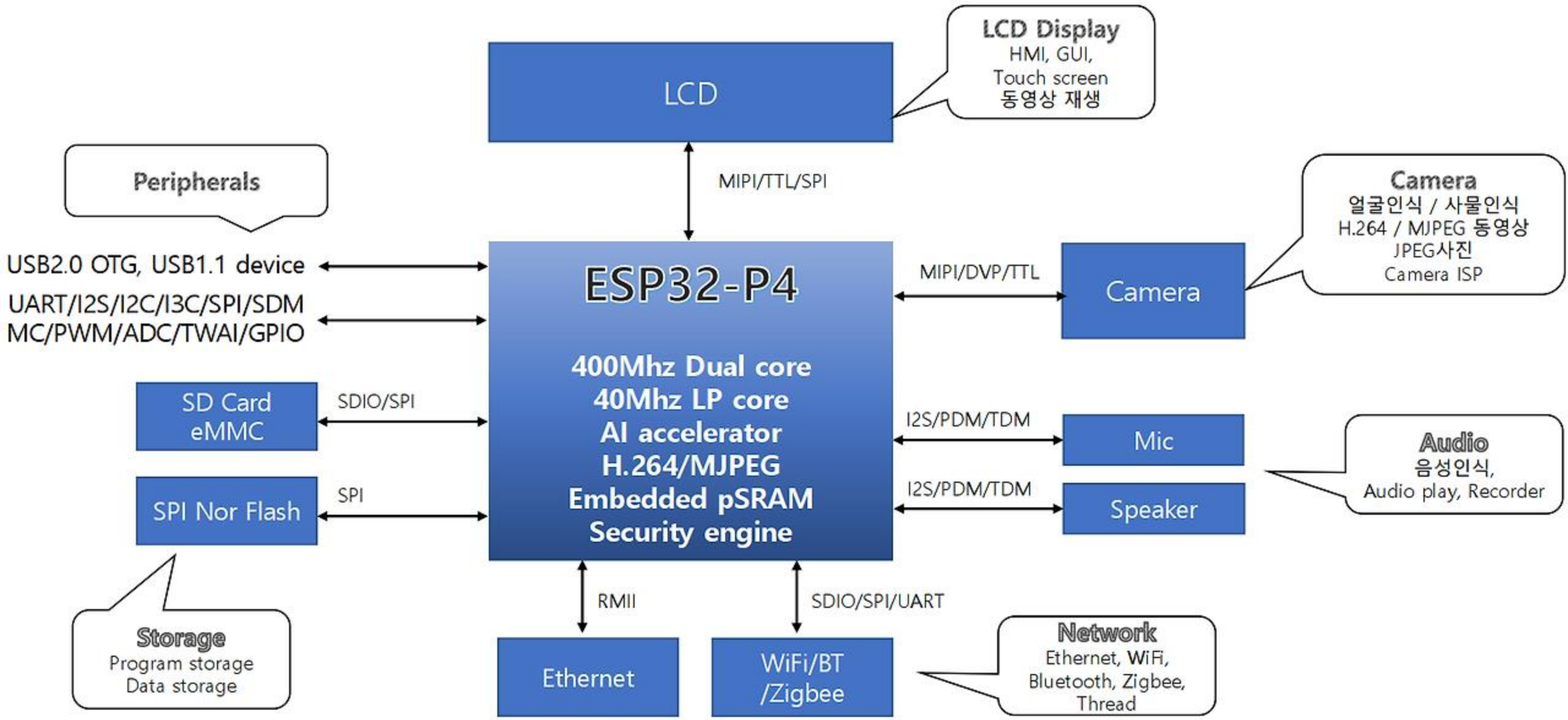
- 질문과 기술 문의는 kimbongzo@gmail.com

ESP32 History

- 상하이에 본사를 둔 에스프레시프 시스템즈(Espressif Systems) 는 싱가포르 기업가이자 엔지니어인 테오 스위 앤(Teo Swee Ann)이 2008년에 설립
- 2013년에 출시된 2.4GHz Wi-Fi 시스템온칩(SoC)인 ESP8089
- 2014년 ESP8266, Wi-Fi 및 블루투스 모뎀 + (Tensilica) Xtensa L106 32비트 마이크로컨트롤러
- 2016년 ESP32 출시

- CEO 메시지 https://www.espressif.com/en/company/about-us/ceo-letter
- 감사하다는 고객 https://www.reddit.com/r/esp32/comments/a2pnk9/i_want_to_say_thank_you_to_espressif_for_doing/
- 에스프레시프 시스템즈 - 꼭 알아야 할 10가지 중요한 사항 https://dotcommagazine.com/2023/06/espressif-systems-top-ten-most-important-things-you-need-to-know/
- NUS 공과대생에서 억만장자로 https://vulcanpost.com/769625/singapore-billionaire-teo-swee-ann-espressif-tech/

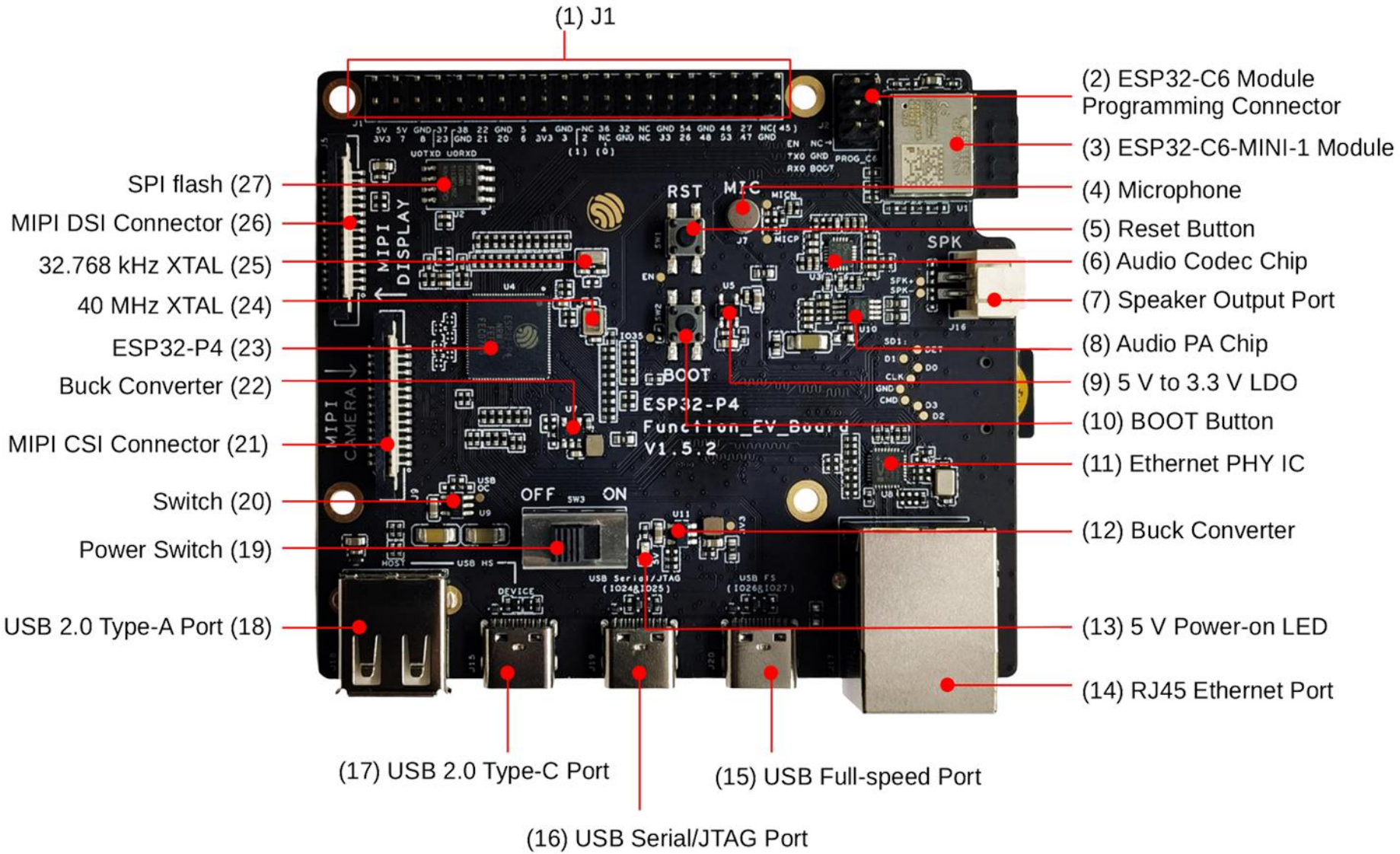
ESP32-P4 Introduction



https://espressif.co.kr/25

ESP32-P4-Function-EV-Board

Datasheet https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32p4/esp-dev-kits-en-master-esp32p4.pdf/
하드웨어 설계 https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32p4/schematic-checklist-esp32p4.html

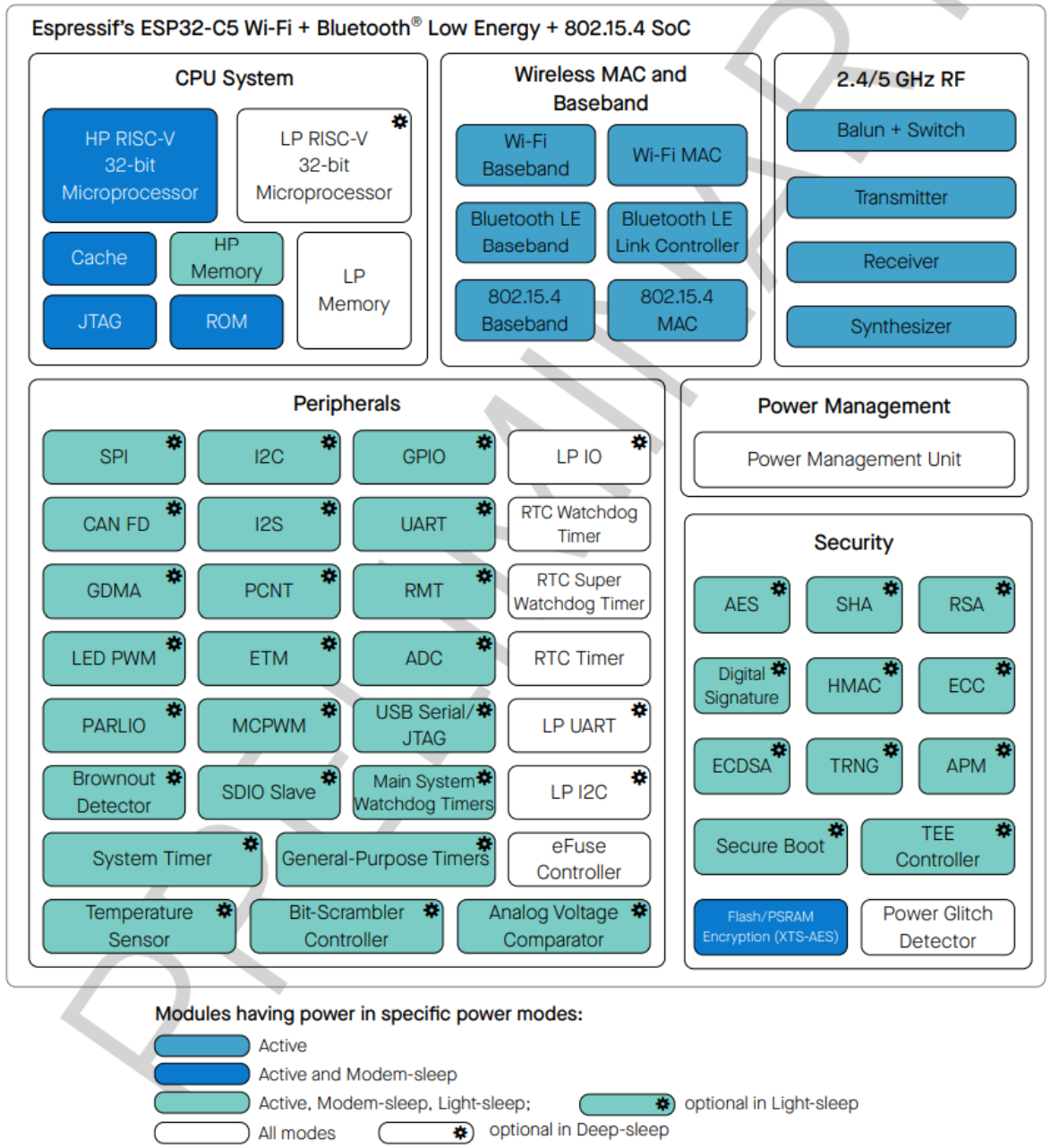


ESP32-C5 SoC 듀얼밴드 WiFi+BLE+Zigbee

- 프로세서: 32비트 RISC-V 싱글 코어, HP core 240 MHz + LP core 48Mhz
- 무 선 연 결 : 2.4Ghz + 5Ghz Wi-Fi 6 (802.11ax), Bluetooth 5/6 (LE), Zigbee3.0, Thread 1.4
- 플래시/RAM: 외부 SPI 플래시 및 PSRAM 지원
- GPIO: 최대 22개의 GPIO
- 보안: 보안 부트, 플래시 암호화, 디지털 서명, HMAC 등
- 패키지: QFN48 (6mm x 6mm)
- 전원 관리: 저전력 설계, 다양한 슬립 모드 지원

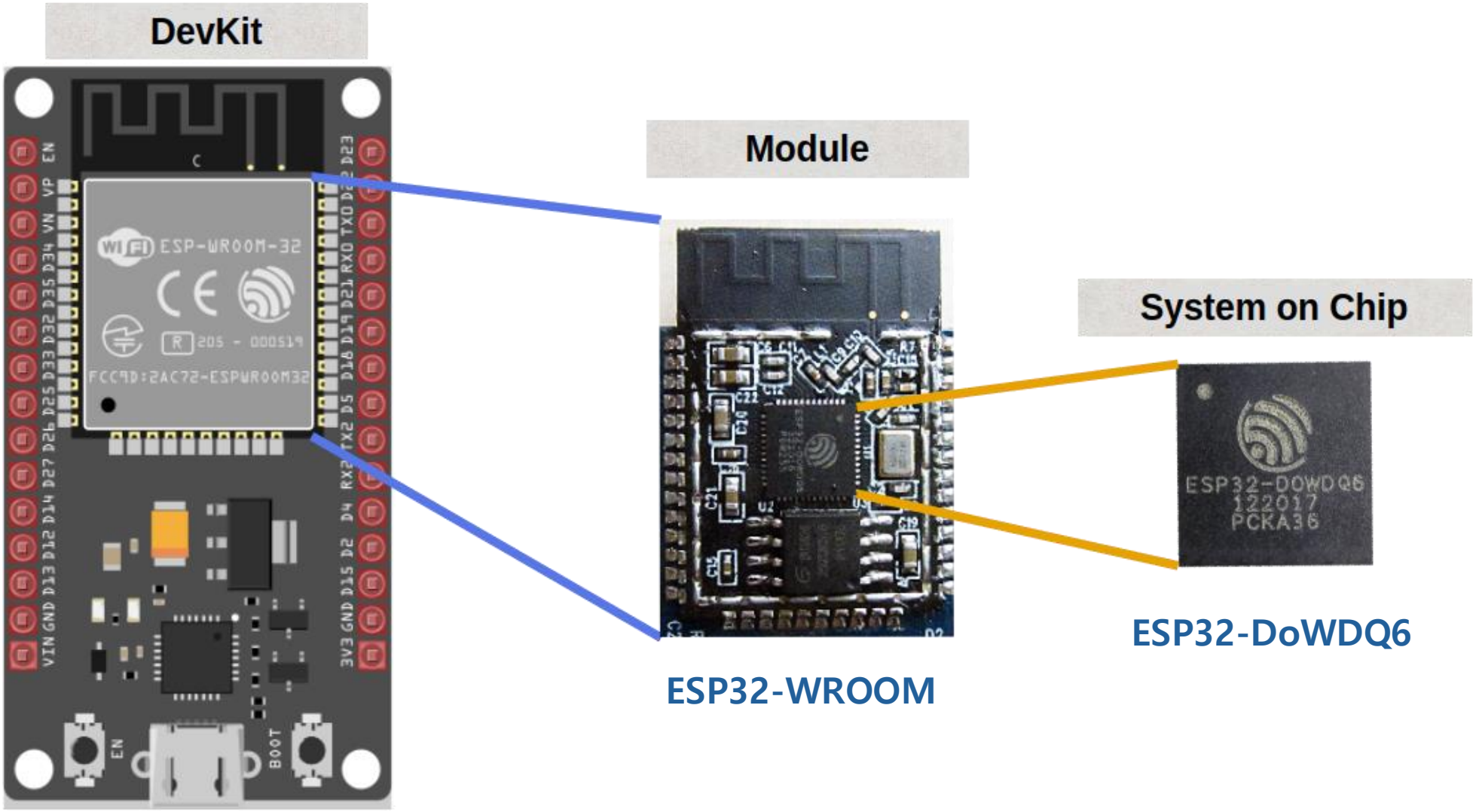
무선 (RF)

- Wi-Fi 6 (802.11ax): 2.4/5 GHz Dual band, Target Wake Time (TWT), BSS Coloring 지원. 높은 동시 연결성과 저전력 효율성 제공.
- Bluetooth 5 /6(LE): BLE Long Range 지원, BLE extended adv, BLE Long Range 지원.
- Zigbee 3.0 / Thread 1.4 (MATTER) 지원



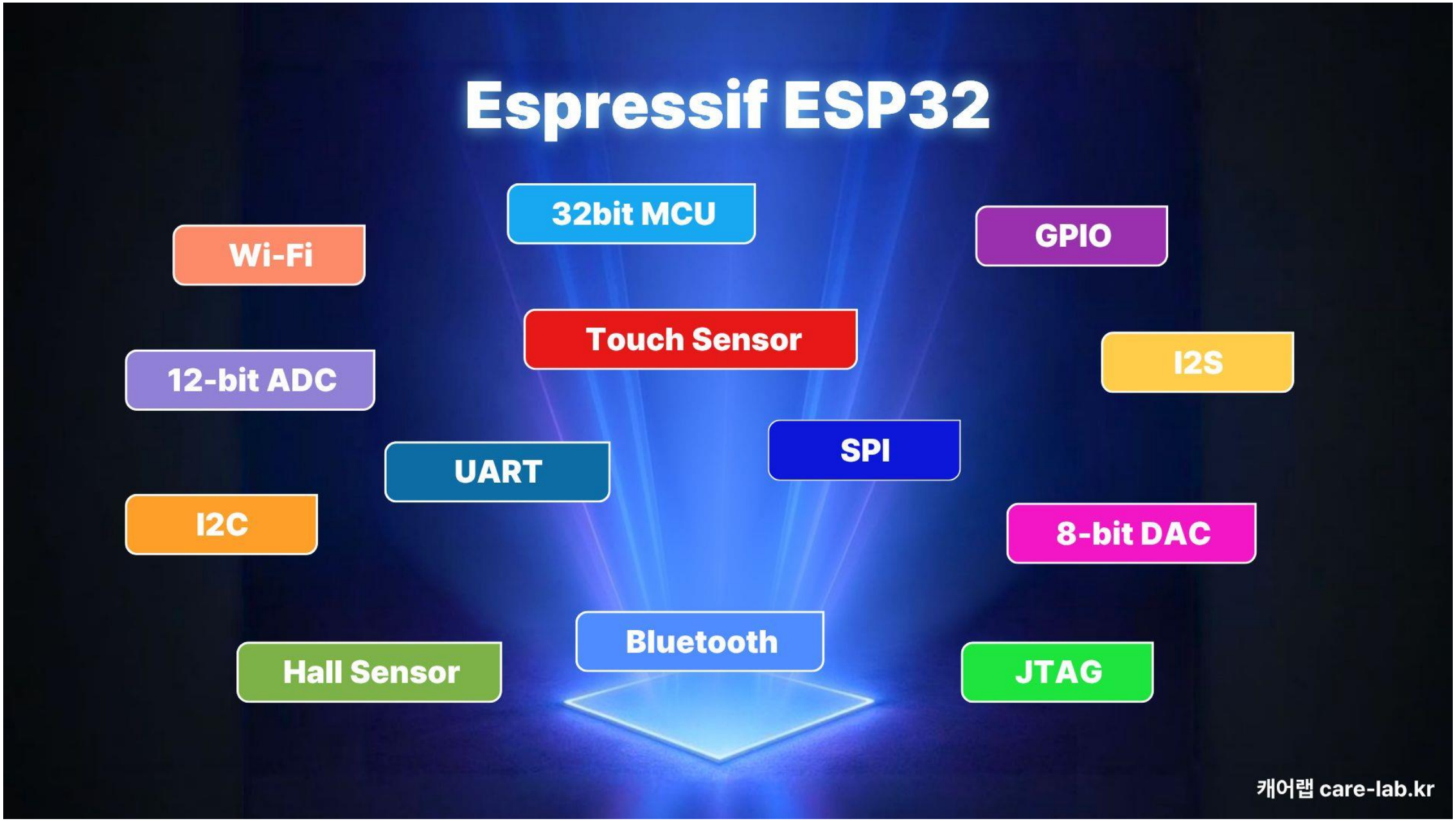
https://espressif.co.kr/26

ESP32 SoC, 모듈 및 DevKit

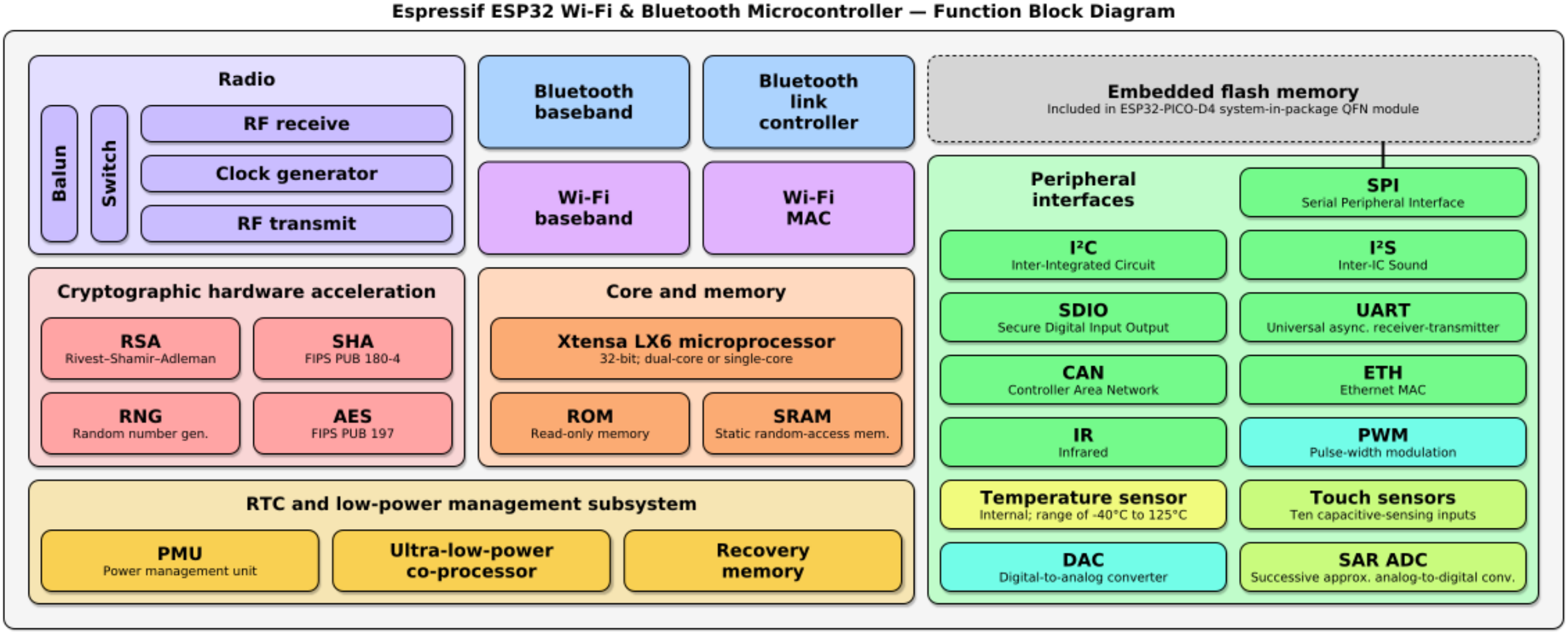


ESP32 제품 선택기 https://products.espressif.com/#/product-selector?language=en&names=

Espressif ESP32



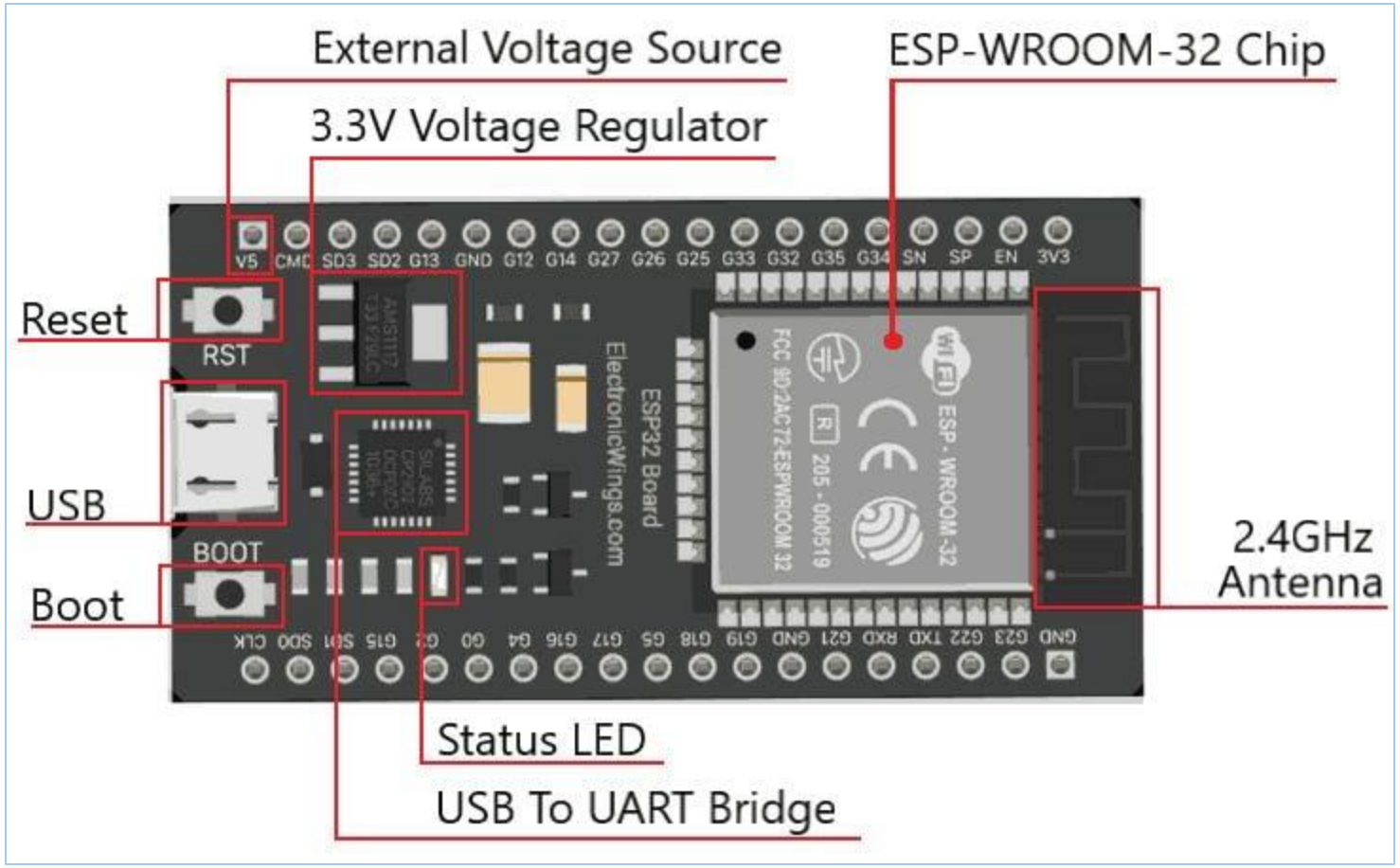
ESP32 Function Block Diagram



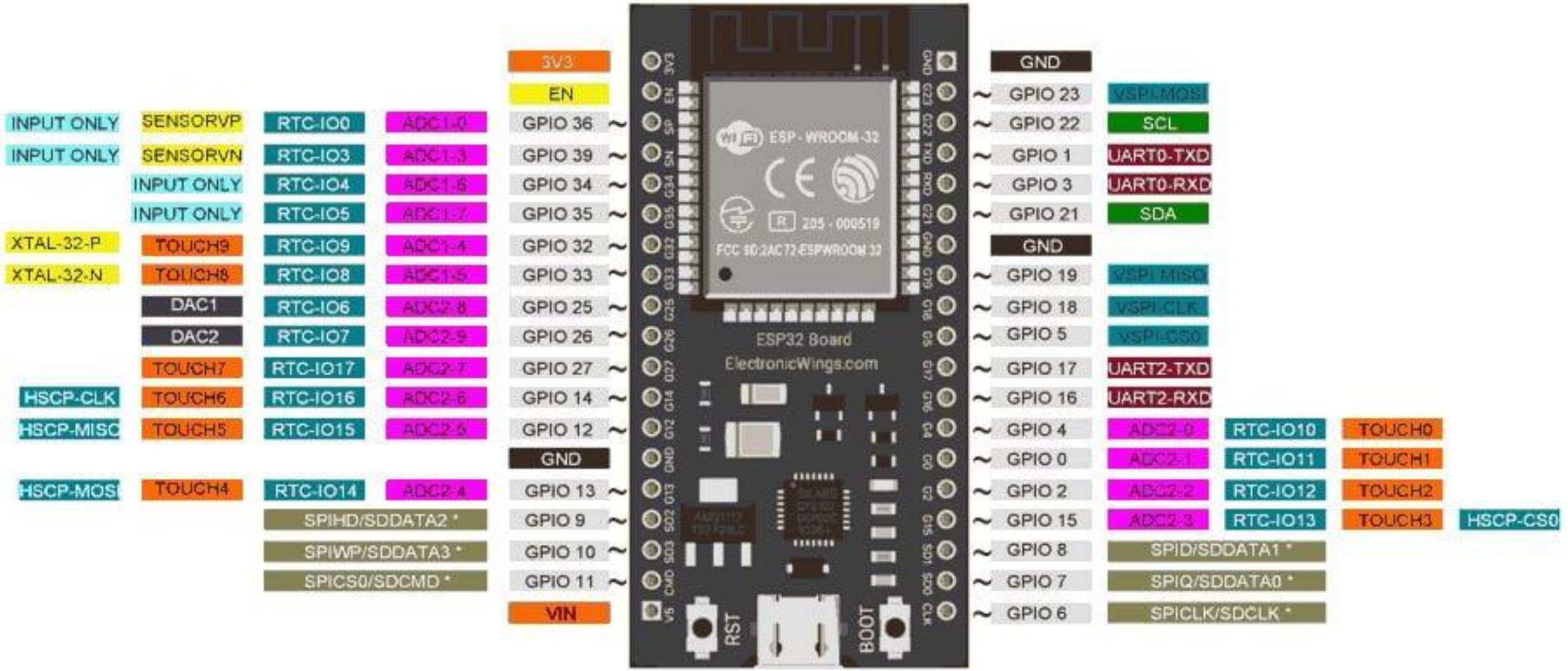
Introduction to ESP32

ESP-Wroom-32 Dev Kit

- ESP-Wroom-32 contains a low-power **Tensilica Xtensa® Dual-Core 32-bit LX6 microprocessor** at 240 MHz: 994.26 CoreMark; 4.14 CoreMark/MHz
- 448 KB of ROM for booting and core functions.
- 520 KB of on-chip SRAM for data and instructions.
- 4MB of Flash Memory
- 16 KB SRAM in RTC
- Wi-Fi 802.11b/g/n
- Bluetooth v4.2 BR/EDR and Bluetooth LE specifications



ESP32 GPIO Pinout



ESP32 프로그래밍 방법

ESP32는 C/C++ 및 MicroPython 프로그래밍 언어를 모두 지원, 지원되는 개발환경(IDE)

C/C++로 ESP32 시리즈 보드를 프로그래밍할 수 있는 IDE 목록

- **ESP-IDF**
- **Arduino IDE**
- **VS Code**

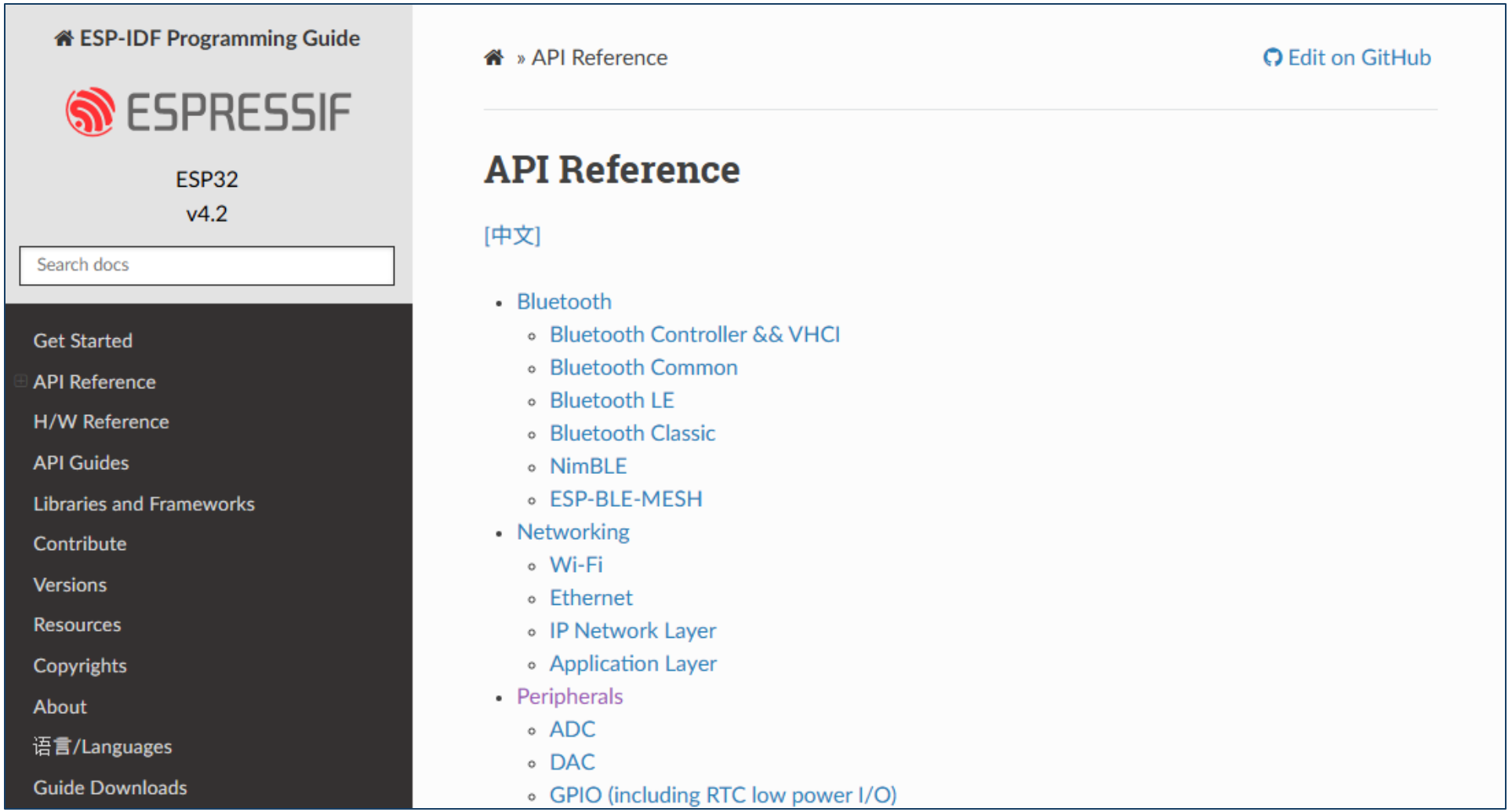
MicroPython으로 ESP32 시리즈 보드를 프로그래밍할 수 있는 IDE 목록

- **Thonny IDE**
- **PyCharm**
- **Mu Editor**
- **uPyCraft IDE**
- **VS Code + Pymakr 확장 프로그램**

ESP-IDF API Reference

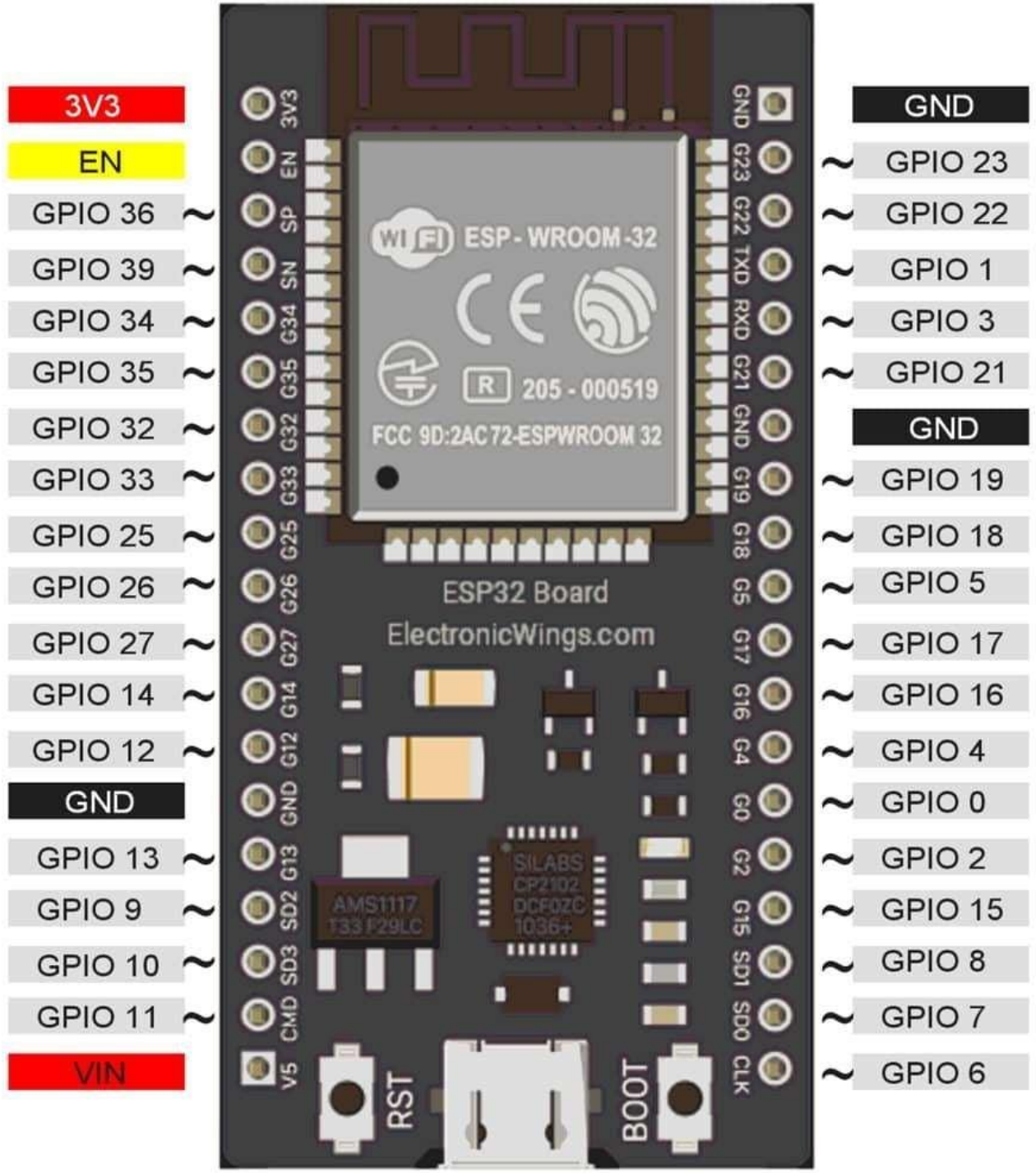
API Reference

https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/index.html

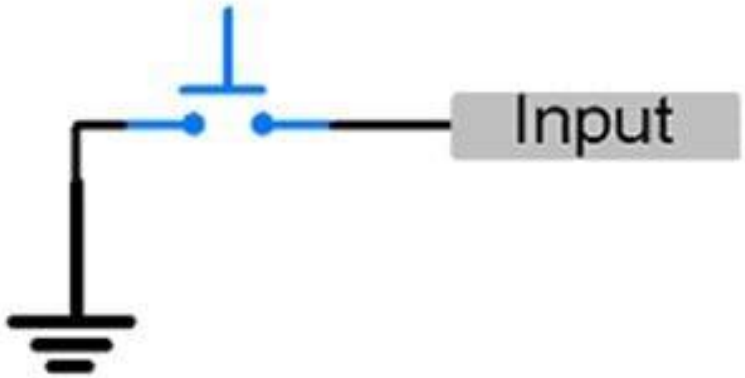


ESP32 GPIO Pin

- **pinMode(pin no, Mode)**
  - INPUT
  - OUTPUT
  - INPUT_PULLUP
- **digitalWrite(pin no, Output value)**
  - HIGH
  - LOW
- **digitalRead(pin no)**

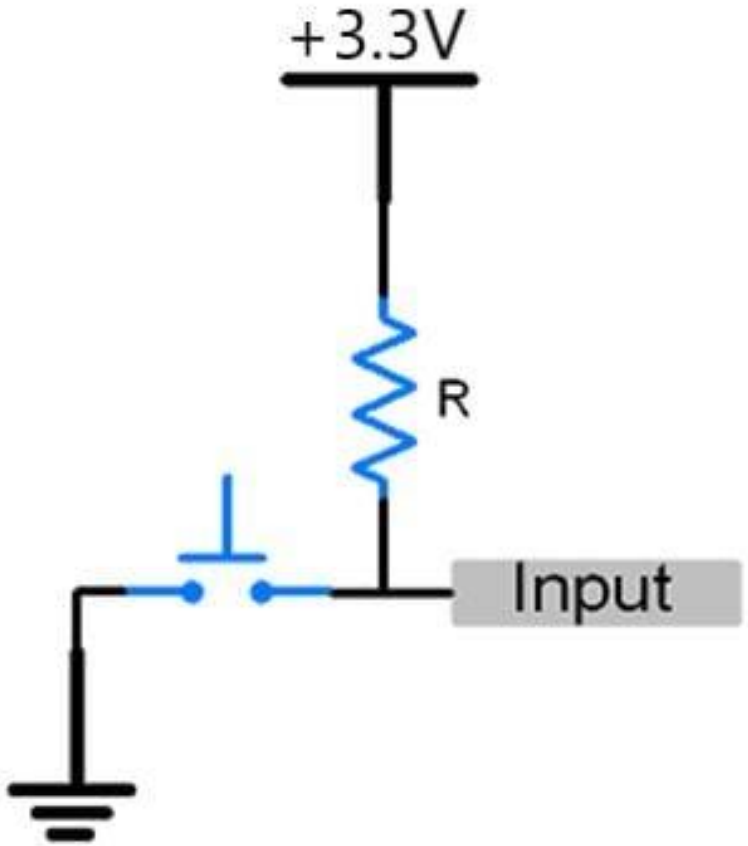


ESP32 GPIO Pull-up, Pull-Down



switch close = 0 V
switch open = undefine

High Impedance State

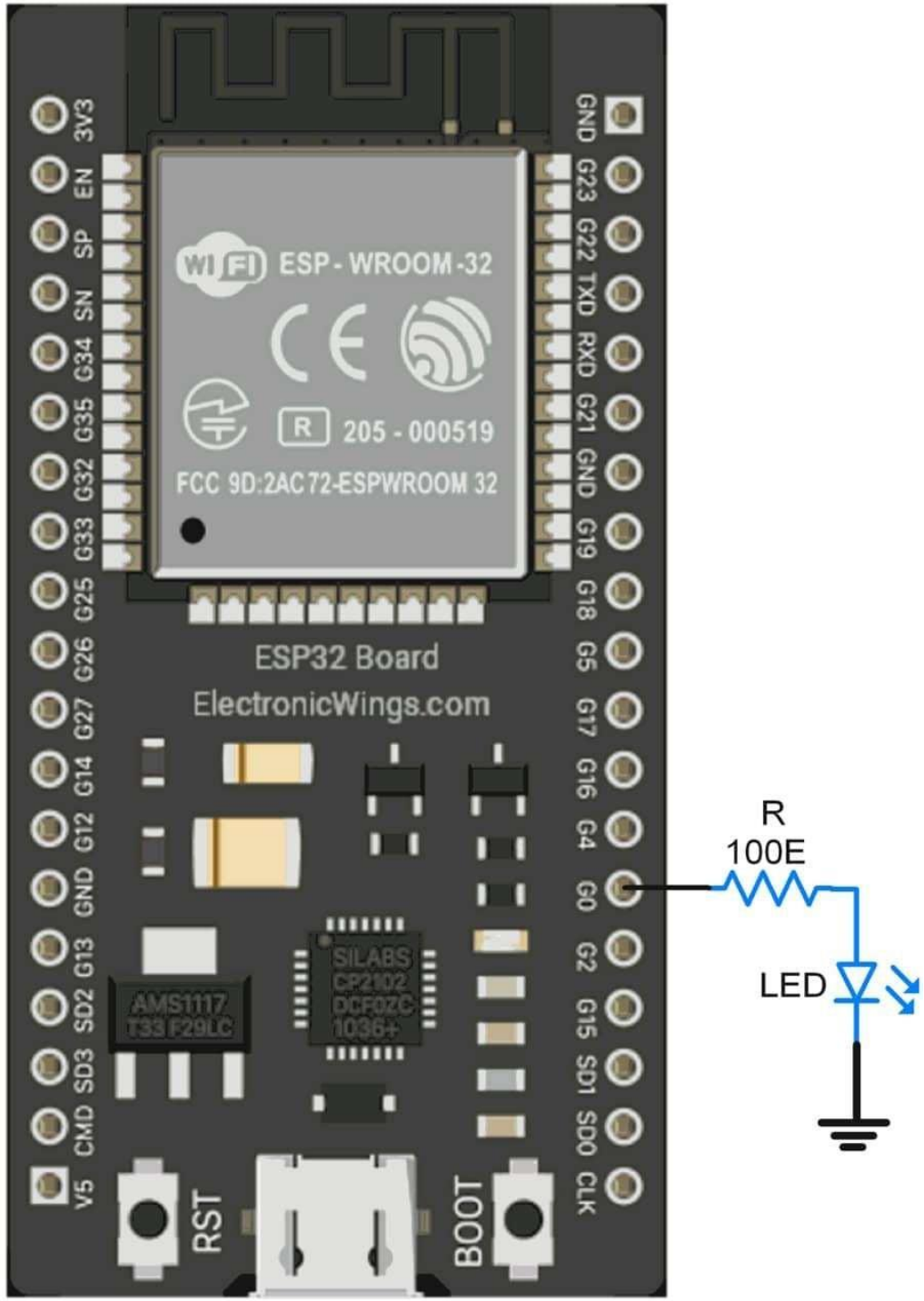


switch close = 0 V
switch open = +5 V

Pull-Up resistor

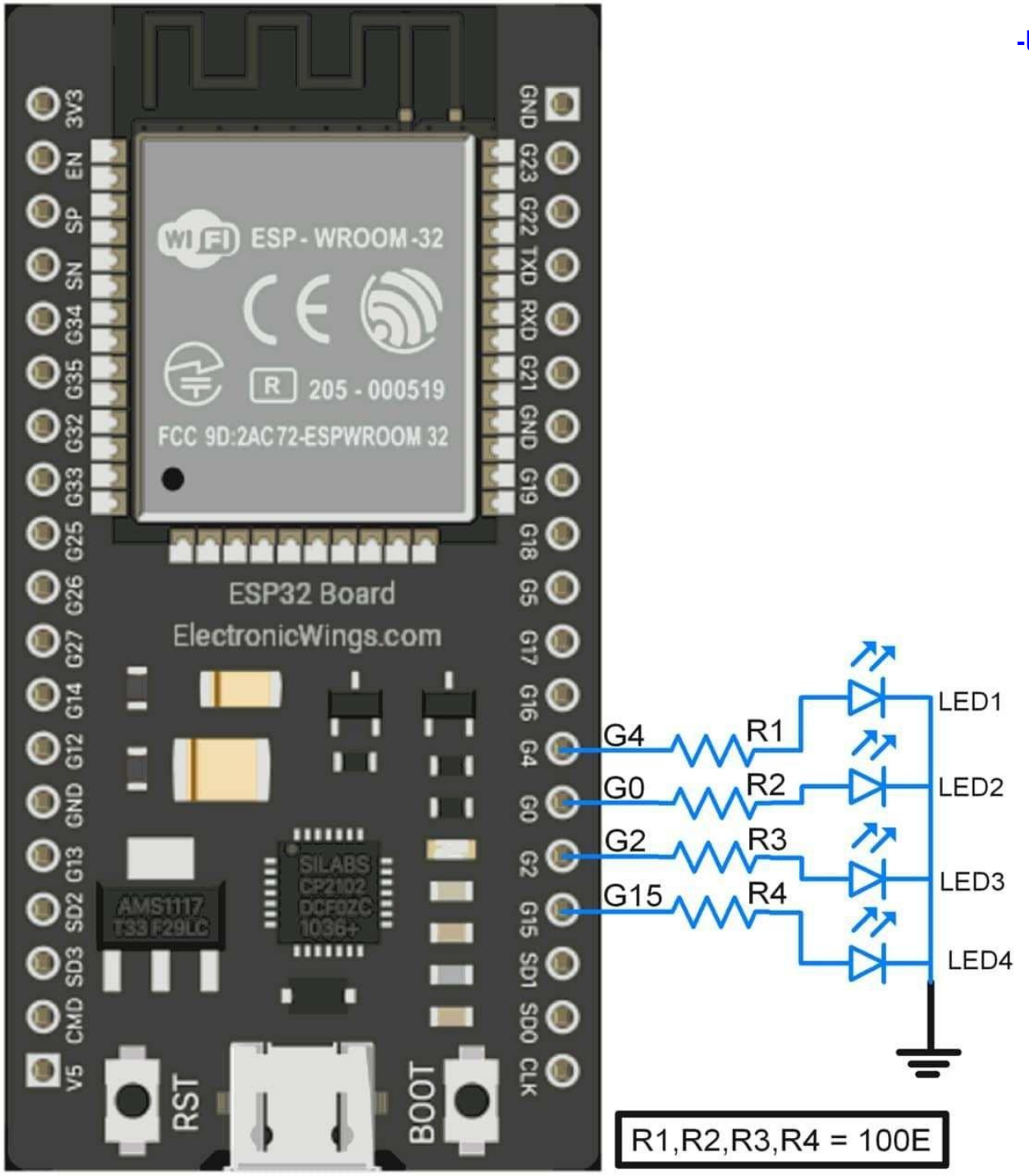
ESP32 GPIO 0 Blink

```
void setup() {
  pinMode(0, OUTPUT);
}

void loop() {
    digitalWrite(0, HIGH);
    delay(1000);
    digitalWrite(0, LOW);
    delay(1000);
}
```



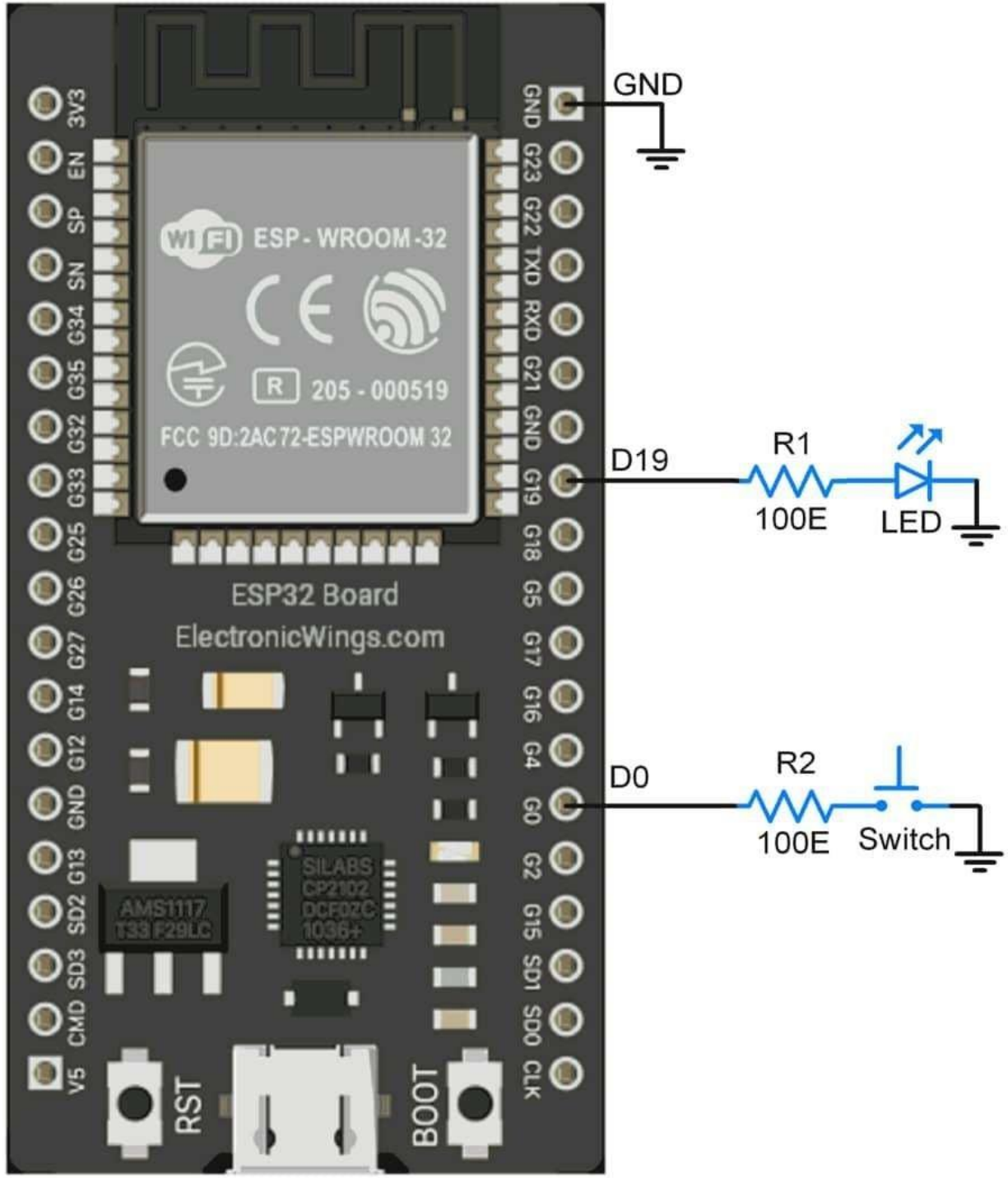
ESP32 순차적 점등


```
void setup() {
  //set gpio pin {0,4,2,15} as output
  pinMode(0, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(15, OUTPUT);
}

void loop() {
 digitalWrite(4, HIGH);          // sets the digital pin 4 on
 delay(1000);                    // waits for a second
 digitalWrite(0, HIGH);          // sets the digital pin 0 on
 delay(1000);                    // waits for a second
 digitalWrite(2, HIGH);          // sets the digital pin 2 on
 delay(1000);                    // waits for a second
 digitalWrite(15, HIGH);         // sets the digital pin 15 on
 delay(1000);                    // waits for a second

 digitalWrite(15, LOW);          // sets the digital pin 15 off
 delay(1000);                    // waits for a second
```



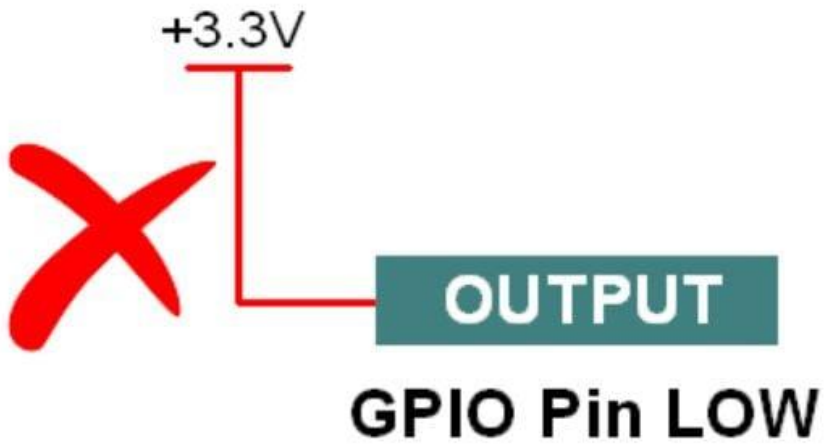
ESP32 스위치 사용

```
int pushButton = 0;
int LED = 19;

void setup() {
    // make the pushbutton's pin an input:
    pinMode(pushButton, INPUT_PULLUP); //config
    pinMode(LED, OUTPUT);              //config
}

void loop() {
    // read the input pin:
    int buttonState = digitalRead(pushButton);
    digitalWrite(LED, (!(buttonState)));    //t
    delay(1);                               //
}
```



ESP32 GPIO 전기적 주의사항

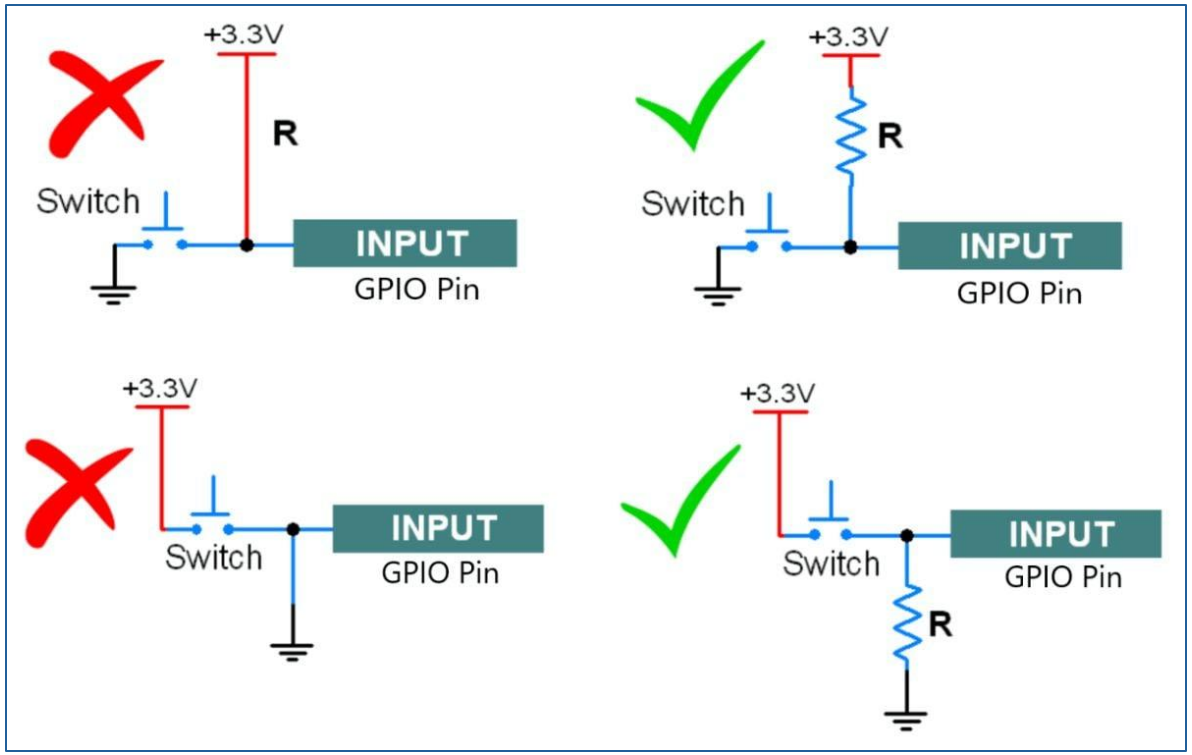
- Output LOW 설정 후 VCC에 연결하지 않기



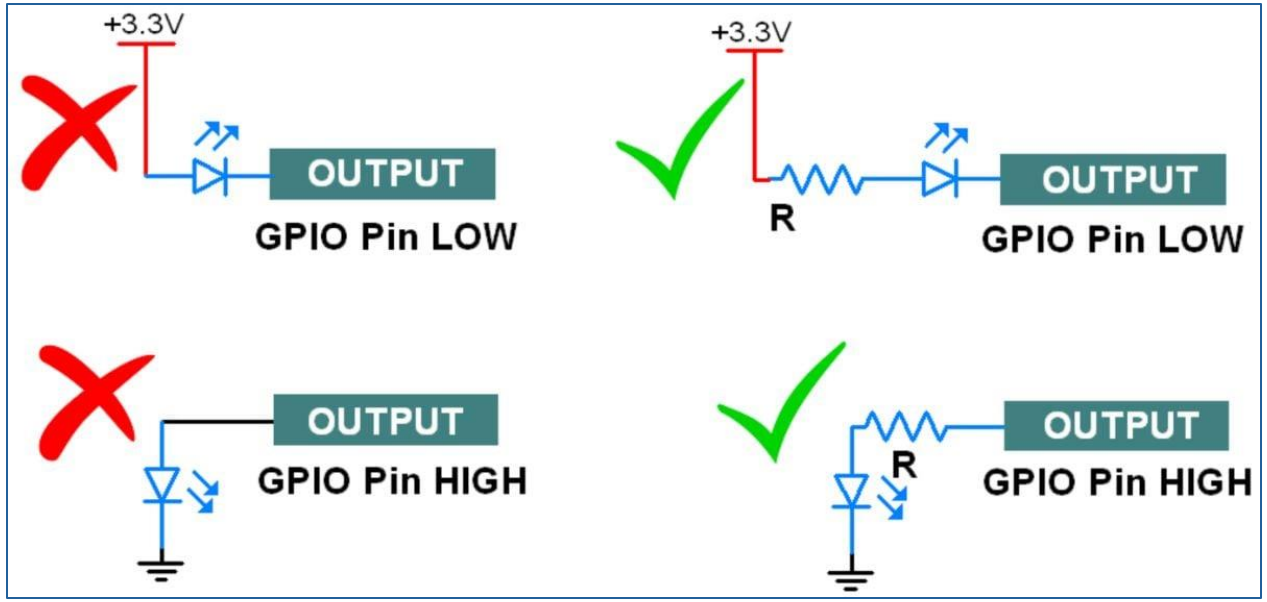
- Output HIGH 설정 후 GND에 연결하지 않기

ESP32 GPIO 전기적 주의사항

- VCC 핀을 직접 GPIO에 연결하지 않기



- LED 연결할 때에는 저항 사용



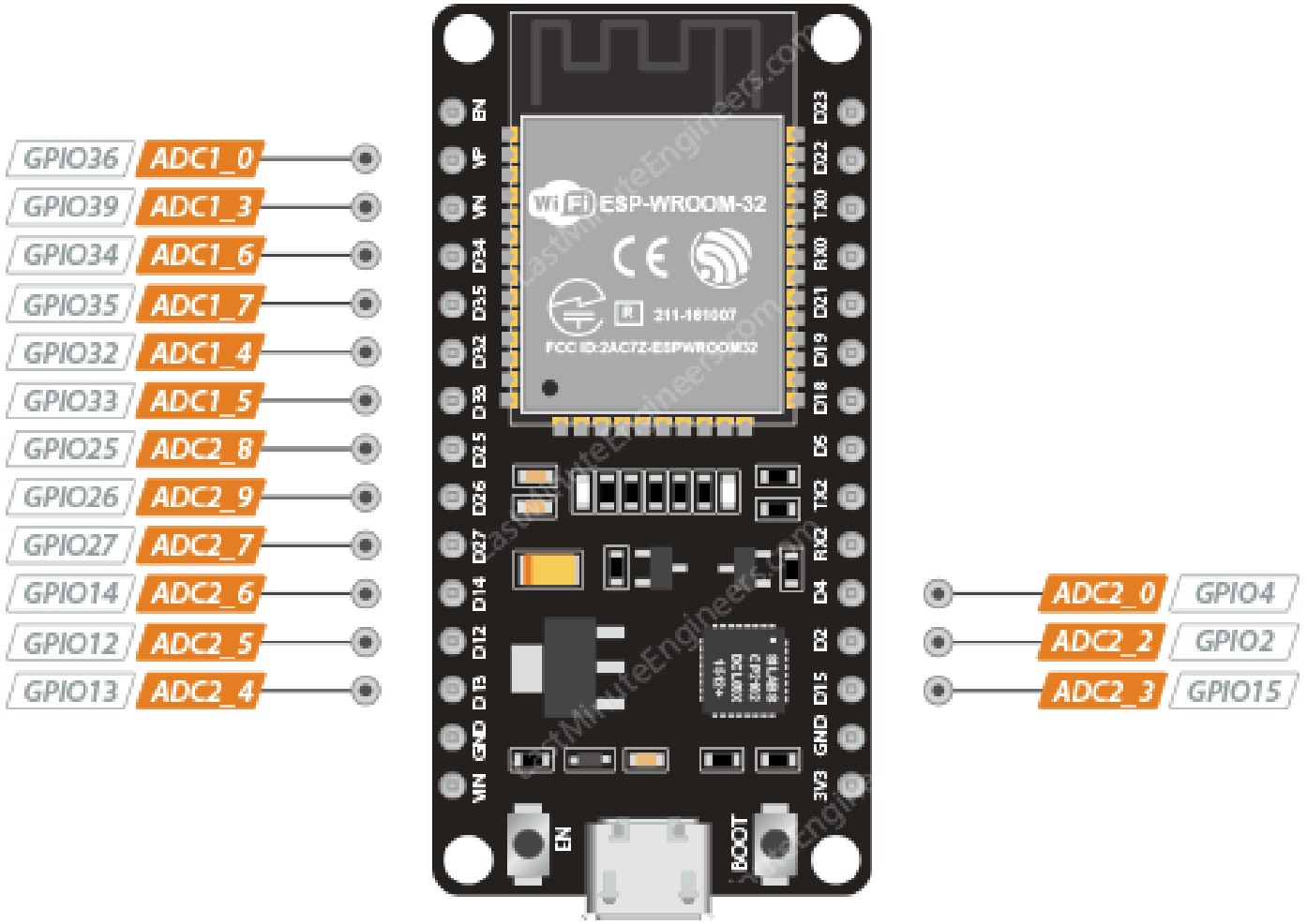
ESP32 12Bit ADC

- **analogRead (pin)**

  returns - digital value 0 – 4095

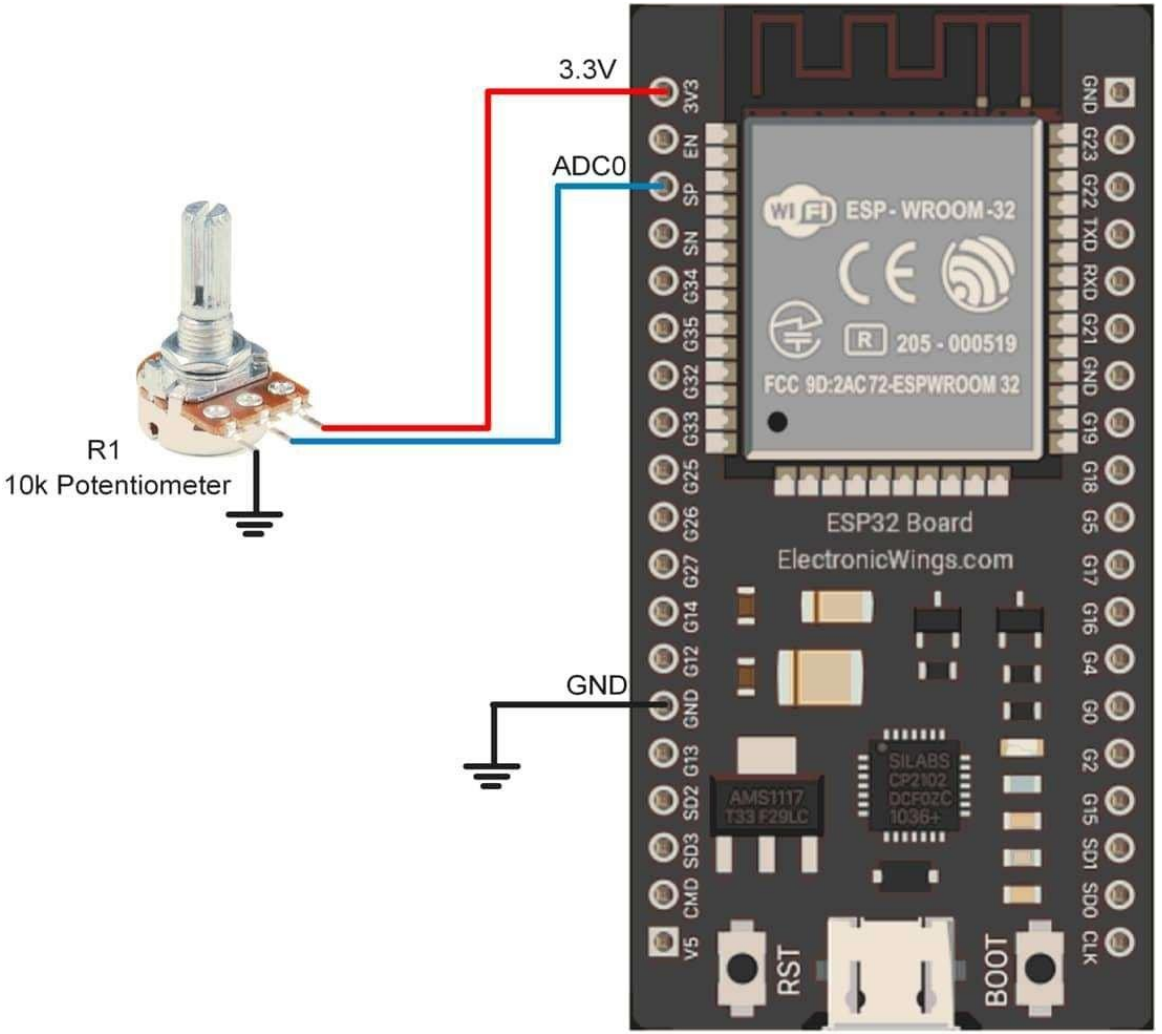
- **analogReference (type)**

- ESP32는 두 개의 12비트 SAR(Successive Approximation Register) ADC를 통합하여 총 18개의 측정 채널(아날로그 지원 핀)을 지원합니다.
- ADC 드라이버 API는 ADC1(8채널, GPIO 32~39에 연결)과 ADC2(10채널, GPIO 0, 2, 4, 12~15 및 25~27에 연결)를 지원합니다. 그러나 애플리케이션에서 ADC2 사용에는 몇 가지 제한 사항이 있습니다:
- ADC2는 Wi-Fi 드라이버에 의해 사용됩니다. 따라서 애플리케이션은 Wi-Fi 드라이버가 시작되지 않은 경우에만 ADC2를 사용할 수 있습니다.
- 일부 ADC2 핀은 스트래핑 핀(GPIO 0, 2, 15)으로 사용되므로 자유롭게 사용할 수 없습니다. 다음 공식 개발 키트에서 그러한 경우가 있습니다:
- ESP32 DevKitC: 외부 자동 프로그래밍 회로로 인해 GPIO 0을 사용할 수 없습니다.
- ESP-WROVER-KIT: 다양한 용도의 외부 연결로 인해 GPIO 0, 2, 4 및 15를 사용할 수 없습니다.



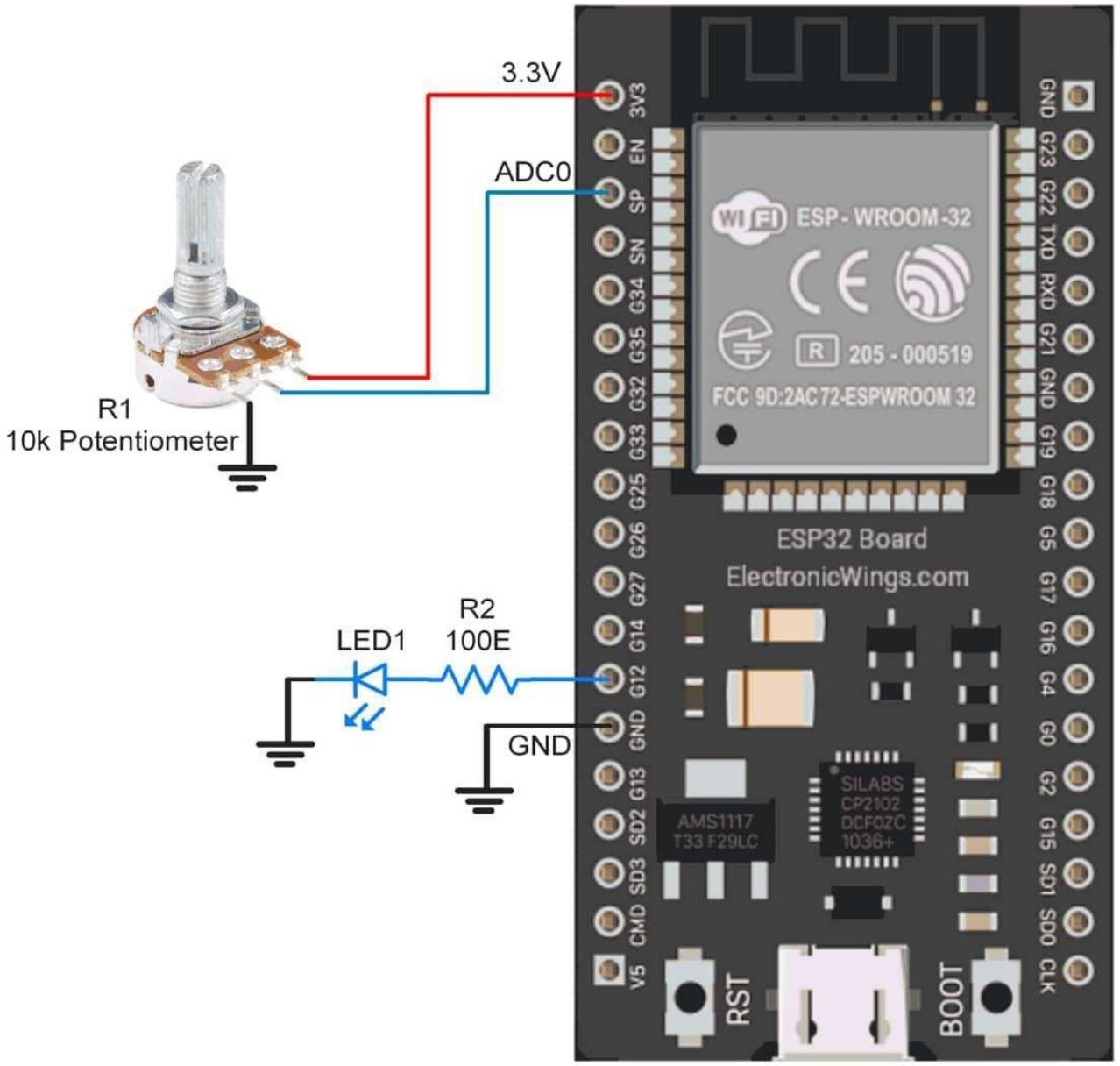
ESP32 12Bit ADC Interface


```
int sensorPin = A0; // input pin for the potentiome
int digitalValue = 0;// variable to store the value

void setup(){
 Serial.begin(9600);
}

void loop(){
 digitalValue = analogRead(sensorPin);// read t
 Serial.print("digital value = ");
 Serial.println(digitalValue);          //print di
  delay(1000);
}
```

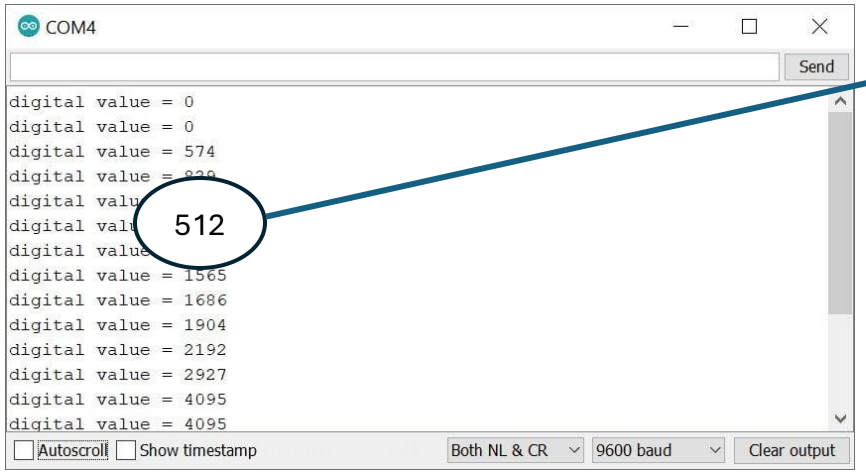


ESP32 ADC Led Control using Potentiometer


```
int ledPin = 12;        // LED connected to digital pin D12
int analogPin = A0;     // potentiometer connected to analog p
int val = 0;            // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);     // sets the pin as output
}

void loop()
{
  val = analogRead(analogPin); // read the input pin
  analogWrite(ledPin, val / 16);// analogRead values go f
}
```

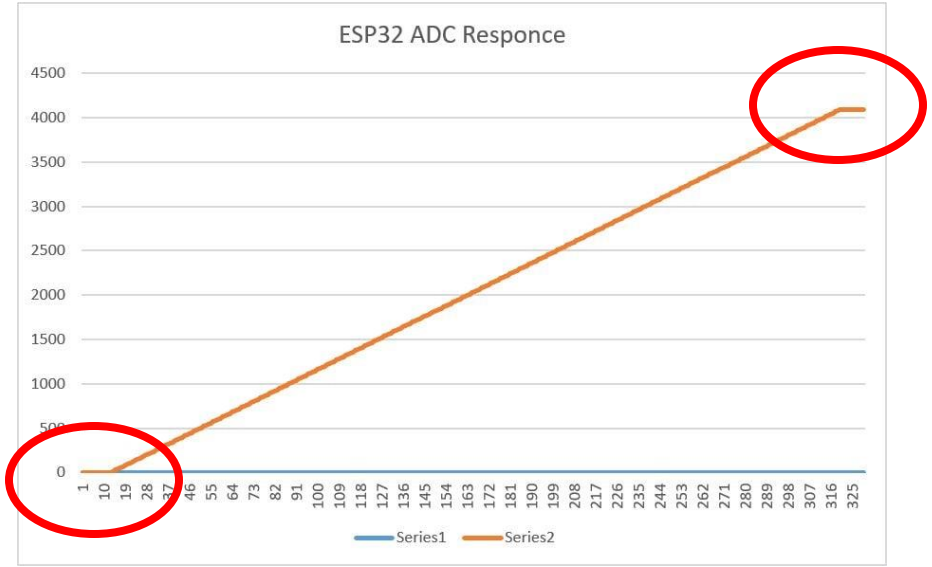


ESP32 12Bit ADC 비선형성



$$\text{Vout} = \text{digital value}\left(\frac{Vref}{2^n - 1}\right)$$

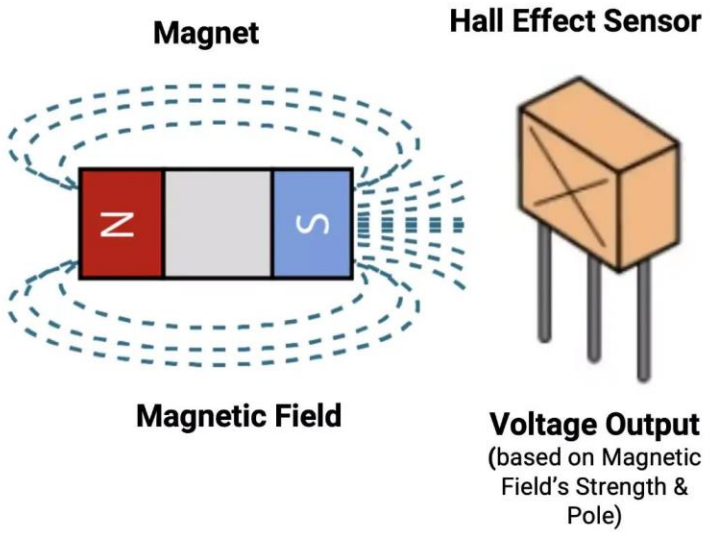
$$\text{Vout} = 512\left(\frac{3.3}{4095}\right) = 0.412\text{V}$$



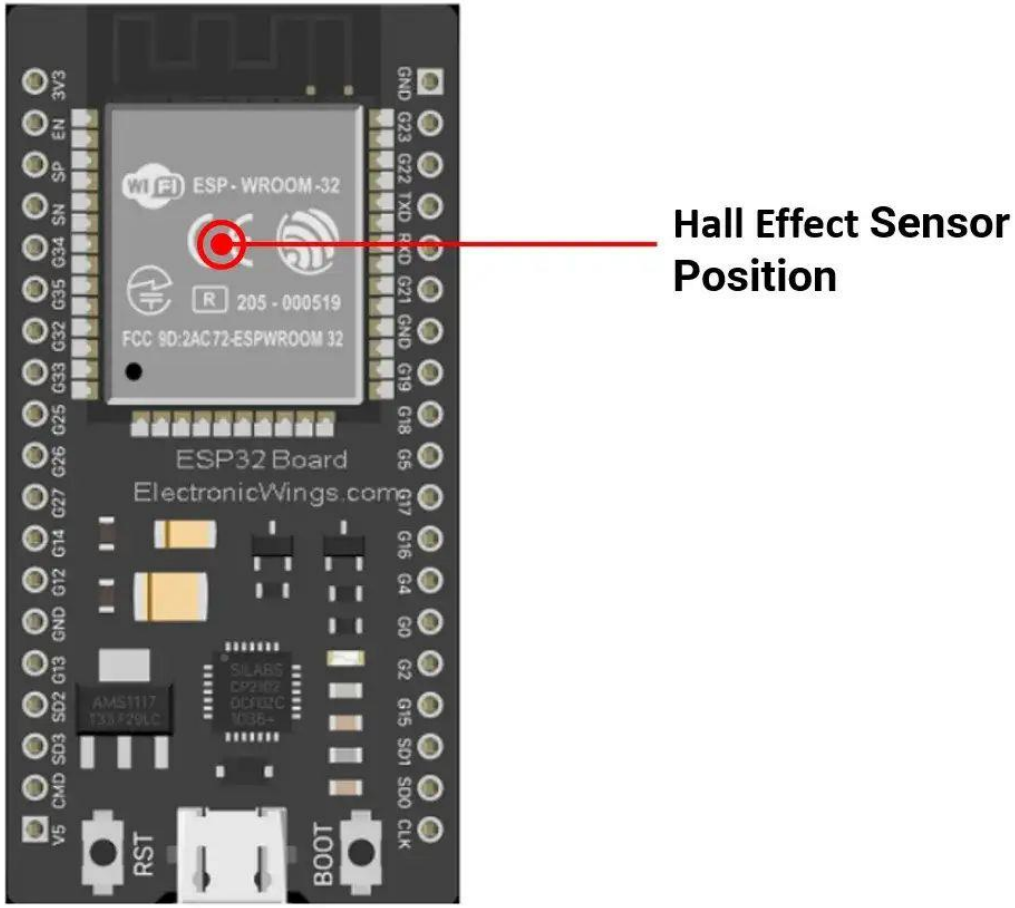

```
int sensorPin = A0;   // select the input pin for the potentiometer
int digitalValue = 0;   // variable to store the value coming from the sensor
float analogVoltage = 0.00;

void setup(){
 Serial.begin(9600);
}

void loop(){
 digitalValue = analogRead(sensorPin);// read the value from the analog chan
 Serial.print("digital value = ");
 Serial.print(digitalValue);          //print digital value on serial monitor
  //convert digital value to analog voltage
 analogVoltage = (digitalValue * 3.3)/4095.00;
 Serial.print(" analog voltage = ");
  Serial.println(analogVoltage);
  delay(1000);
}
```

ESP32 Hall Effect Sensor

자계를 감지하여 전압으로 출력하는 센서, 출력 전압은 자석의 세기에 비례



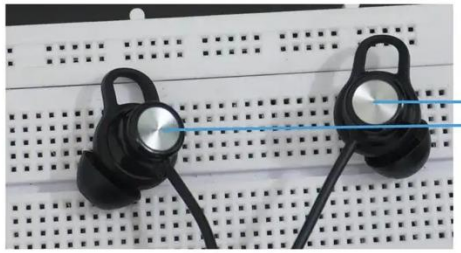
hallRead()



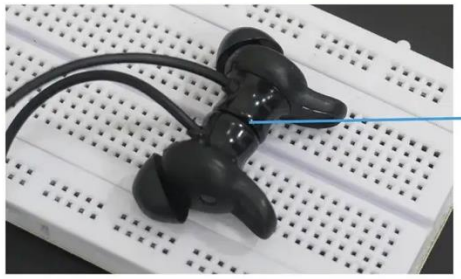
Code

Hall Effect Sensor Readings Without Magnet

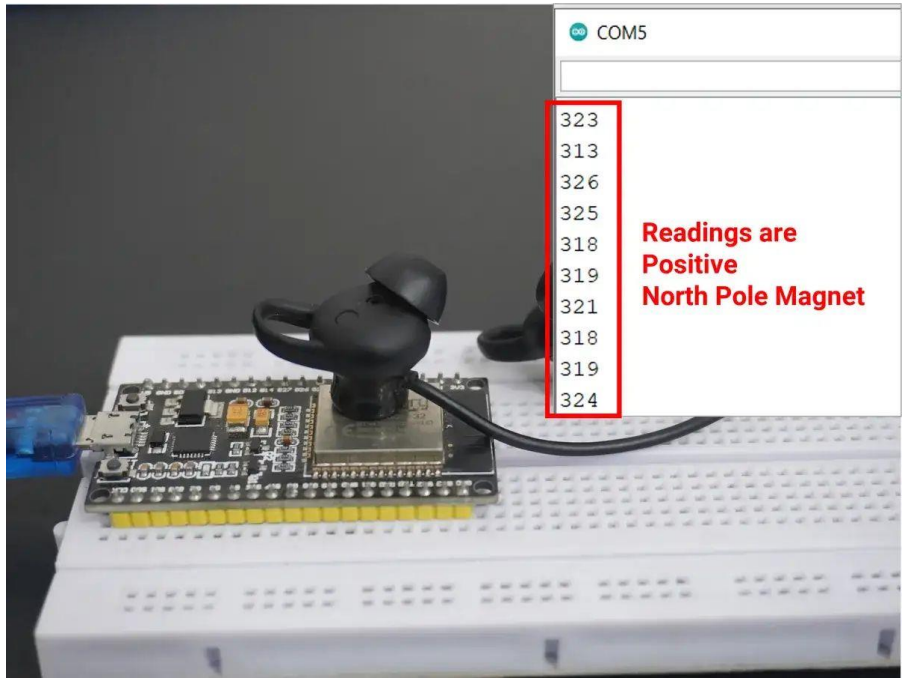
Earphone Magnet

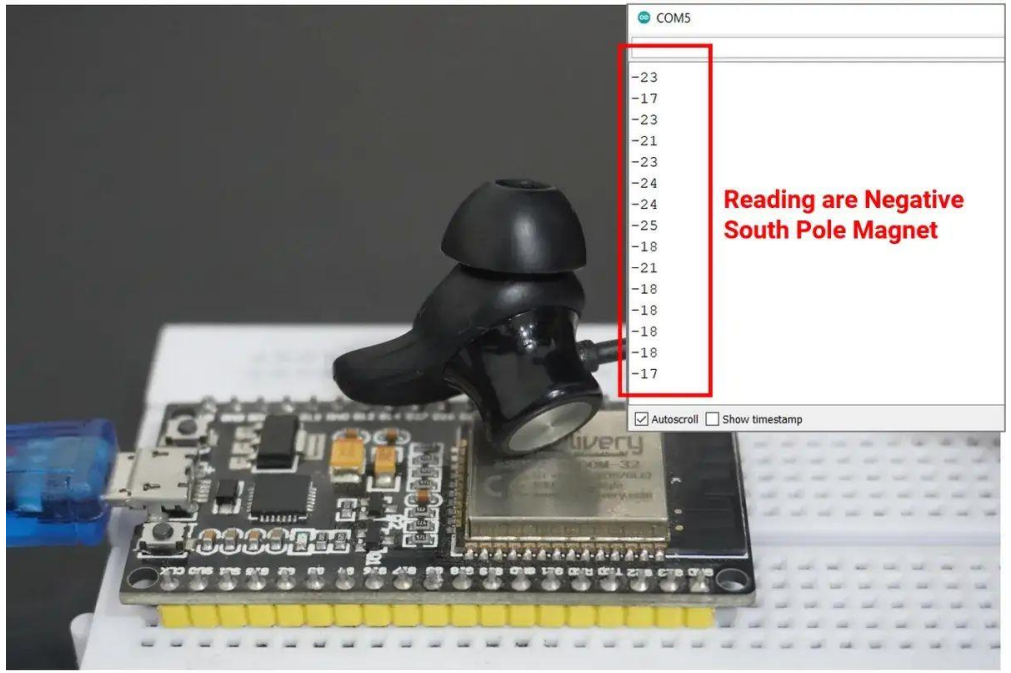


Permanent Magnet
With South & North Pole



Opposite Pole Magnet
Attracts & Sticks

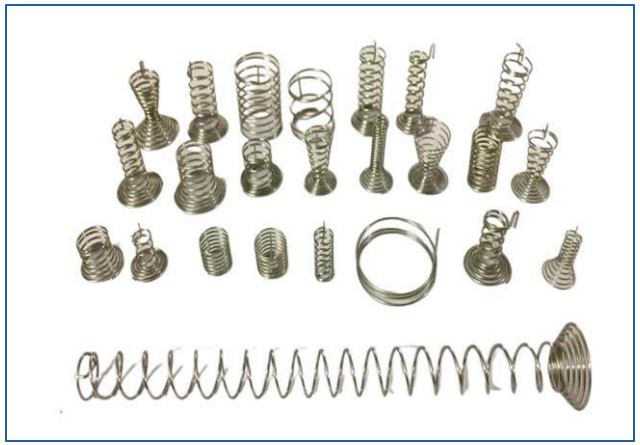


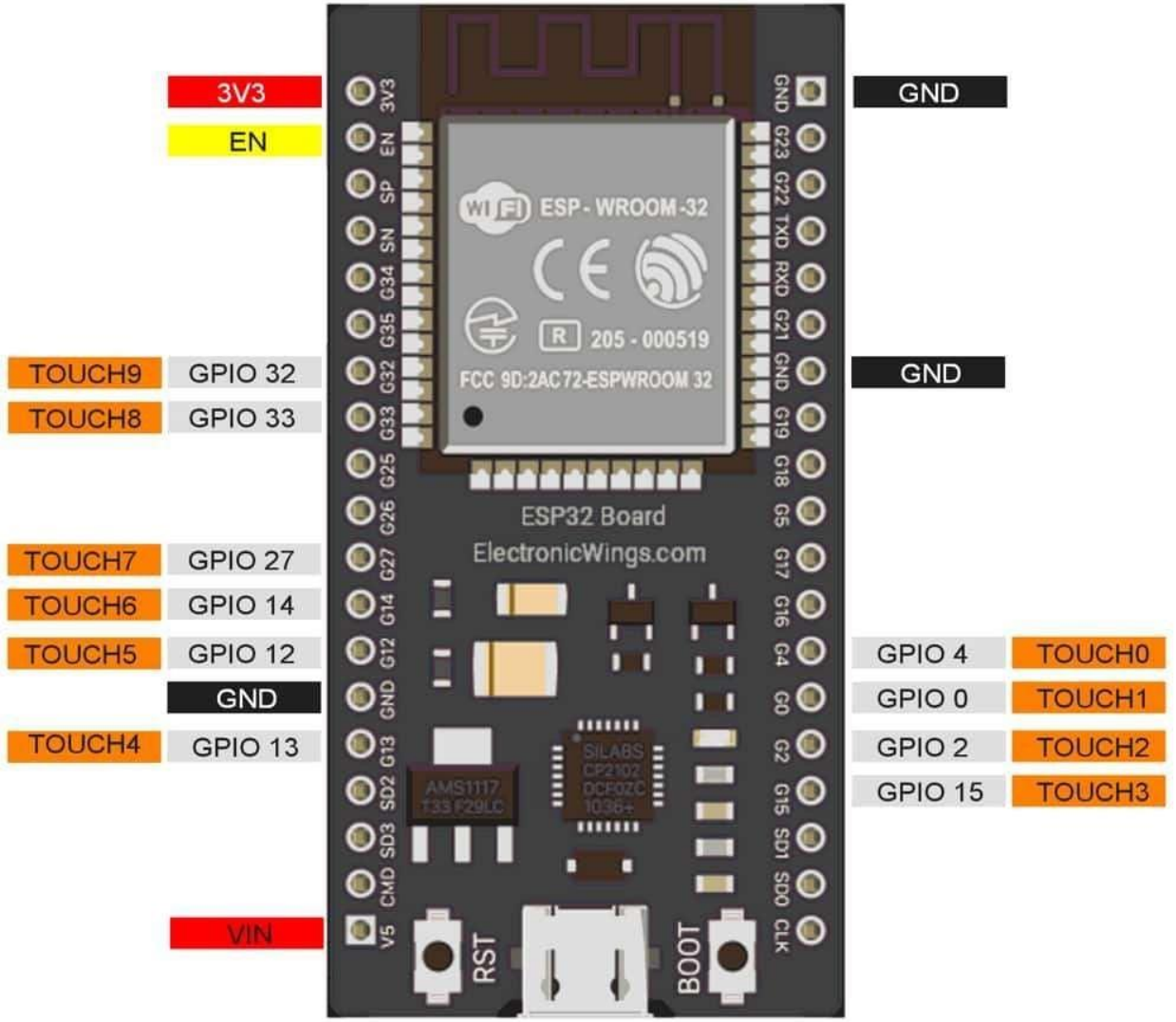


ESP32 TOUCH Sense: 10 정전식 터치 센서

touchRead(touch_sensor_pin_number)

- 터치 센서는 인체의 물리적 접촉을 감지하는 입력 장치입니다.
- 터치 센서의 종류: 저항식 터치 센서, 정전식 터치 센서



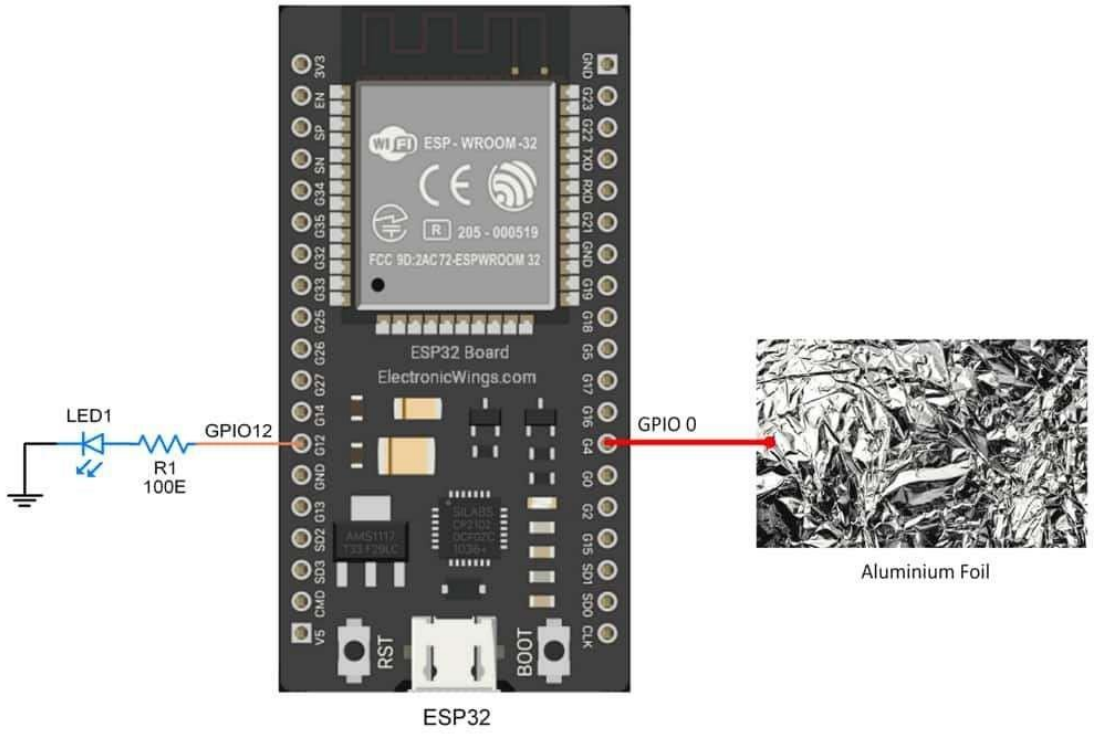


- 저항식 터치: 입력으로 압력을 필요로 합니다. 저항식 터치스크린은 여러 층으로 구성되어 있으며, 터치스크린을 누르면 상단 층이 하단 층에 접촉하여 출력 저항 값을 변화시킵니다. 저렴하고, 전자파 간섭(EMI)에 대한 내성이 우수하며, 내구성이 뛰어나고, 모든 유형의 압력에 대해 작동합니다.
- 정전식 터치: 두 개의 층으로만 구성되어 있고, 인체 자체에 전하가 있어 인체에만 접촉하기 때문에 저항식 터치스크린보다 선호됩니다. 가격이 비싸고, 터치 감도가 높으며, 듀얼 또는 멀티터치를 지원하고, 햇빛 아래에서도 시인성이 좋으며, 저항막방식 터치스크린보다 내구성이 뛰어납니다.

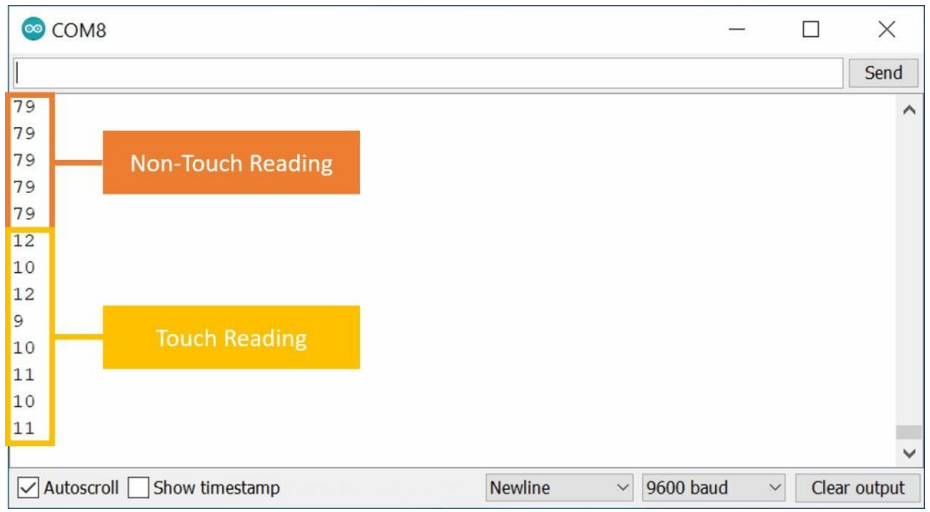
ESP32 TOUCH Sense: 10 정전식 터치 센서

touchRead(touch_sensor_pin_number)

```
const int Touch = 4;    /* Define Touch Input Pin to T0 (GPIO 4)*
const int led  = 12;    /* Define led Pin to GPIO 12 */
int TouchVal, TouchThreshold = 50;

void setup() {
 Serial.begin(9600);   /* Set the baudrate to 9600 */
 pinMode(led, OUTPUT); /* Define led pin as a output */
 delay(500);           /* Wait for 0.5 Second */
}

void loop() {
 TouchVal = touchRead(Touch);
 Serial.println(TouchVal);       /* Get value of Touch 0 pin = GP
 if (TouchVal < TouchThreshold)  /* Check the touch status  */
   digitalWrite(led, HIGH);      /* Turn ON led */

 else
   digitalWrite(led, LOW);       /* Turn OFF led */

 delay(100);                     /* Wait for 100 Mili Second */
}
```



ESP32



ESP32 PWM

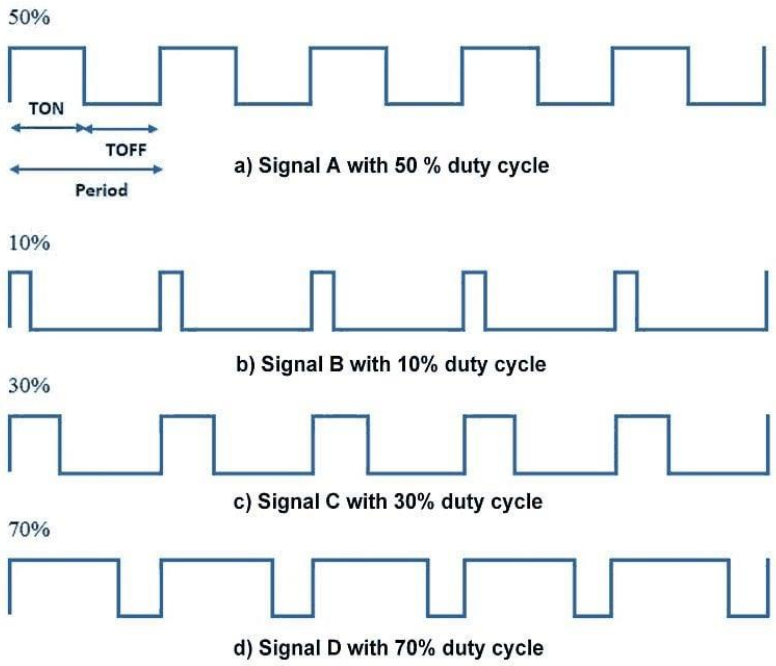
28 Pulse Width Modulation Pin

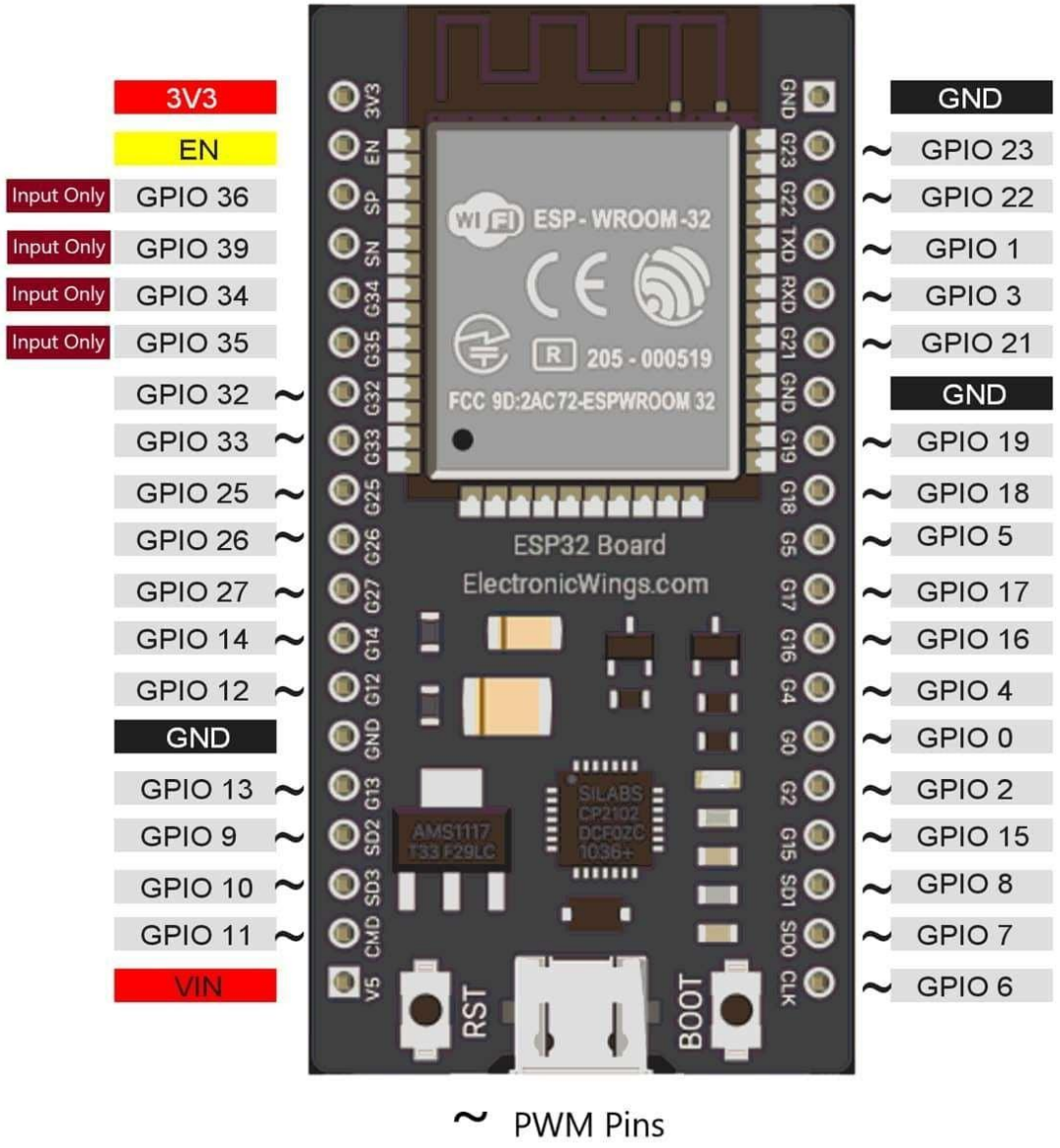
Duty Cycle of Signal

$$DutyCycle = \frac{Ton}{Ton + Toff} * 100$$

예) 10ms 주기 펄스가 2msaks high 일 경우

D = 2ms / 10ms = 20%



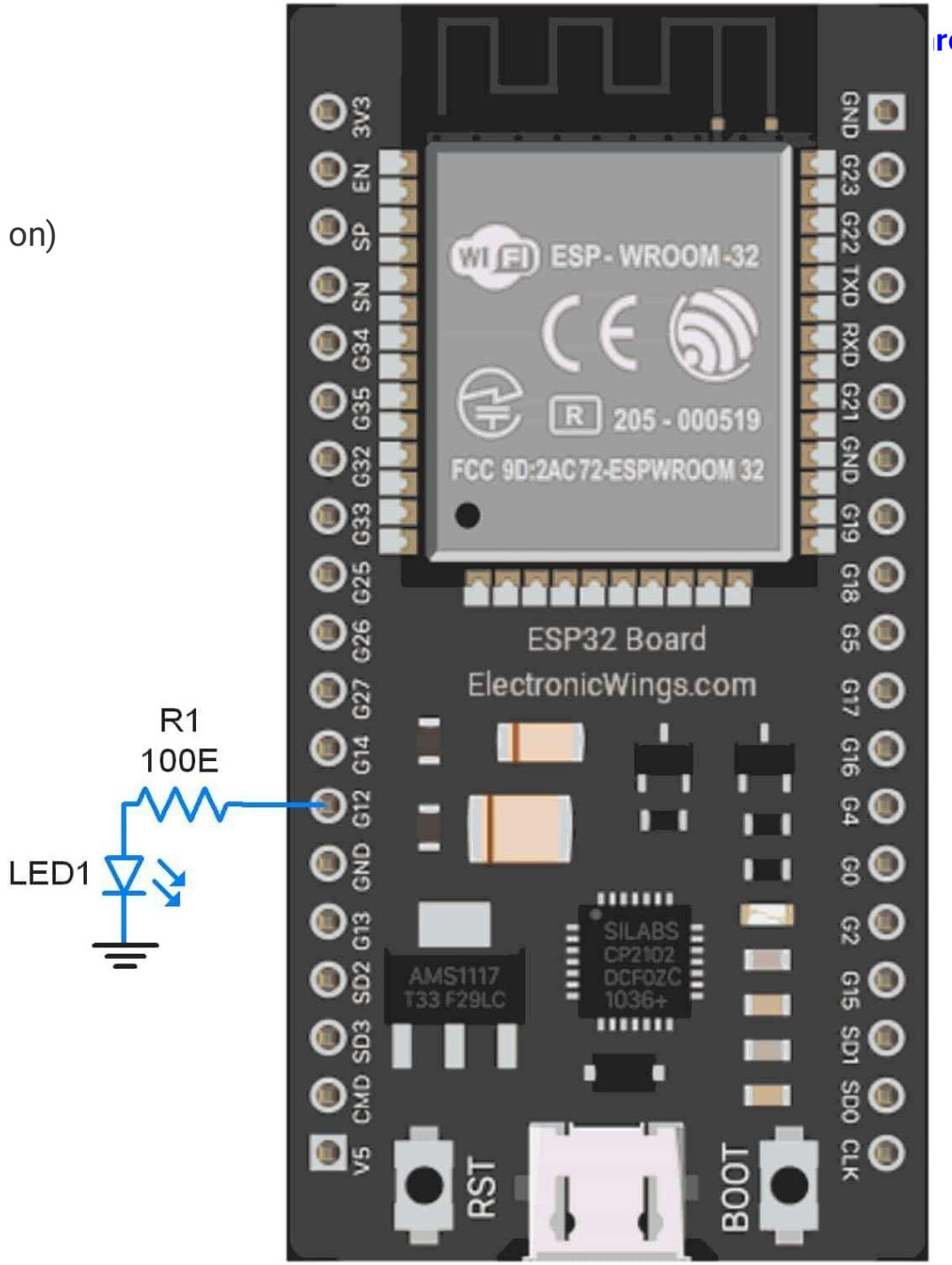


ESP32 Fading using PWM

- **analogWrite (pin, duty cycle)**

  duty cycle – 0 (0%, always off) ~ 255 (100%, always on)

```
int led =12;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

void setup() {
 pinMode(led, OUTPUT); // declare pwm pin to be an output:
}

void loop() {
 analogWrite(led, brightness); // set the brightness of led

 // change the brightness for next time through the loop:
 brightness = brightness + fadeAmount;

 // reverse the direction of the fading at the ends of the fade:
 if (brightness <= 0 || brightness >= 255) {
   fadeAmount = -fadeAmount;
 }
 delay(30);   // wait for 30 milliseconds to see the dimming effect
}
```



ESP32 Interrupt

- 인터럽트는 연속적인 흐름 속에서 무작위로 발생하는 이벤트 ↔ Polling
- ESP32는 두 개의 코어를 가지고 있으며, 각각 32개의 인터럽트를 처리합니다. 각 인터럽트에는 특정 우선순위가 있으며, 대부분의 인터럽트(전부는 아님)는 인터럽트 멀티플렉서에 연결.
- GPIO 핀 인터럽트는 attachInterrupt, detachInterrupt와 같은 Arduino 인터럽트 함수를 통해 지원.
- 인터럽트는 모든 GPIO 핀에 연결될 수 있습니다. HIGH, LOW, CHANGE, RISING, FALLING과 같은 표준 Arduino 인터럽트 유형이 지원.
- 외부 인터럽트: 특정 GPIO 핀의 논리 상태 변화에 의해 발생합니다. 버튼 누름, 동작 센서 작동, 로터리 인코더의 펄스 발생 등을 감지하는 데 외부 인터럽트를 사용할 수 있습니다. 기본적으로 외부 인터럽트는 ESP32가 외부 환경의 변화에 즉각적으로 반응하도록 합니다.
- 내부 인터럽트: 타이머, 터치 센서, 통신 주변기기(UART, I2C, SPI) 등 ESP32 내부 하드웨어에 의해 생성됩니다. 예를 들어, 1초 카운트 타이머를 설정하면 전체 프로그램을 정지시키는 delay() 함수 없이도 자동으로 인터럽트를 트리거할 수 있습니다. 마찬가지로 ESP32가 시리얼 포트를 통해 데이터를 수신하면 UART 인터럽트가 즉시 프로그램에 알립니다.

https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/system/intr_alloc.html?highlight=interrupt

ESP32 Interrupt - attachInterrupt(GPIOPin, ISR, Mode);

- GPIOPin은 ESP32가 변경을 모니터링해야 할 특정 핀을 지정합니다.
- ISR(Interrupt Service Routine)은 인터럽트가 발생했을 때 즉시 실행하려는 함수의 이름입니다.
- Mode는 어떤 종류의 신호 변화가 실제로 인터럽트를 트리거해야 하는지 정의합니다. 다섯 가지 가능한 모드가 있습니다:

Mode Constant	Trigger Condition	Execution Frequency	Primary Use Case
RISING	LOW -> HIGH transition	Once per event	Button release (pull-down), sensor activation
FALLING	HIGH -> LOW transition	Once per event	Button press (pull-up), sensor deactivation
CHANGE	Any state change (LOW -> HIGH or HIGH -> LOW)	Once per event	Frequency measurement, state toggling
LOW	Pin remains LOW	Continuously while asserted	Fault detection (Use with extreme caution)
HIGH	Pin remains HIGH	Continuously while asserted	Fault detection (Use with extreme caution)

ESP32 Interrupt - Interrupt Service Routine (ISR)

- 인터럽트 해제 detachInterrupt(GPIOPin);
- ISR(Interrupt Service Routine)

```
void IRAM_ATTR ISR() {
  // Code to handle the interrupt
}
```

ISR에 IRAM_ATTR 속성이 필요한 이유는 무엇일까요?
프로그램은 일반적으로 SPI를 통해 ESP32에 연결된 별도의 칩인 외부 플래시 메모리에 저장됩니다. ESP32는 캐시 컨트롤러를 통해 이 플래시 메모리에 액세스하는데, 이는 컴퓨터가 더 빠른 액세스를 위해 자주 사용되는 데이터를 캐시하는 방식과 유사합니다. 이 시스템은 일반적인 프로그램 코드에는 잘 작동합니다. 느린 외부 플래시에서 명령어를 가져오는 동안 약간의 지연이 발생하더라도 큰 문제가 되지 않기 때문입니다. 하지만 인터럽트는 지연 없이 즉시 처리되어야 합니다.

여기서 문제가 발생합니다. 인터럽트가 발생하면 ESP32는 ISR 함수 코드를 즉시 로드하고 실행해야 합니다. 해당 코드가 플래시 메모리에 저장되어 있는 경우, 프로세서는 캐시 컨트롤러를 거쳐 코드를 가져와야 합니다. 하지만 캐시 컨트롤러가 바로 그 순간에 Wi-Fi 전송이나 센서 데이터 읽기와 같이 다른 작업을 하고 있다면 어떨까요? 아니면 인터럽트된 코드가 캐시 자체를 관리하는 도중이라면 어떨까요? 이러한 중요한 순간에 플래시 메모리에 접근하려고 하면 ESP32가 "로드 금지"와 같은 오류와 함께 충돌하거나 예측할 수 없는 동작을 할 수 있습니다.

해결 방법은 ISR 함수를 IRAM(Instruction RAM)이라는 특수 메모리에 저장하는 것입니다. 플래시 메모리와 달리 IRAM은 ESP32 칩에 직접 내장되어 있어 프로세서가 플래시 메모리를 기다릴 필요 없이 즉시 액세스할 수 있습니다. 칩에서 다른 작업이 진행 중이더라도 훨씬 빠르고 항상 사용할 수 있습니다.

IRAM_ATTR 속성을 적용하면 컴파일러가 ISR 함수를 표준 플래시 메모리가 아닌 이 특수 메모리에 저장하도록 지시하여 필요한 순간에 실행할 수 있도록 합니다.

ESP32 Interrupt - Interrupt Service Routine (ISR) 규칙

- ISR은 정상적인 프로그램 작동을 중단시키므로 가능한 한 빠르게 실행되어야 합니다. 길거나 느린 ISR은 전체 프로그램이 예측 불가능하게 동작하거나 다른 중요한 이벤트를 놓치게 할 수 있습니다. 이는 ISR 내부에서 delay(), Serial.print() 또는 실행에 상당한 시간이 소요되는 기타 함수를 절대 사용해서는 안 된다는 의미입니다. 대신 플래그 변수를 설정하거나 메인 프로그램이 나중에 처리할 수 있는 값을 기록하는 것이 가장 좋습니다.

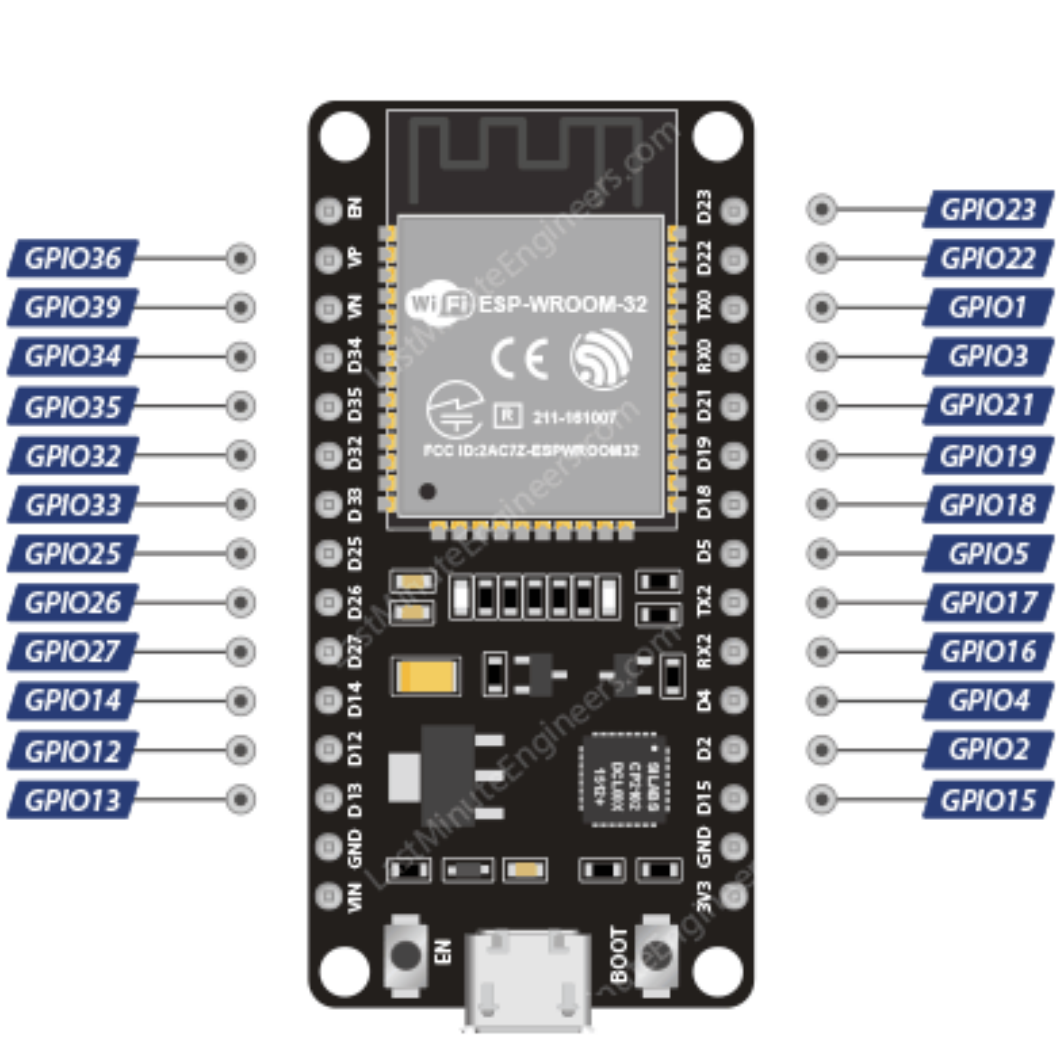
- ISR은 입력 매개변수를 받을 수 없으며 값을 반환할 수도 없습니다. 이는 ISR이 코드가 아닌 하드웨어에 의해 자동으로 호출되기 때문에 정보를 전달하거나 반환 받을 방법이 없기 때문입니다.

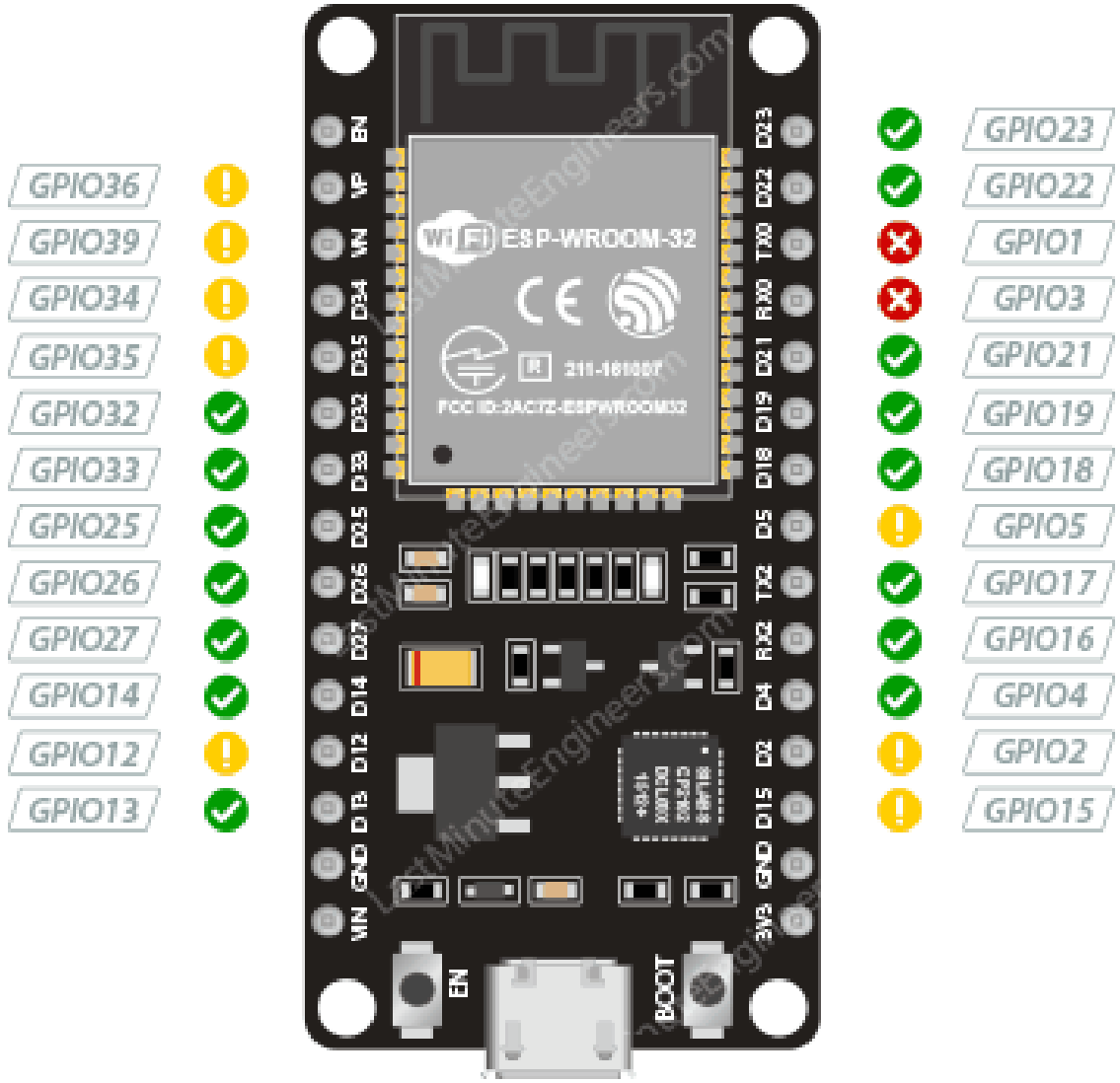
- 공유 변수는 volatile로 선언하십시오. ISR 내부에서 수정되고 외부에서 접근되는 변수(또는 그 반대의 경우)는 반드시 volatile 키워드로 선언해야 합니다. 이는 컴파일러에게 해당 변수가 인터럽트로 인해 언제든 예기치 않게 변경될 수 있으므로, 캐시된 값이나 최적화된 값을 사용하지 말고 항상 메모리에서 실제 현재 값을 읽어야 함을 알립니다. volatile을 사용하지 않으면 프로그램이 인터럽트 중 변경된 내용을 인식하지 못해 추적하기 어려운 버그가 발생할 수 있습니다.

- 복잡한 연산을 피하십시오. 부동 소수점 연산, 복잡한 계산 또는 다른 함수 호출은 절대적으로 필요한 경우가 아니면 ISR 내부에서 수행하지 마십시오.

- IRAM_ATTR 속성을 사용하십시오. 모든 ISR 함수 이름 앞에 IRAM_ATTR 키워드를 추가하여 해당 함수가 외부 플래시 메모리가 아닌 빠른 내부 RAM에 저장되도록 해야 합니다.

ESP32 Interrupt GPIO

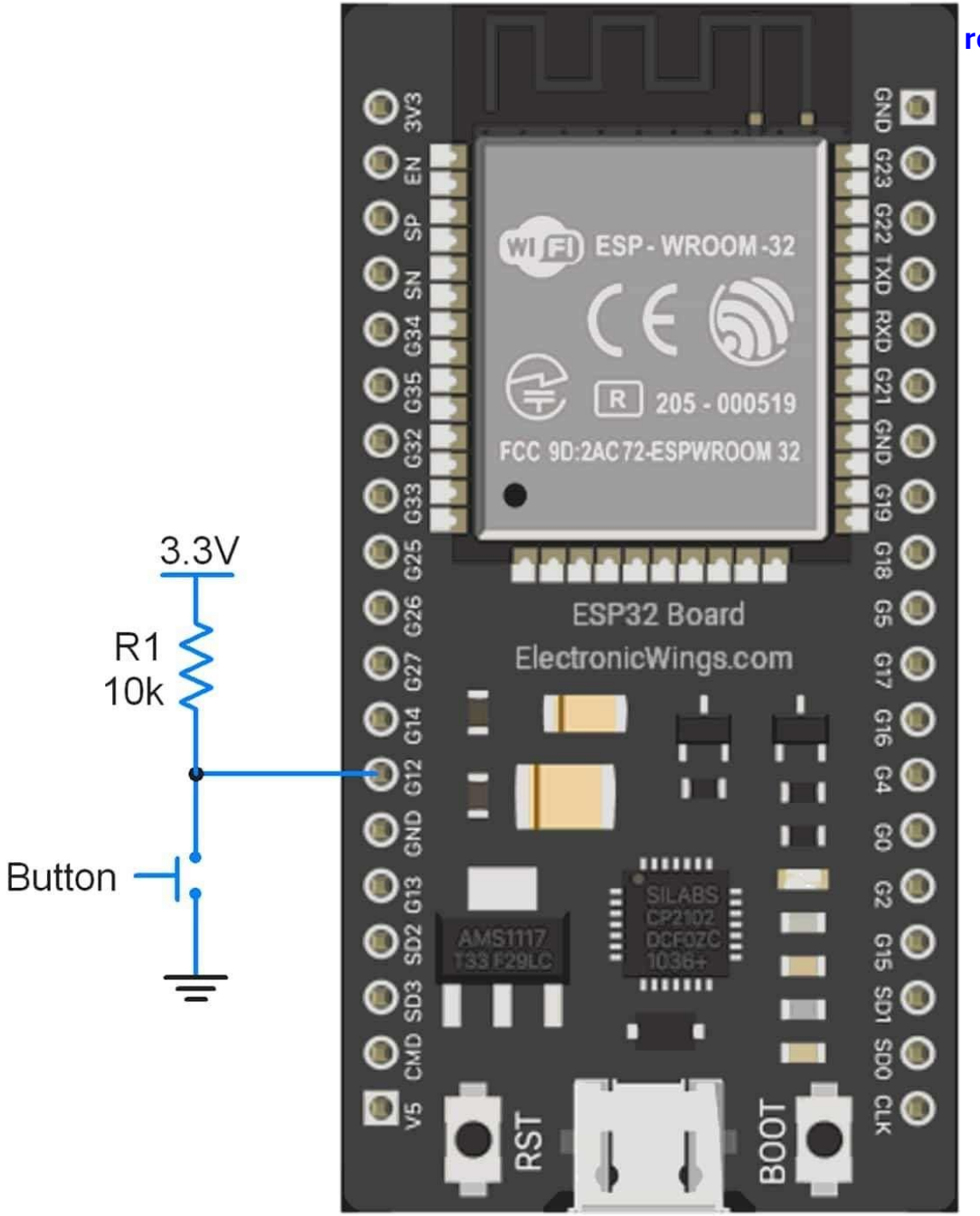




https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/system/intr_alloc.html?highlight=interrupt

ESP32 Interrupt

```
const uint8_t InterruptPin = 12;
bool Request;

void IRAM_ATTR isr() {
 Request = true;
}

void setup() {
 Serial.begin(9600);
 pinMode(InterruptPin, INPUT);
 attachInterrupt(InterruptPin, isr, RISING);
}

void loop() {
 if (Request){
   Serial.println("Interrupt Request Received!");
   Request = false;
 }
}
```

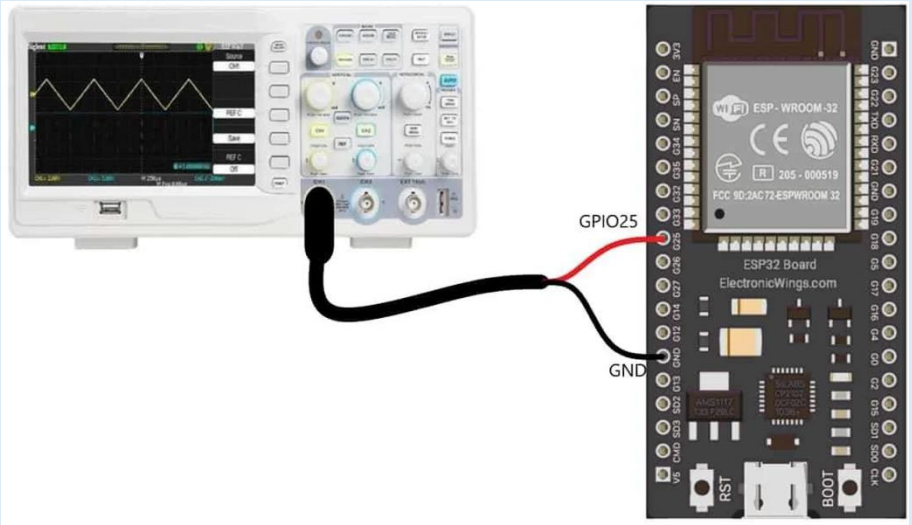


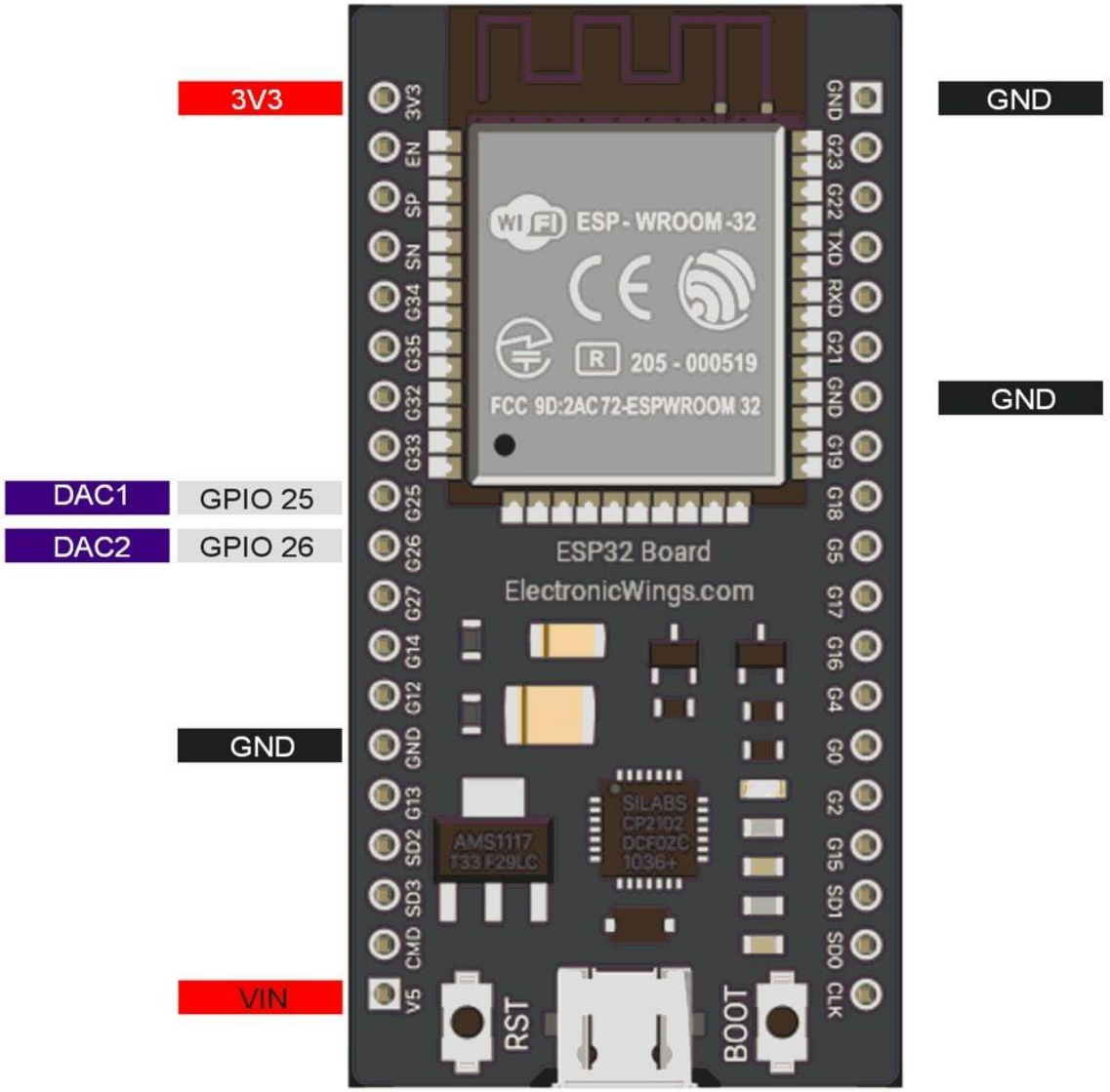
ESP32 DAC

dacWrite(pin, value)

dacDisable(pin)

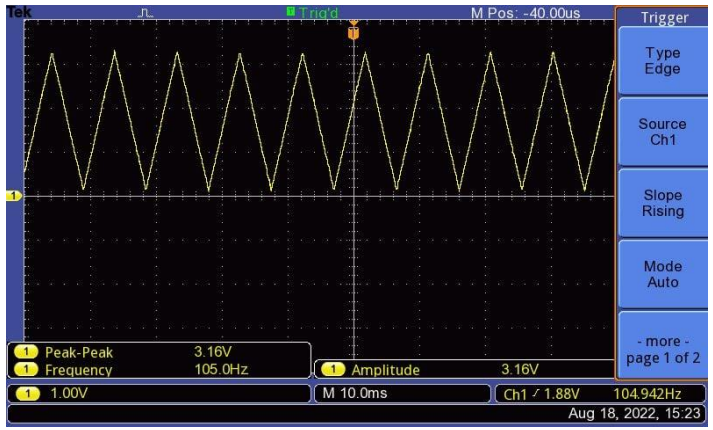
Digital Storage Oscilloscope



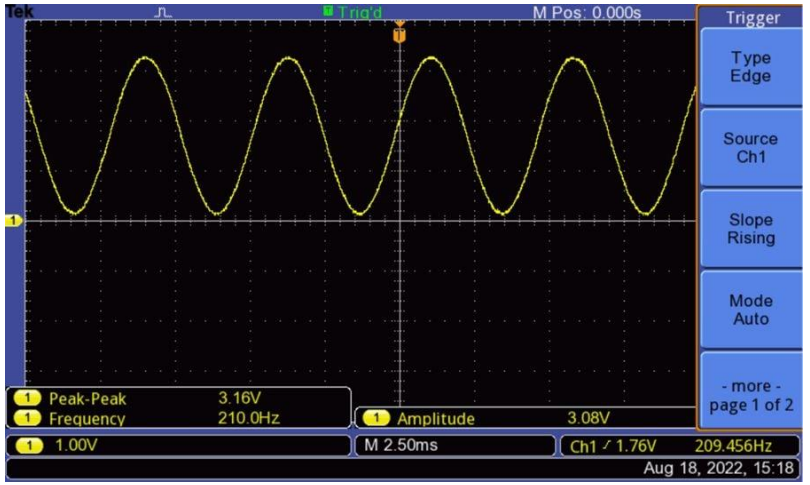


ESP32 DAC

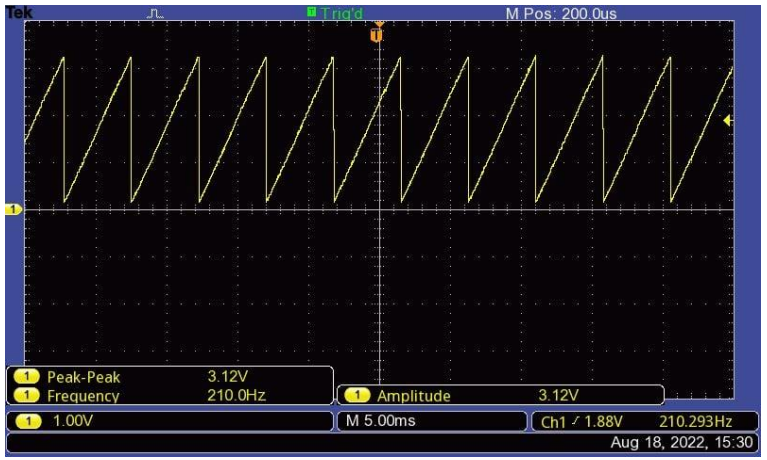

```
#define DAC1 25     //Define Pin no 25 for DAC Output

void setup(){}

void loop() {
/*
* here we are using 8 bit DAC so 2^8=255
* Means 0 = 0V and 255=3.3V
*/
 for (int i=0; i<255; i++){
   dacWrite(DAC1, i);           //write on DAC Pin
 }

 for (int i=255; i>=0; i--){
   dacWrite(DAC1, i);           //write on DAC Pin
 }
}
```




```
/*
 ESP32 DAC Sine Wave Generation
 http://www.electronicwings.com
*/

int Sin_Array[256];                  // Define array to store sine wave values
float Period = (2*PI)/256;           // Define the sine wave conversion factor
float Rad_Angle;

void setup(){
 for(int Angle=0; Angle<256; Angle++) {
   Rad_Angle = Angle*Period;         // calculate the angle in radian
   Sin_Array[Angle] = (sin(Rad_Angle)*127)+128;   // caclulate the anagle an
 }
}

void loop(){
 for(int i=0;i<256;i++)
   dacWrite(25,Sin_Array[i]);        // Write the sinewave vale on the DCA pi
}
```

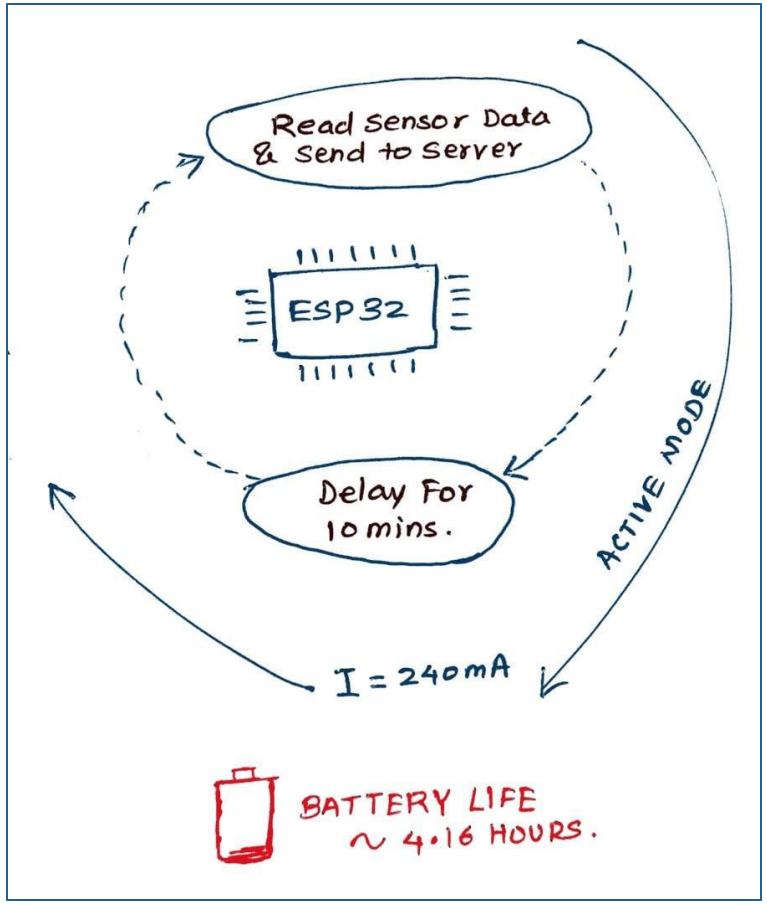


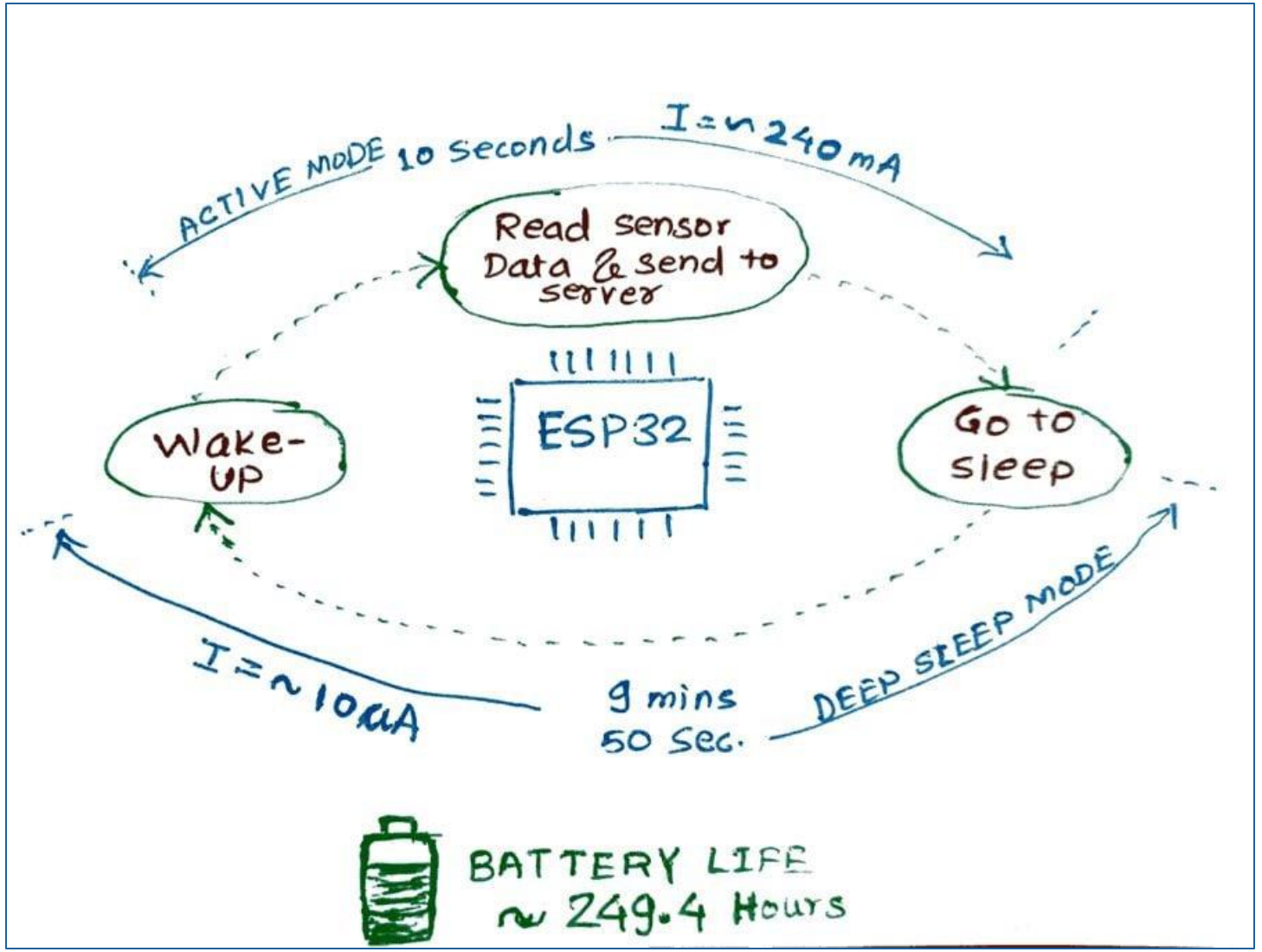

```
/*
 ESP32 DAC Sawtooth Wave Generation
 http://www.electronicwings.com
*/


#define DAC1 25     //Define Pin no 25 for DAC Output

void setup(){}

void loop() {
/*
* here we are using 8-bit DAC so 2^8=255
* Means 0 = 0V and 255=3.3V
*/
  for (int i=0; i<255; i++){
    dacWrite(DAC1, i);           //write on DAC Pin
  }

}
```



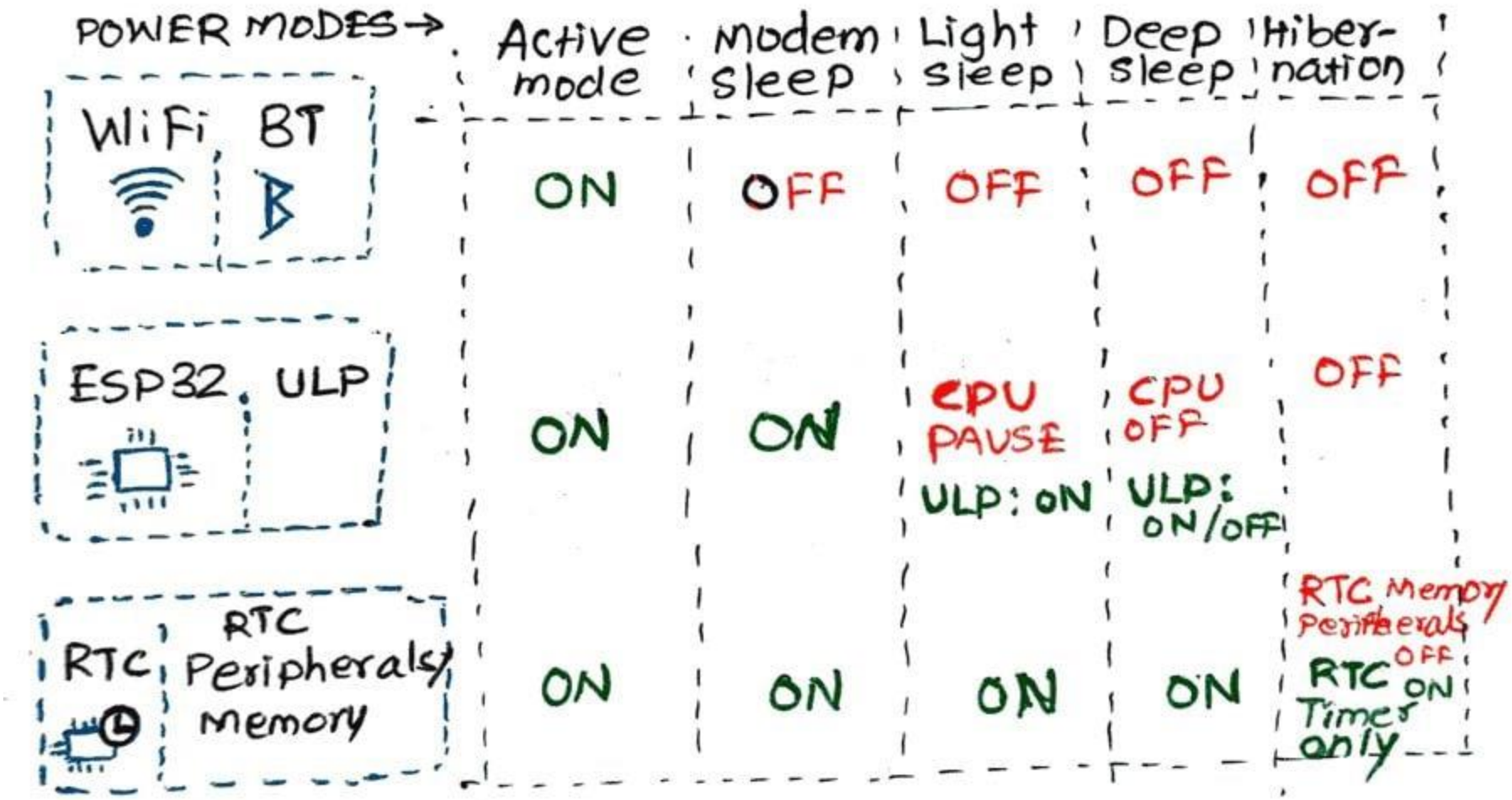
ESP32 Deep Sleep Mode

예) 1000mAh 배터리로 온도 및 습도를 모니터링하고 10분마다 서버로 데이터를 전송하는 애플리케이션





ESP32 Power Mode



Power 소비

Powe Mode	Description	Power Consumption
Active (RF Working)	Wi-Fi and Bluetooth ON	95-240 mA
Modem Sleep	CPU ON	Max Speed: 20mA
		Normal: 5-10mA
		Slow: 3mA
Light Sleep	-	0.8 mA
Deep Sleep	ULP ON	150 uA
	RTC Timer+ RTC Memory	10 uA
Hibernation	RTC timer Only	5 uA
Power Off	-	0.1 uA

Deep Sleep Mode

Application: Sending Data to Server.

Active mode I = 240 mA

Deep sleep mode I = 10uA.

10 second Active Mode

5 mins, 50 sec. Sleep Mode

WITH SLEEP MODE

BATTERY LIFE

249.4 HOURS

All time Active Mode

WITHOUGHT SLEEP MODE

BATTERY LIFE

4.16 HOURS

Deep Sleep Mode

참고: 딥 슬립 모드에서는 CPU와 메모리의 전원이 차단됩니다. ESP32가 깨어날 때 리셋과 함께 재시작됩니다.
이는 각 깨어날때마다 프로그램 실행이 처음부터 시작됨을 의미합니다.

esp_deep_sleep_start() Deepsleep 모드로 진입

RTC memory에 데이터 저장

슬립/리부팅 중에 데이터를 사용해야 하는 경우, RTC_DATA_ATTR 속성을 가진 전역 변수를 정의하여 RTC 메모리에 데이터를 저장

```
RTC_DATA_ATTR int bootCount= 0;
```

Wake Up Source

https://www.electronicwings.com/esp32/esp32-deep-sleep-mode

- **Timer**
- **Touch Pins**
- **External Interrupts (ext0 and ext1)**
- **ULP co-processor(Ultra Low Power)**

ESP32 Deep Sleep Mode: Timer Wake UP

- RTC 컨트롤러에는 내장된 타이머가 있어 정의된 시간이 지난 후 ESP32를 깨울 수 있습니다.
- 시간은 마이크로초 단위로 지정됩니다. 아두이노 IDE에서는 매우 간단하고 직관적입니다.

```
esp_sleep_enable_timer_wakeup(time_in_us)
```

Code

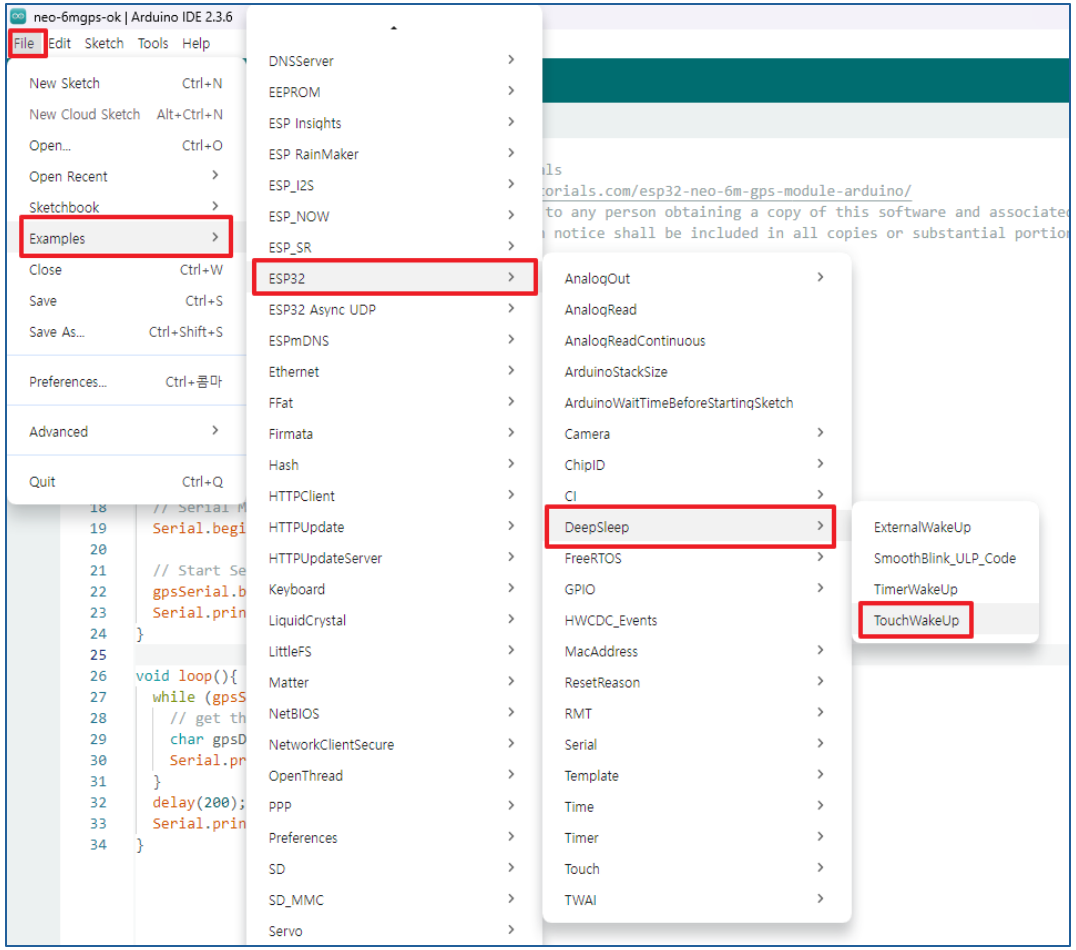
ESP32를 딥 슬립 모드로 전환하고 5초 후 타이머 소스를 사용하여 깨웁니다. 또한 부트 카운트 변수를 RTC 메모리에 저장하고 부팅 횟수를 출력합니다.

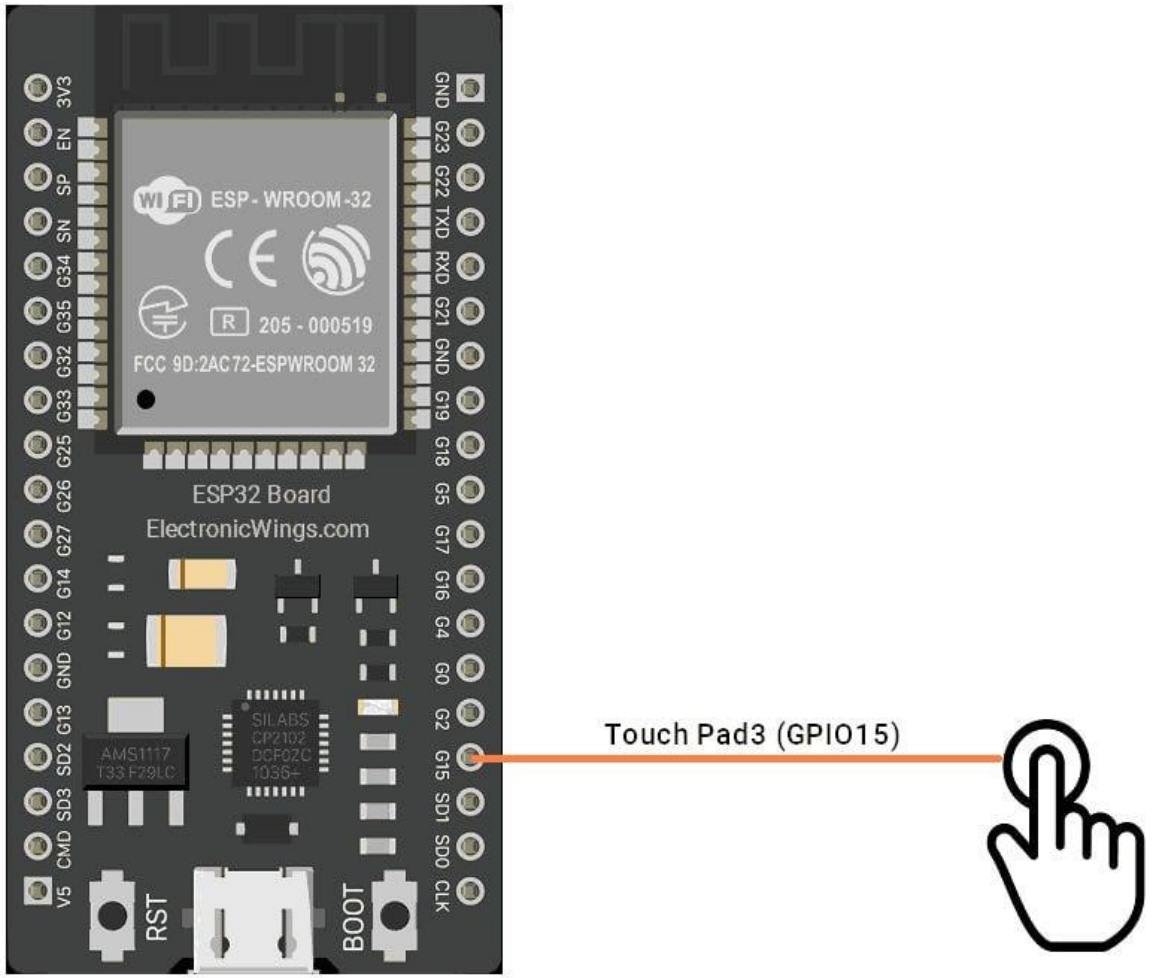
ESP32 Deep Sleep Mode: Touch Pad Wake UP

esp_sleep_enable_touchpad_wakeup()

Code

File>> Example>> ESP32>> Deep Sleep, and open **TouchWakeUP** Sketch.





ESP32 Deep Sleep Mode: External Interrupt Wake Up

RTC 컨트롤러에는 RTC GPIO를 사용하여 웨이크업을 트리거하는 로직이 포함되어 있습니다.

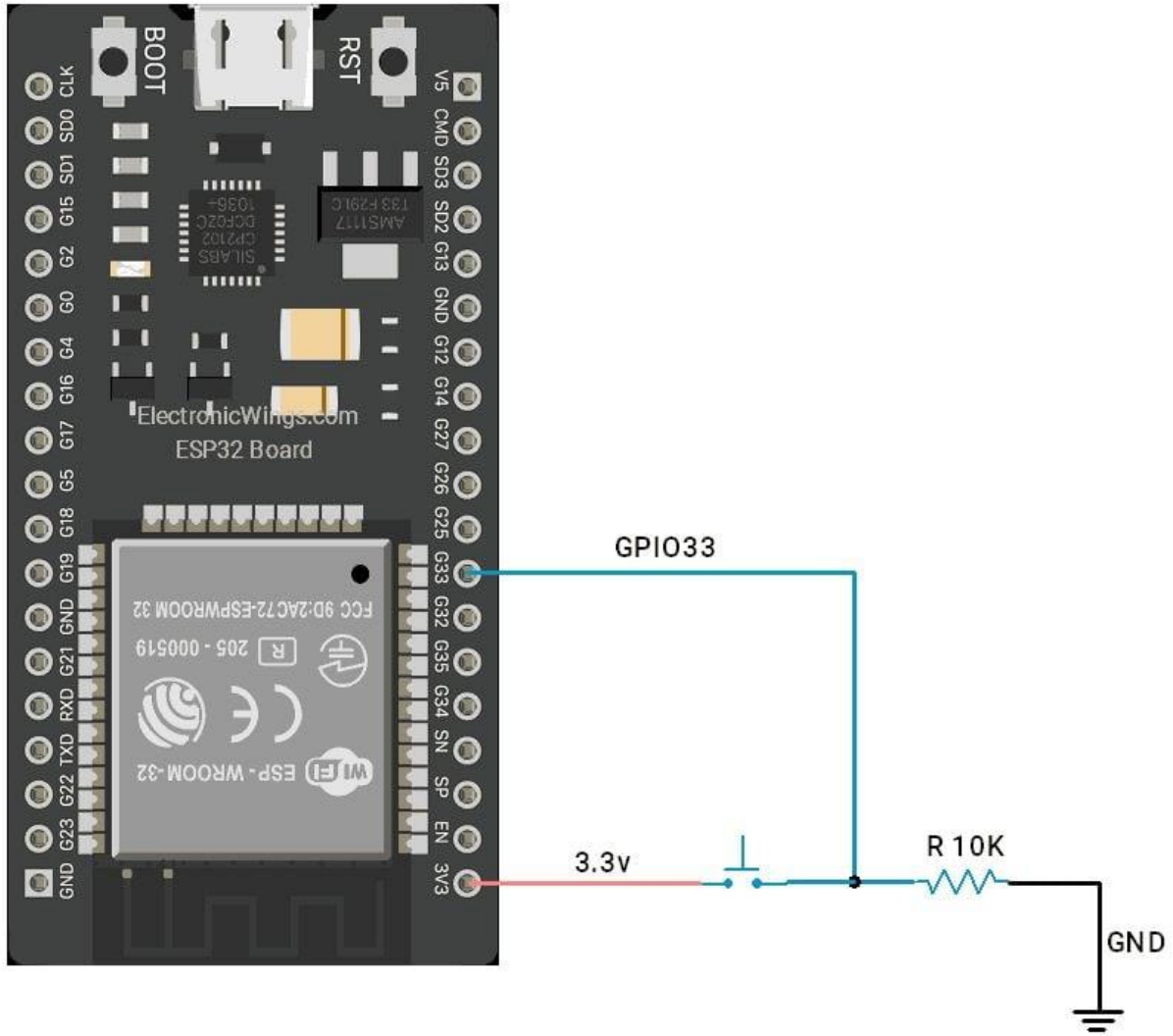
ext0 – 특정 핀 하나로 ESP32를 웨이크업해야 할 때 사용합니다. 모든 RTC GPIO를 사용할 수 있습니다.

ext1 – 웨이크업을 위한 여러 개의 버튼이 있을 때 사용합니다.

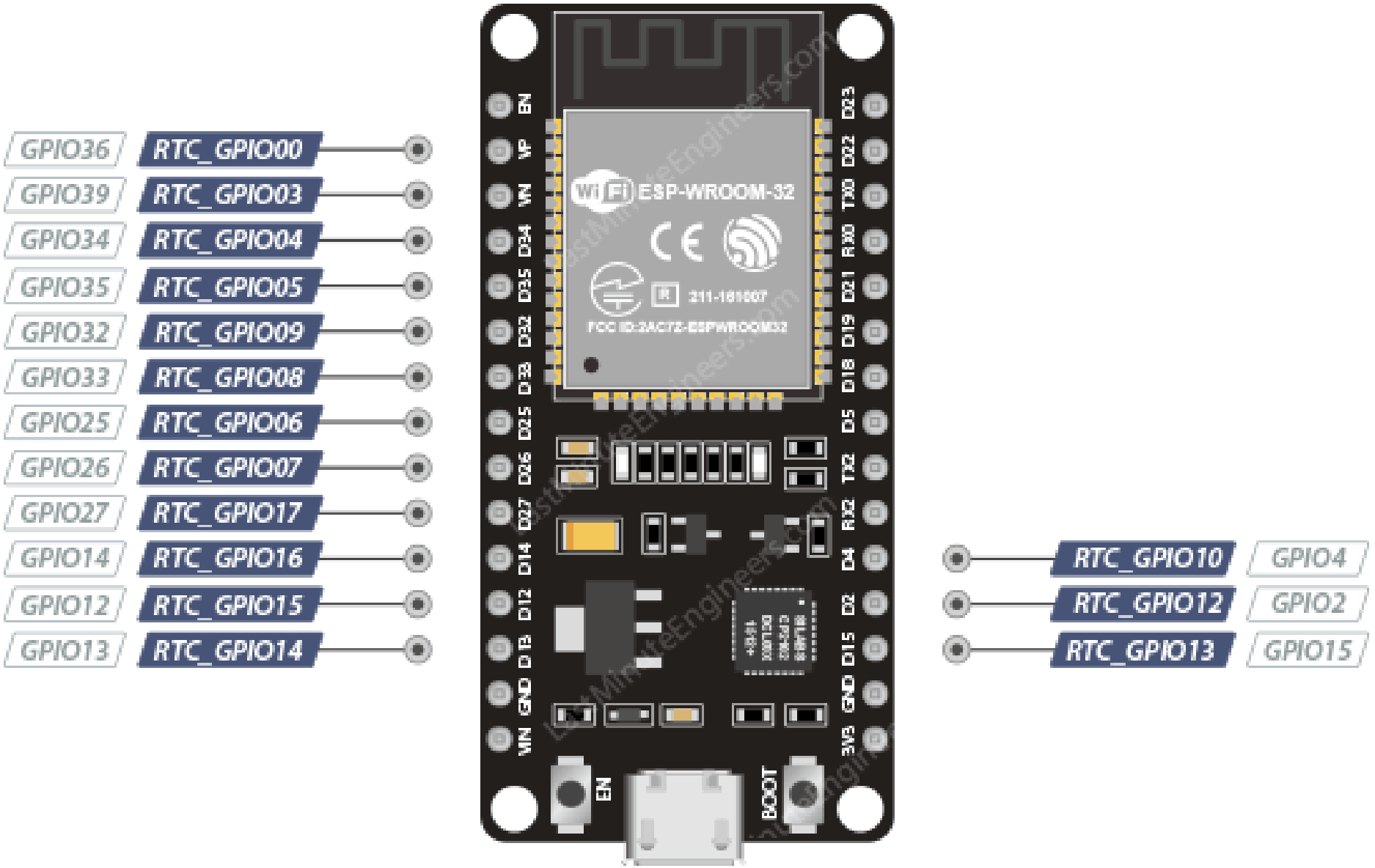
ESP32 wake up using ext0

```
esp_sleep_enable_ext0_wakeup(GPIO_NUM_X, level)

esp_sleep_enable_ext0_wakeup(GPIO_NUM_33,1);
//1 = High, 0 = Low
```

Code
File > Examples > ESP32 > Deep Sleep > ExternalWakeUp:



RTC GPIO Pins



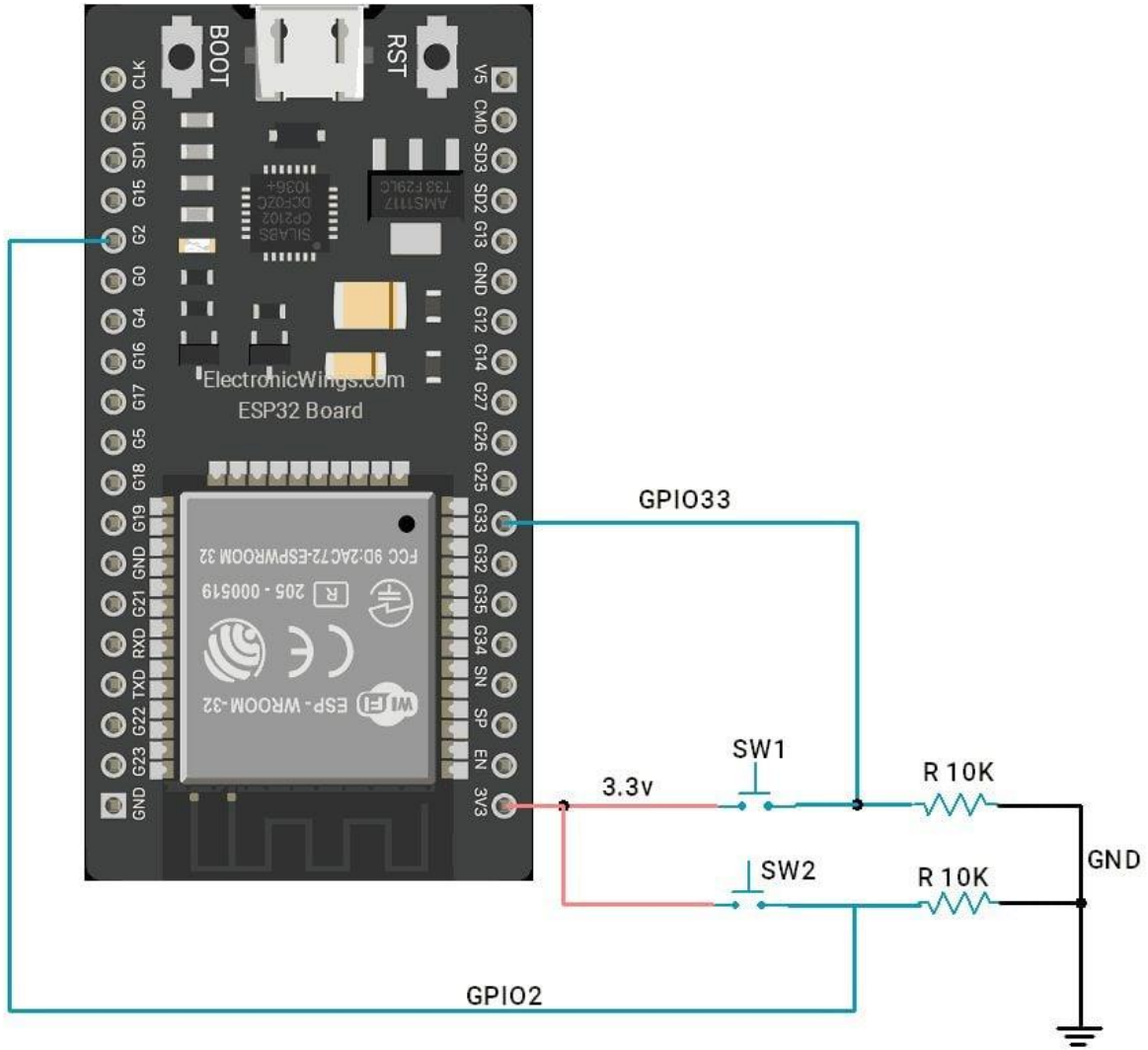
https://lastminuteengineers.com/esp32-pinout-reference/

ESP32 wake up using Ext1

Ext1을 사용하면 여러 개의 버튼으로 ESP32를 깨울 수 있습니다. ext0처럼 단일 핀에만 의존할 필요가 없습니다.

코드는 유사합니다. esp_sleep_enable_ext0_wakeup() 함수 대신 esp_sleep_enable_ext1_wakeup() 함수를 사용해야 합니다.

```
esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,ESP_EXT1_WAKEUP_ANY_HIGH);
```



ESP32 Timer Interrupt

타이머

- ESP32에는 두 개의 타이머 그룹이 있습니다.
- 각 그룹에는 두 개의 범용 하드웨어 타이머가 있어, 총 4개의 타이머가 있습니다.
- 모든 타이머는 64비트 카운터와 16비트 프리스케일러 (분할기)를 기반으로 합니다.
- 프리스케일러는 기본 클럭 주파수(보통 80MHz)를 분할하는 데 사용됩니다.
- 나누어진 주파수는 타이머 카운터에 인크리먼트 또는 디크리먼트용으로 제공됩니다.
- 프리스케일러는 16비트이므로 80MHz 기본 클럭 주파수를 2부터 65536까지 나눌 수 있어 매우 유연합니다.
- 타이머는 카운트 업(UP) 또는 카운트 다운(DOWN)으로 구성할 수 있습니다.

알람 생성 (ESP32의 타이머 인터럽트)

- 타이머 값이 설정값과 일치할 때 타이머가 알람을 발생시킬 수 있습니다.
- 이 경우 타이머가 자동으로 0으로 재설정되며, 인터럽트가 활성화된 상태라면 인터럽트를 발생시킵니다.

Programming Timer Interrupt(Alarm Generation)

- **Step1: initialize timerBegin() Functione**

  타이머 전역 변수 timer= timerBegin(0, 80, true);

  - 타이머 번호 0 ~ 3
  - 클럭 / 프리스케일러 : 80Mhz/80 = 1MHz, 1 microsecond
  - Up(true) Down(false) 플래그

- **Step2: Attach Interrupt**

  timerAttachInterrupt(timer, &onTimer, true);

  - 인터럽트 ISR 함수에 연결 onTimer() 함수 실행
  - 전역변수 타이머의 포인터
  - 함수 호출 주소

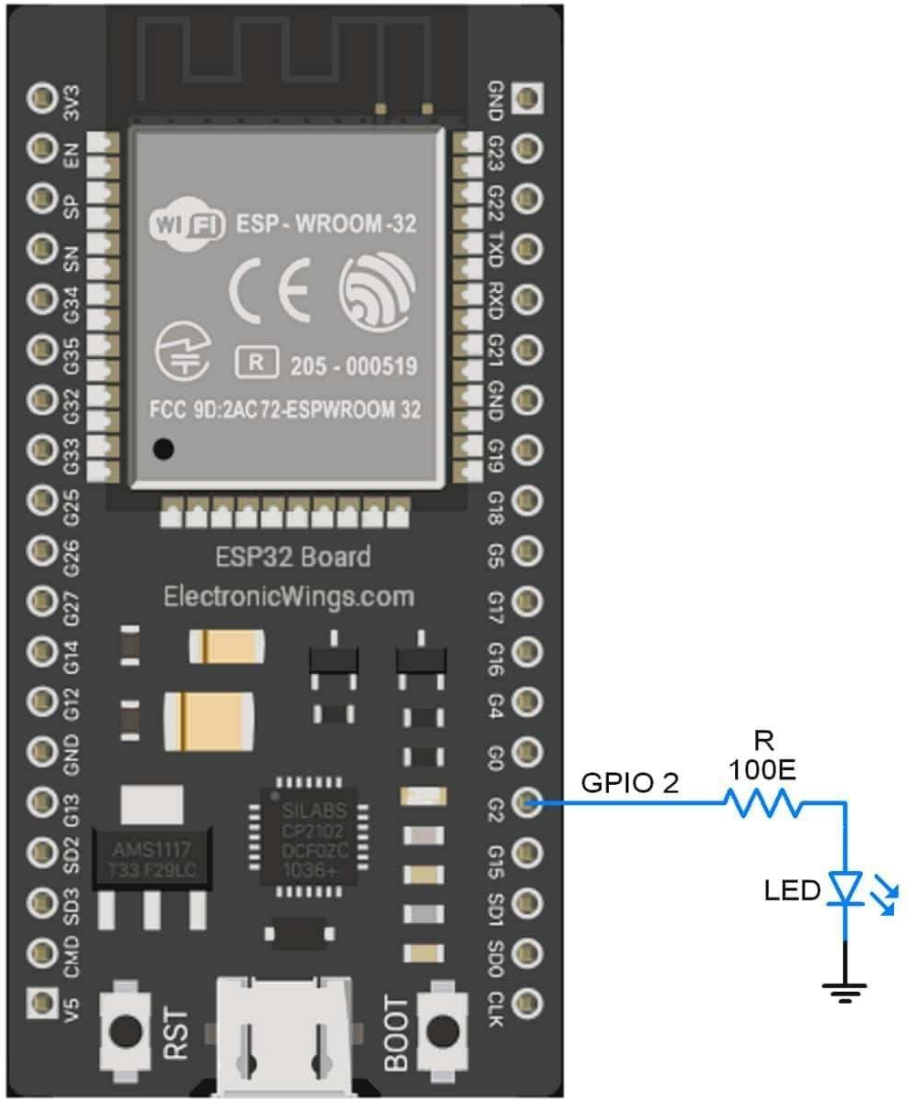
- **Step3: Define the match timer value**

  timerAlarmWrite(timer, 1000000, true);

  - 인터럽트를 발생시킬 값을 정의
  - 1초 지연을 위해 1000000 (500 ms= 500000, 10ms=10000, 0.8s=800000).
  - 자동으로 '0'으로 재설정되며 인터럽트 ISR 함수가 실행을 시작합니다.
  - 먼저 timerAlarmEnable(timer); 함수로 인터럽트/알람을 활성화해야 합니다

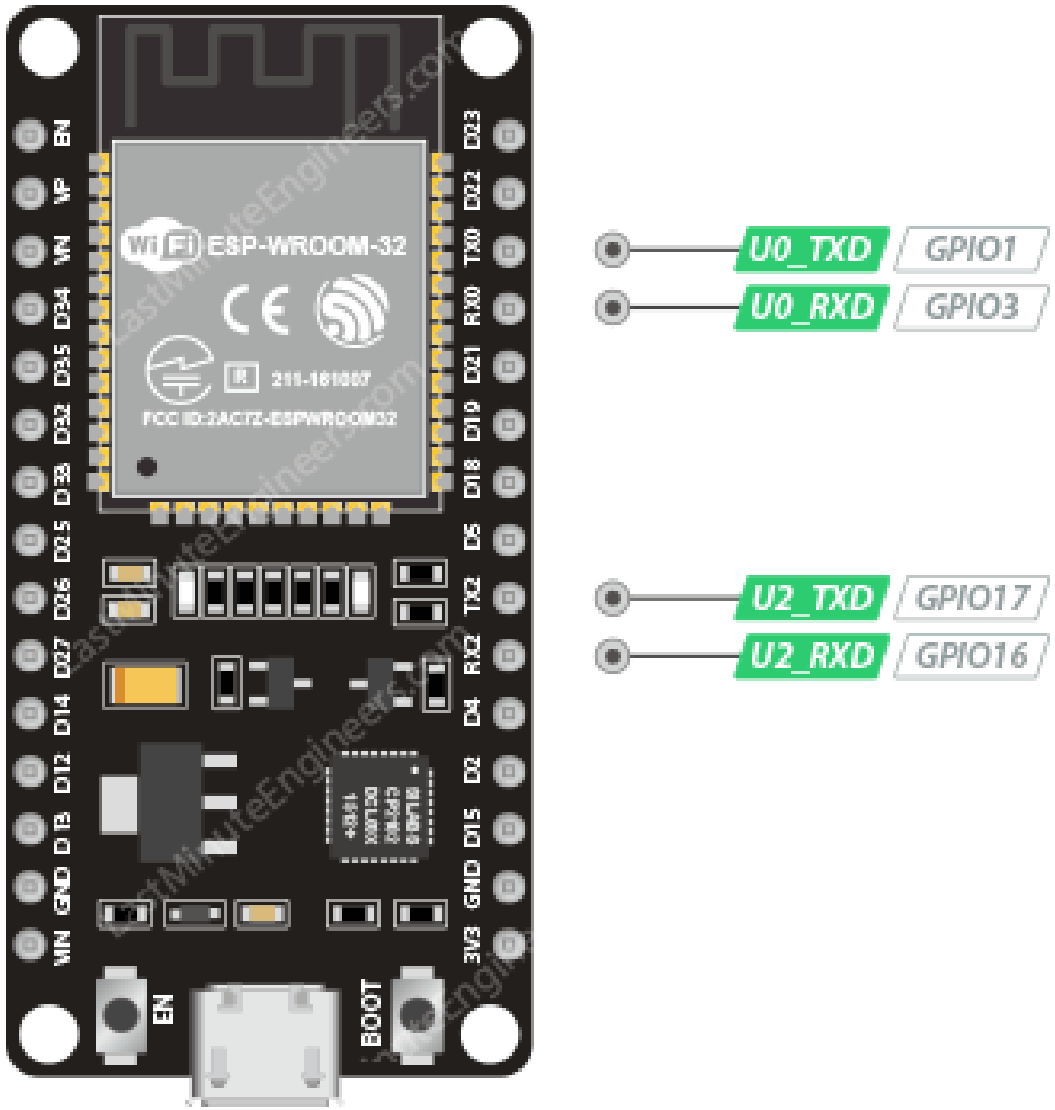
Code

```
#define LED_PIN 2

volatile int interruptCounter;  //for counting interrupt
.
.
void IRAM_ATTR onTimer() {     //Defining Inerrupt function with IRAM_ATTR for faster access
 portENTER_CRITICAL_ISR(&timerMux);
 interruptCounter++;
 portEXIT_CRITICAL_ISR(&timerMux);
}

void setup() {
 timer = timerBegin(0, 80, true);                  // timer 0, prescalar: 80, UP counting
 timerAttachInterrupt(timer, &onTimer, true);      // Attach interrupt
 timerAlarmWrite(timer, 1000000, true);            // Match value= 1000000 for 1 sec. delay.
 timerAlarmEnable(timer);                          // Enable Timer with interrupt (Alarm Enable)
}

void loop() {
 if (interruptCounter > 0) {
    portENTER_CRITICAL(&timerMux);
   interruptCounter--;
   portEXIT_CRITICAL(&timerMux);

   totalInterruptCounter++;              //counting total interrupt
.
.
```

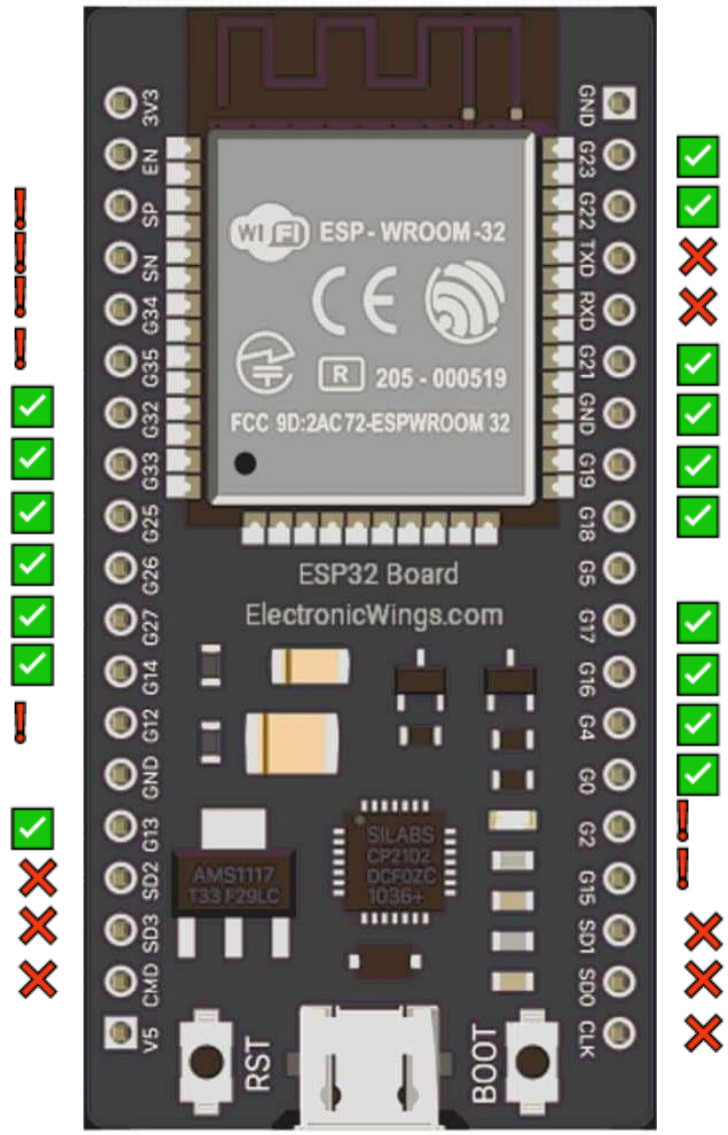


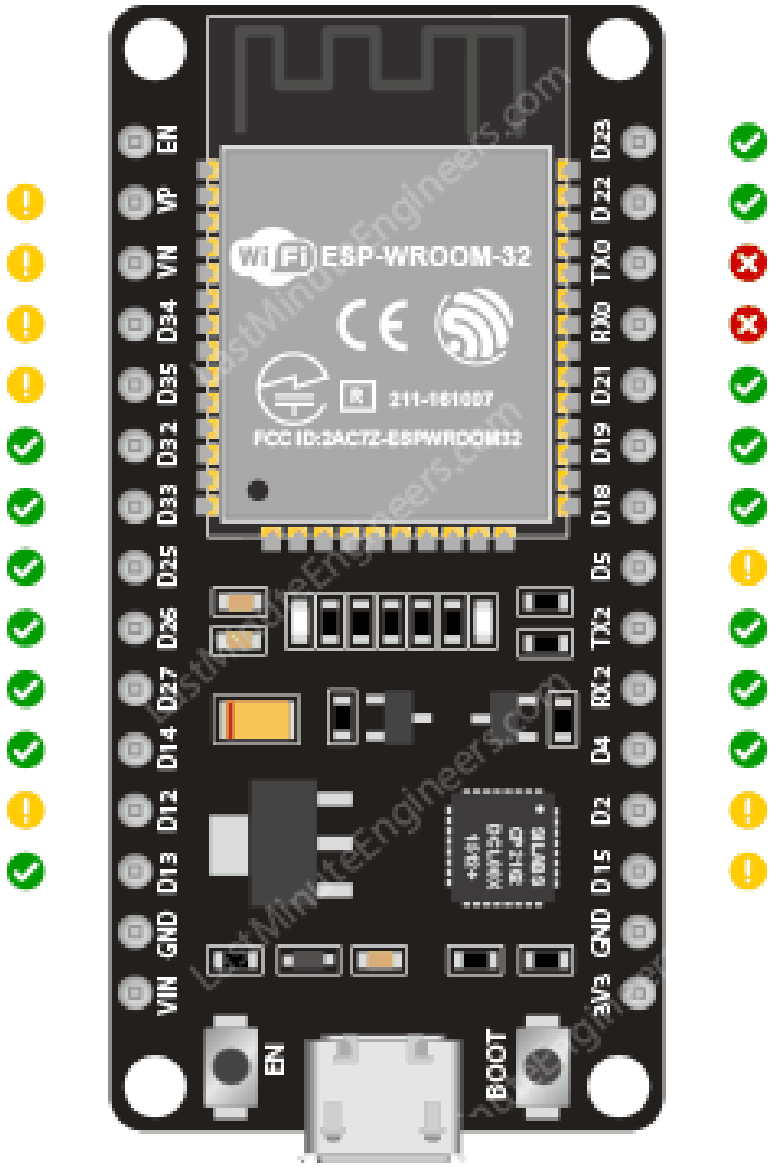
UART

- ESP32 개발 보드에는 UART0, UART1, UART2 세 개의 UART 인터페이스가 있으며, 비동기 통신(RS232 및 RS485)과 최대 5Mbps의 IrDA를 지원합니다.
- UART0 핀은 USB-시리얼 변환기에 연결되어 플래싱 및 디버깅에 사용됩니다. 따라서 UART0 핀 사용은 권장되지 않습니다.
- UART1 핀은 내장 플래시 메모리 칩 전용으로 예약되어 있습니다.
- 반면 UART2는 GPS, 지문 센서, 거리 센서 등과 같은 UART 장치 연결에 안전한 옵션입니다.
- CTS 및 RTS 신호의 하드웨어 관리와 소프트웨어 흐름 제어(XON 및 XOFF)도 제공합니다.

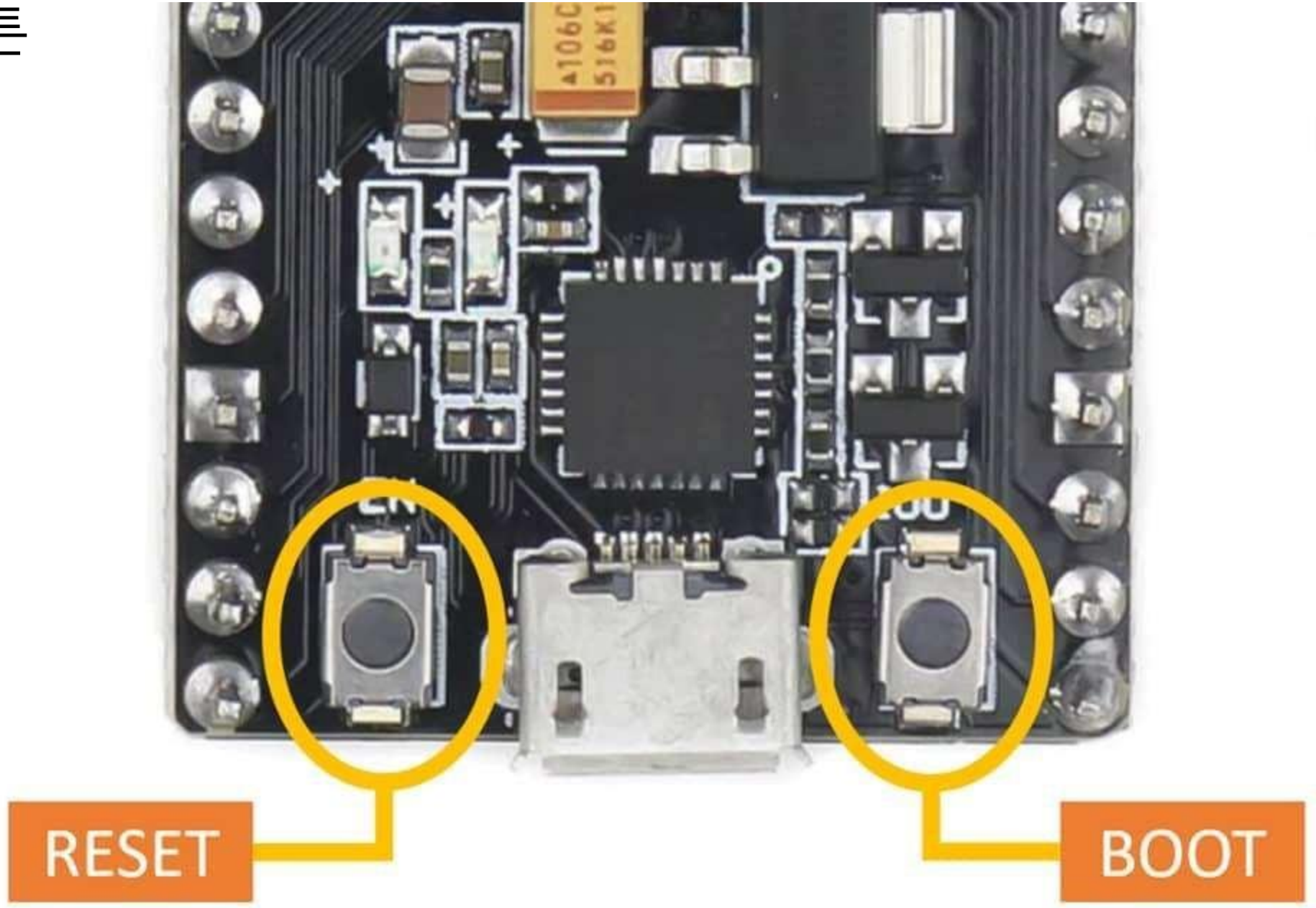


설계할 때 주의점

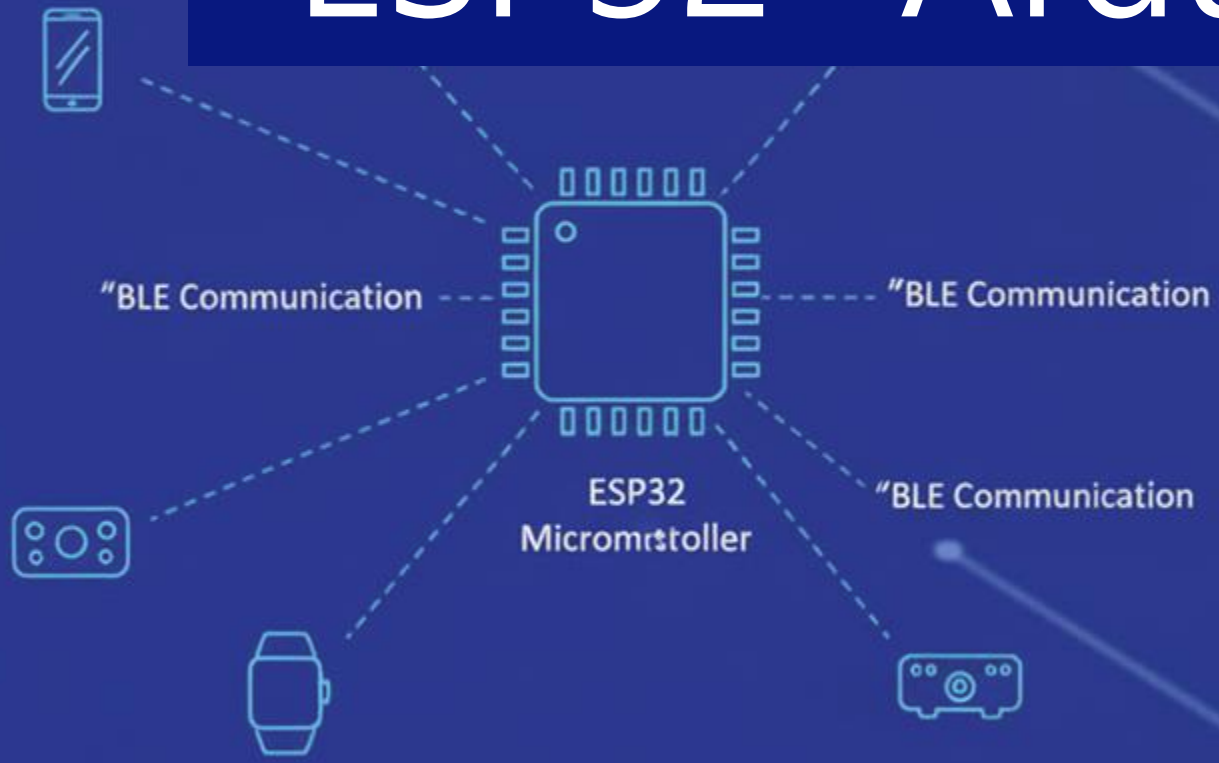

ESP - WROOM -32
205 - 000519
FCC 9D:2AC72-ESPWROOM 32
ESP32 Board
ElectronicWings.com
AMS1117
SILABS
CP2102
RST
BOOT


ESP-WROOM-32
EN
BOOT

Boot 버튼



ESP32 Arduino IDE



Care-lab.kr

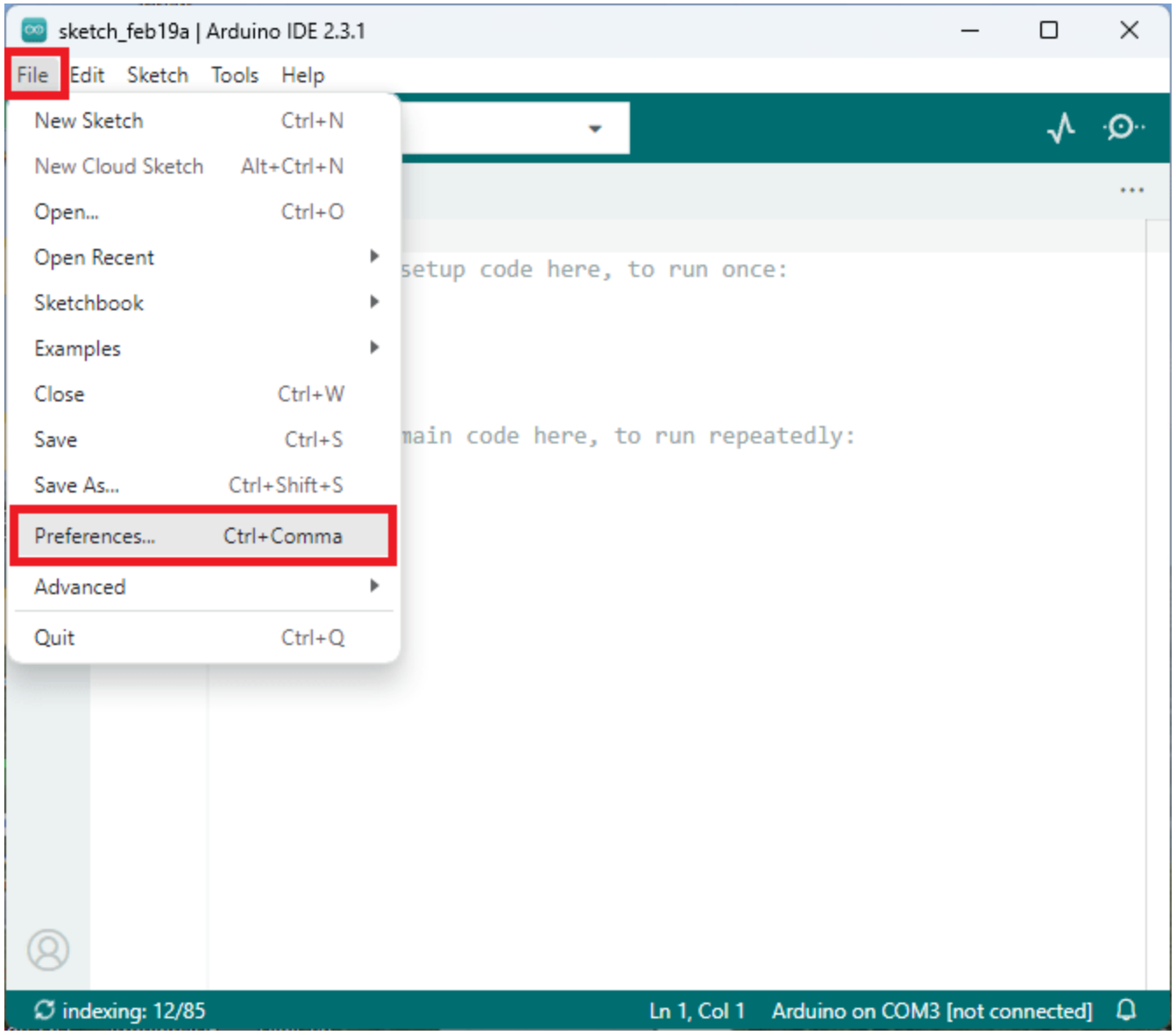
Arduino IDE 설치

Download

https://www.arduino.cc/en/software/

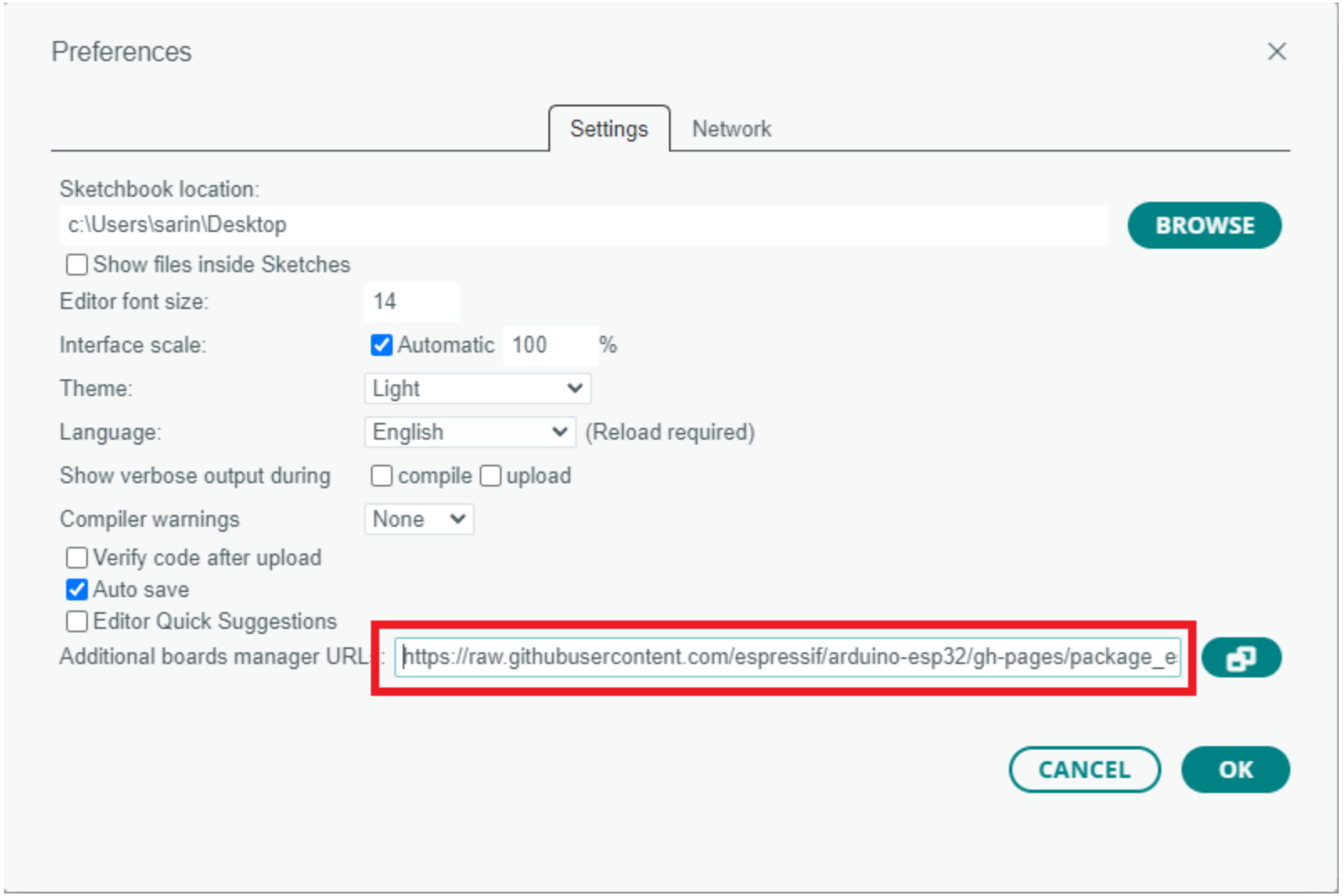
Arduino IDE ESP32 보드 설치

File - Preference

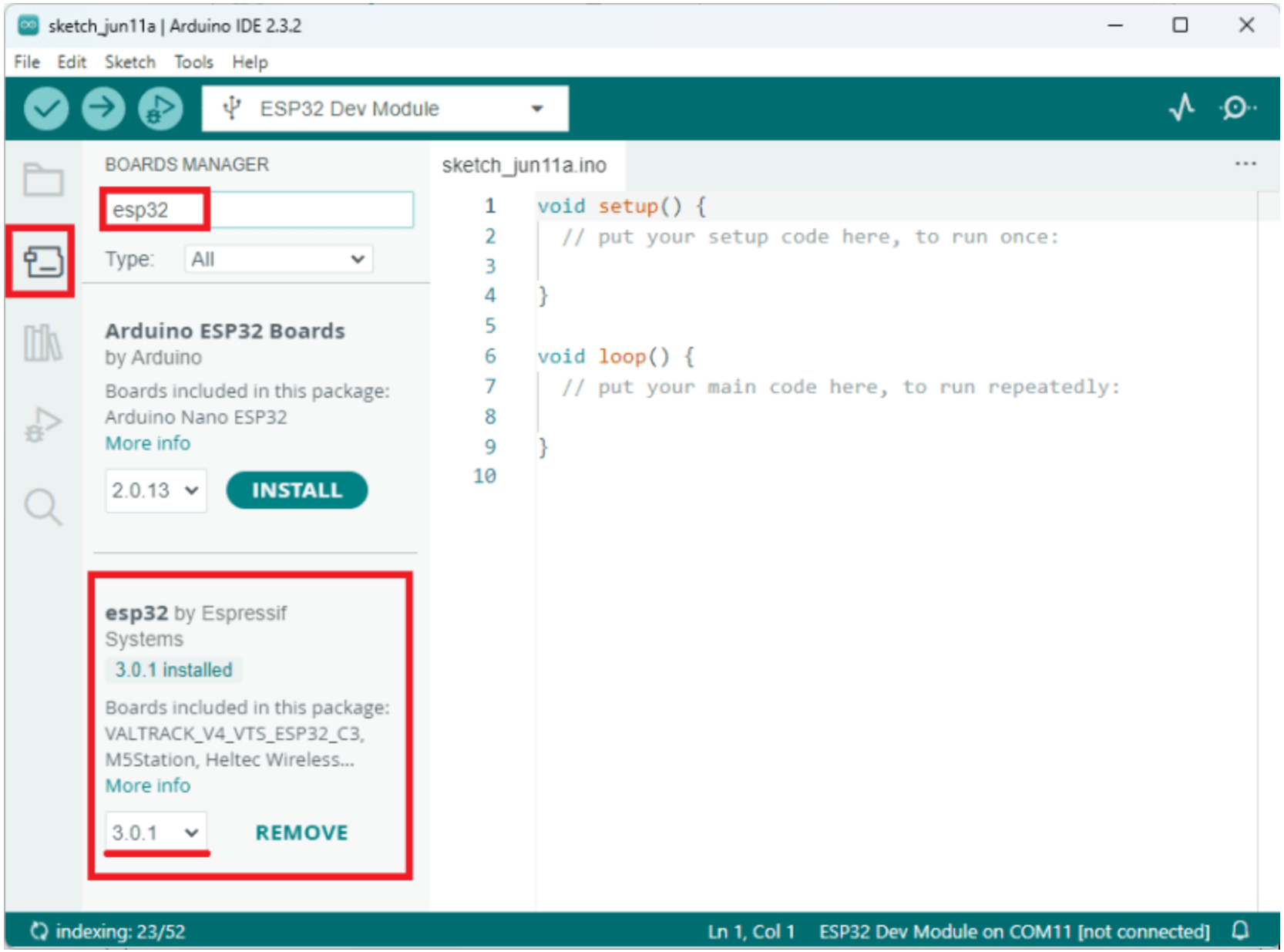


Arduino IDE ESP32 보드 설치

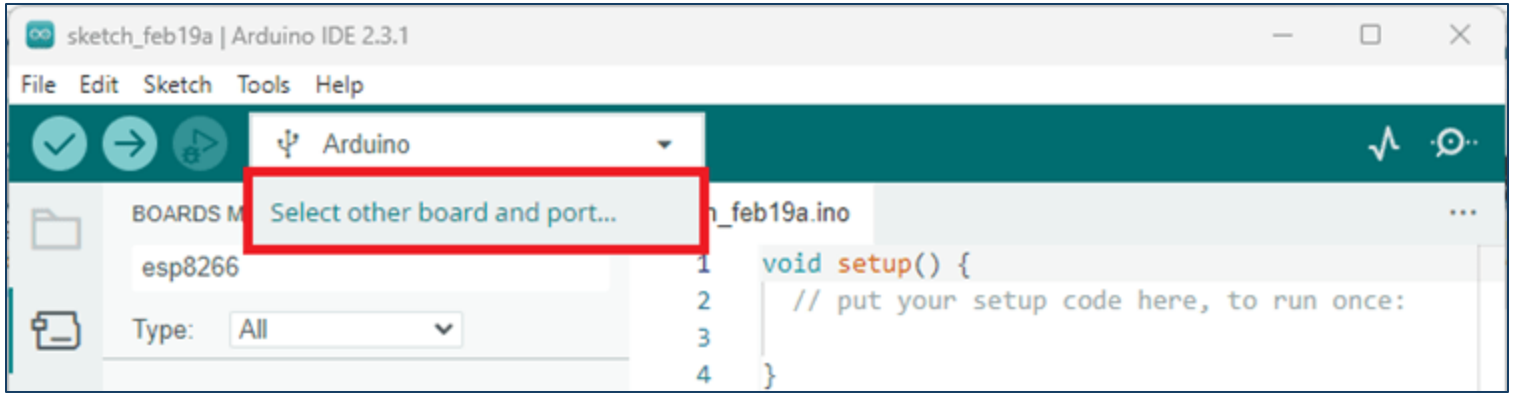
붙여넣기
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

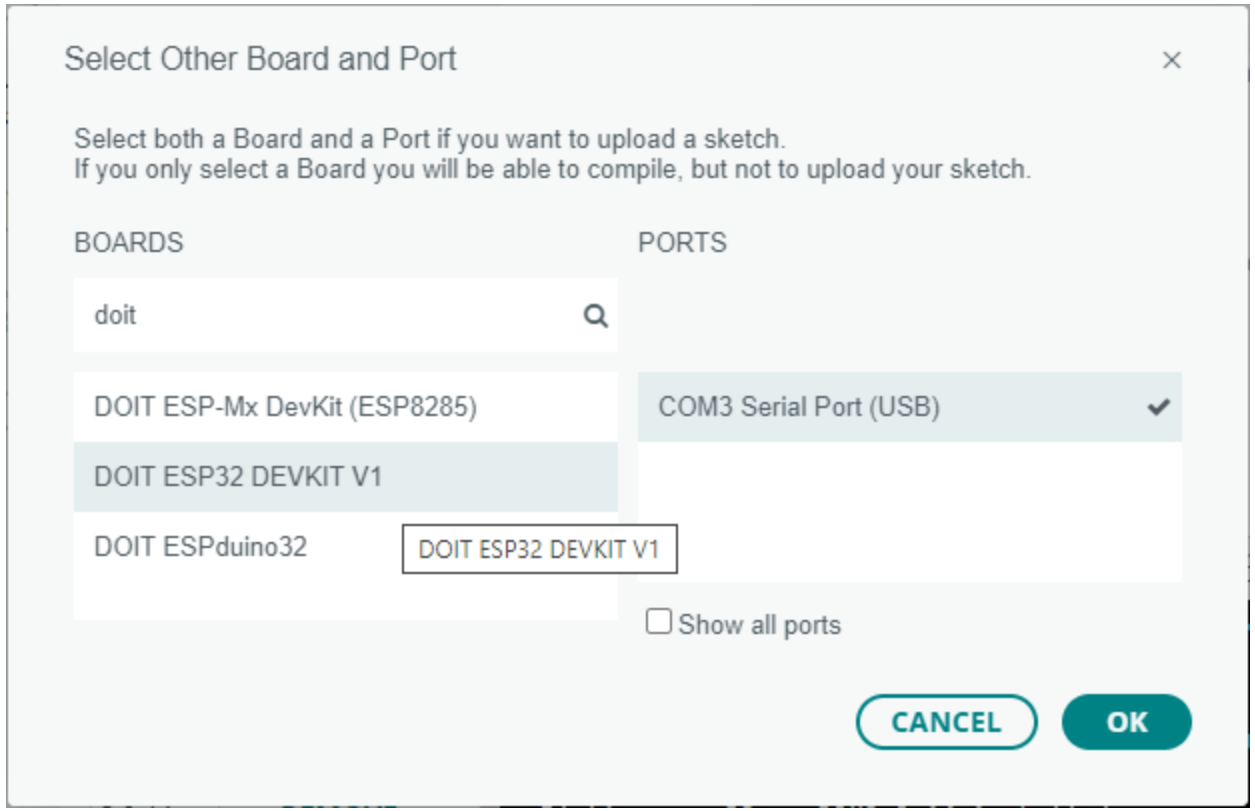


보드 관리자 ESP32 설치



스케치 업로드


sketch_feb19a | Arduino IDE 2.3.1
File Edit Sketch Tools Help
Arduino
Select other board and port...
BOARDS M
esp8266
Type: All
_feb19a.ino
1 void setup() {
2 // put your setup code here, to run once:
3
4 }


Select Other Board and Port
Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.
BOARDS
PORTS
doit
DOIT ESP-Mx DevKit (ESP8285)
DOIT ESP32 DEVKIT V1
DOIT ESPduino32
DOIT ESP32 DEVKIT V1
COM3 Serial Port (USB)
Show all ports
CANCEL
OK

업로드 완료 – 자동 Reset - 실행

sketch_feb20a | Arduino IDE 2.3.1

File Edit Sketch Tools Help

DOIT ESP32 DEVKIT V1

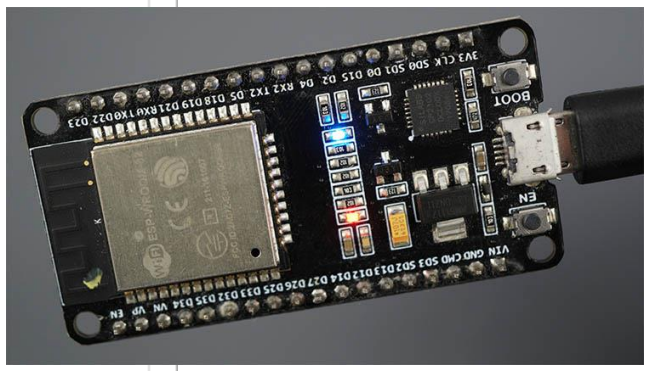
sketch_feb20a.ino

```
11    // put your setup code here, to run once:
12    Serial.begin(115200);
13    pinMode(LED, OUTPUT);
14  }
15
16  void loop() {
17    // put your main code here, to run repeatedly:
18    digitalWrite(LED, HIGH);
19    Serial.println("LED is on");
20    delay(1000);
21    digitalWrite(LED, LOW);
22    Serial.println("LED is off");
23    delay(1000);
24  }
```

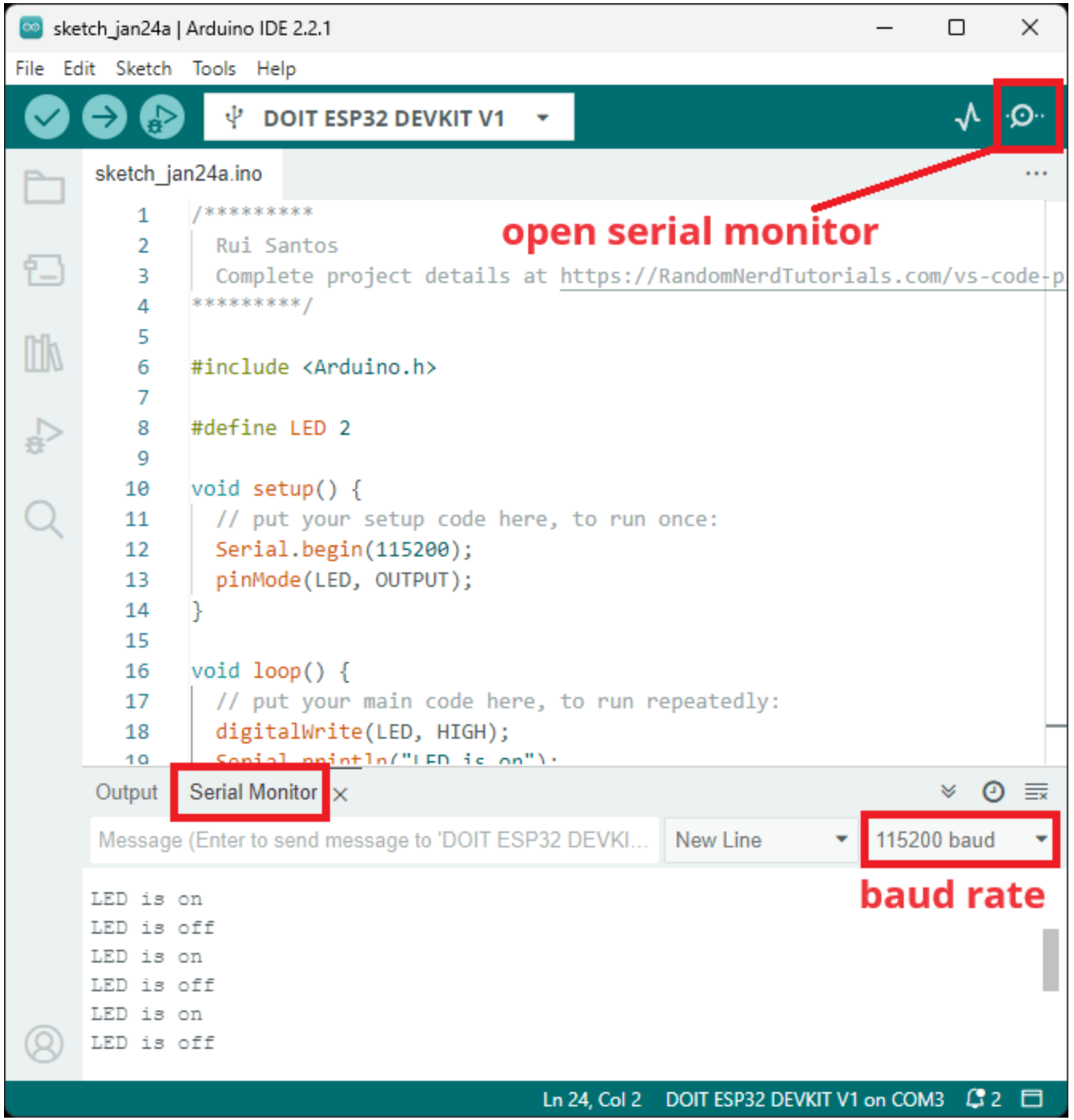
Output

```
Writing at 0x0004b07d... (100 %)
Wrote 262688 bytes (145826 compressed) at 0x00010000 in 2.4 seconds (effective
Hash of data verified.

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

Ln 24, Col 2 DOIT ESP32 DEVKIT V1 on COM3 3



Serial 모니터


sketch_jan24a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
DOIT ESP32 DEVKIT V1
sketch_jan24a.ino
open serial monitor
/*********
Rui Santos
Complete project details at https://RandomNerdTutorials.com/vs-code-p
*********/
#include <Arduino.h>
#define LED 2
void setup() {
// put your setup code here, to run once:
Serial.begin(115200);
pinMode(LED, OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(LED, HIGH);
Output
Serial Monitor
Message (Enter to send message to 'DOIT ESP32 DEVKI...
New Line
115200 baud
baud rate
LED is on
LED is off
LED is on
LED is off
LED is on
LED is off
Ln 24, Col 2
DOIT ESP32 DEVKIT V1 on COM3

IDE 기능 설명


neo_m8ntinyplus | Arduino IDE 2.3.6
제목표시
File Edit Sketch Tools Help
ESP32 Dev Module
업로드
보드와 컴포트 설정
플로터/씨리얼모니터
neo_m8ntinyplus.ino
보드매니저
라이브러리 매니저
/*********
Rui Santos & Sara Santos - Random Nerd Tutorials
Complete instructions at https://RandomNerdTutorials.com/esp32-neo-6m-gps-module-arduino/
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation f
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Softwar
// Define the RX and TX pins for Serial 2
#define RXD2 16
#define TXD2 17
#define GPS_BAUD 9600
// Create an instance of the HardwareSerial class for Serial 2
HardwareSerial gpsSerial(2);
스크롤 토글/시간표시/화면 클리어
Serial Monitor
씨리얼통신 데이터 입력
New Line
115200 baud
SPEED (km/h) = 0.70
ALT (min)= 6.40
HDOP = 1.61
Satellites = 7
Time in UTC: 2025/12/6,13:41:52
씨리얼 모니터 / 컴파일 업로드 실행 상태 표시
Ln 1, Col 1 ESP32 Dev Module on COM3

Trouble Shooting

- ESP32 보드를 인식하지 못하시나요?

  USB 케이블 연결을 제대로 확인하세요(때로는 Micro USB 케이블이 느슨하게 연결될 수 있습니다).
  장치 관리자에서 USB 드라이버가 설치되어 있는지 확인하세요.
  PC 또는 노트북 USB 포트가 작동하는지 확인하세요.
  ESP32의 전원 공급 장치를 확인하세요.
  ESP32 보드가 작동하는지 확인하세요.

- ESP32 보드가 인식하지만 프로그래밍할 수 없나요?

  장치 관리자와 Arduino IDE에서 ESP32의 올바른 COM 포트가 동일한지 확인하세요.
  선택한 보드가 ESP32인지 확인하세요.

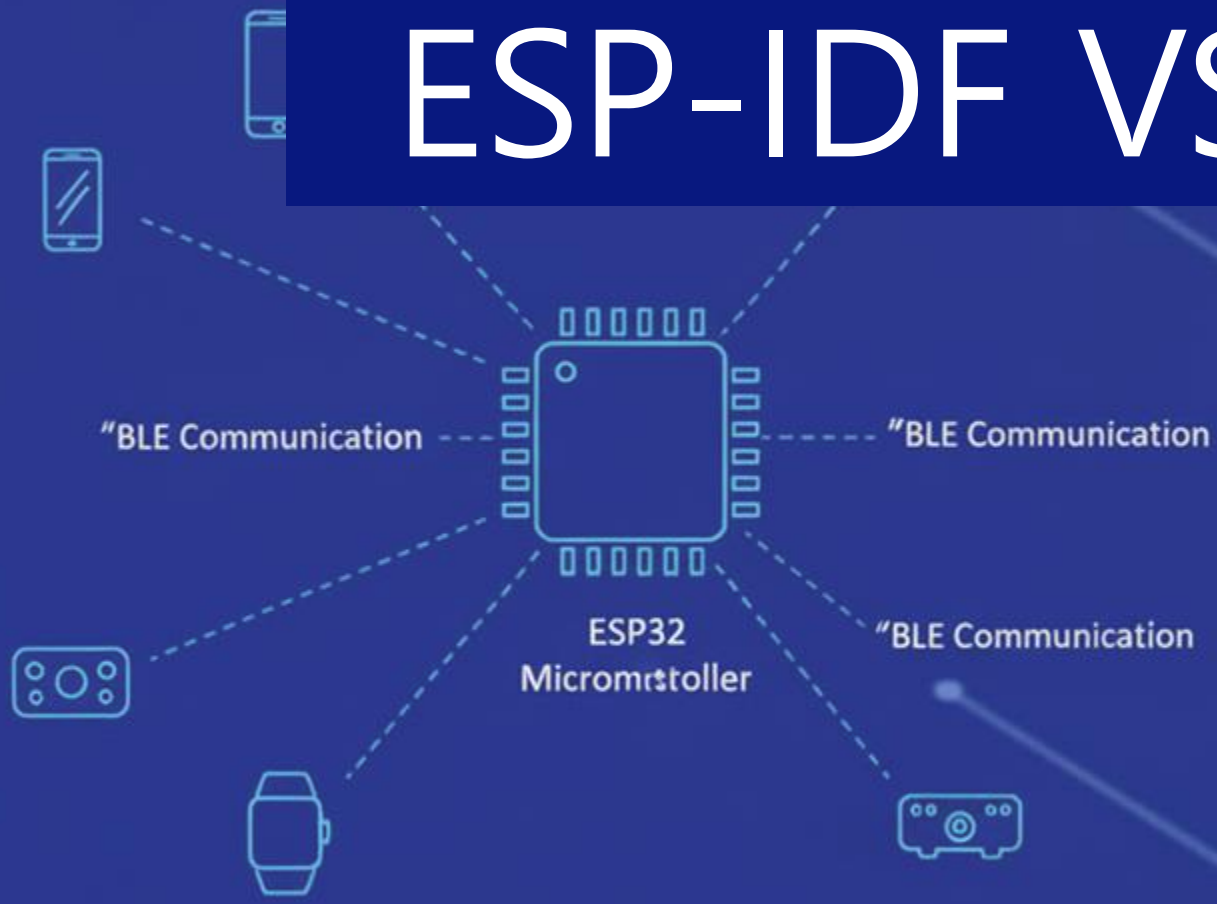
- ESP32 보드 설정은 정상인데도 프로그래밍할 수 없나요?

  ESP32 보드가 부팅 모드에 있는지 확인하세요. 부팅 모드에 진입해도 버튼이 너무 작아서 부팅 모드로 진입하지 못하는 경우가 있습니다.
  ESP32에서 USB 케이블을 제거했다가 ESP32 보드를 다시 연결하고 부팅 모드로 설정하세요.

- ESP32 칩에 USB 케이블을 연결하면 갑자기 열이 발생합니다.

  보드 내부 또는 외부 단락이 발생하면 이 문제가 발생합니다.
  공급 전압을 확인하세요. 공급 전압이 정격 전압보다 높은 경우에도 이 문제가 발생합니다.

ESP-IDF VS Code



Care-lab.kr

Arduino vs ESP-IDF 환경

ESP-IDF

ARDUINO IDE

특징

- 기본 FreeRTOS 지원 : 작업 생성, 우선순위 설정, 큐 및 세마포어 사용.
- 전체 하드웨어 제어 : 정확한 타이밍과 구성으로 모든 주변 장치에 액세스합니다.
- 더 나은 성능 : 더 작은 바이너리, 최적화된 런타임, 더 나은 메모리 관리.
- 전문가 환경 : VS Code, Eclipse와 호환되며 JTAG 디버깅을 지원합니다.

한계

- 학습 곡선이 가파릅니다. 오랫동안 훈련해야 합니다. 초보자는 API가 복잡하다고 느낄 수 있습니다.
- 설치 복잡성 : Python, Git, 툴체인, 환경 변수를 설치해야 합니다.
- 플러그 앤 플레이 라이브러리 감소 : 많은 Arduino 라이브러리를 ESP-IDF로 포팅해야 합니다.

특징

- 간소화된 설정 : Arduino IDE를 설치하고 ESP32 보드 관리자 URL을 추가하면 프로그래밍 준비가 완료됩니다.
- 광범위한 라이브러리 지원 : 센서, 디스플레이, Wi-Fi, BLE 등을 위한 수천 개의 기성 라이브러리.
- 익숙한 구문 : Arduino에 대해 알고 있다면 ESP32로 빠르게 전환할 수 있습니다.
- 빠른 프로토타입 제작 : 소규모 IoT 프로젝트, LED 깜박임, 센서 및 기본 자동화에 적합합니다.

한계

- 제한된 FreeRTOS 제어 : ESP32는 내부적으로 FreeRTOS를 실행하지만 Arduino는 대부분을 추상화합니다.
- 성능 최적화 낮음 : ESP-IDF에 비해 컴파일 시간이 느리고 바이너리 크기가 약간 더 큽니다.
- 하위 제어 : 하드웨어와 주변 장치는 라이브러리를 통해 접근되므로 저수준 최적화가 제한됩니다.

https://fishpoint.tistory.com/12397

Arduino vs ESP-IDF Blink Code

ESP-IDF

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

#define LED_PIN 2

void app_main(void)
{
    gpio_reset_pin(LED_PIN);
    gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);

    while(1) {
        gpio_set_level(LED_PIN, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(LED_PIN, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

ARDUINO IDE

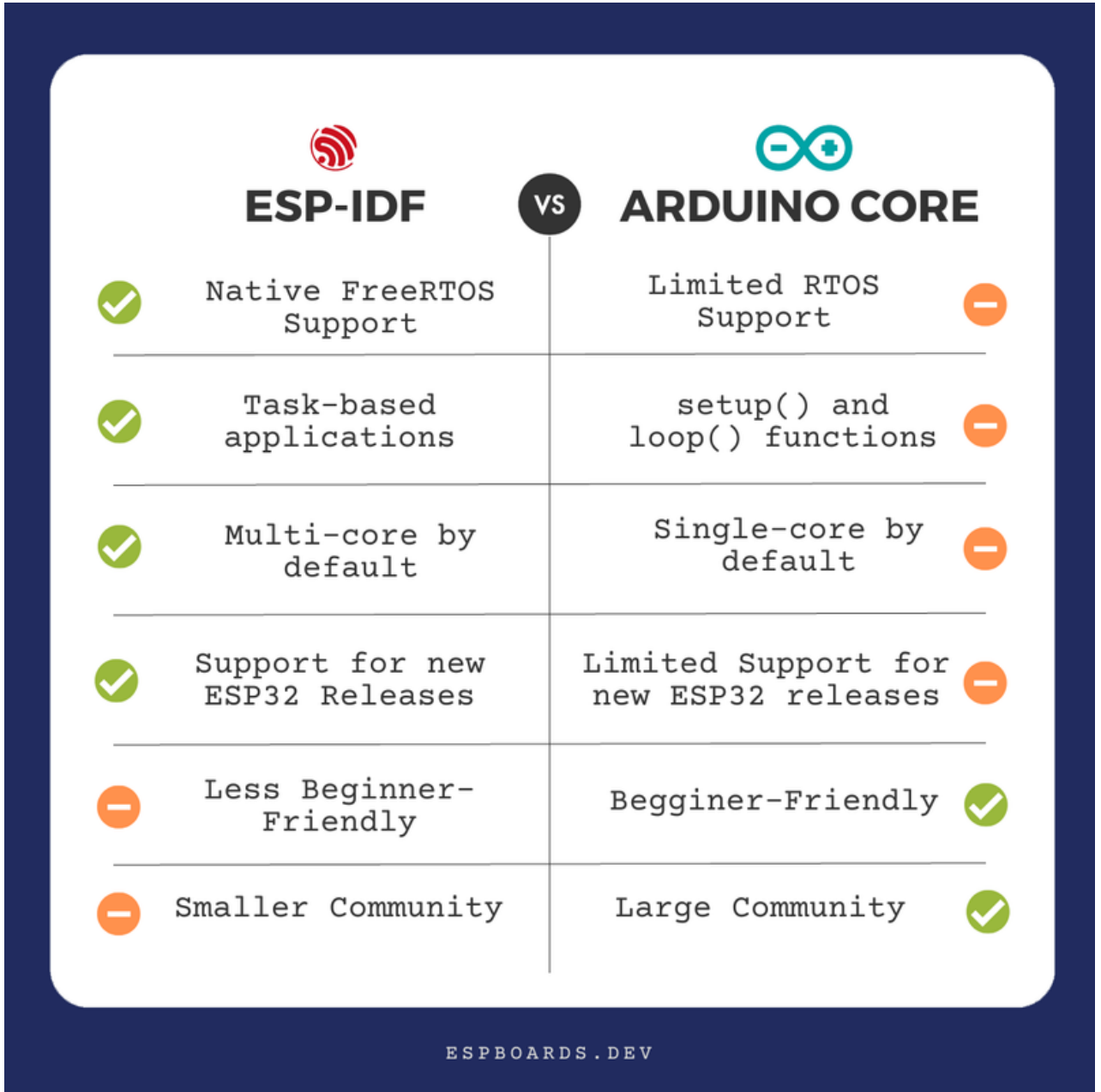
```
#define LED_PIN 2

void setup() {
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_PIN, HIGH);
  delay(1000);
  digitalWrite(LED_PIN, LOW);
  delay(1000);
}
```

Arduino vs ESP-IDF 비교

Feature	Arduino IDE	ESP-IDF
Ease of Use	Very easy	Moderate to complex
Setup	Minimal	Complex (Python, Git, toolchain)
Hardware Control	Limited	Full access
FreeRTOS Tasks	Hidden	Full control
Libraries	Thousands ready-to-use	Must port or use native APIs
Debugging	Serial monitor	Serial + JTAG, breakpoints
Performance	Good for small projects	Optimized, professional grade
Target User	Hobbyists, beginners	Embedded engineers, researchers

Arduino vs ESP-IDF 비교

https://blog.naver.com/nanotoly/223322065269



툴은 툴, 규칙은 규칙

ESP-IDF

ARDUINO IDE

다음의 경우 ESP-IDF를 사용하세요:

- 당신은 전문적인 프로젝트나 대규모 프로젝트를 진행하고 있습니다.
- FreeRTOS 작업과 우선순위를 완벽하게 제어하고 싶습니다.
- 최적화된 성능과 정확한 타이밍이 필요합니다 .
- 당신은 임베디드 산업이나 연구 분야에서 일할 계획입니다.

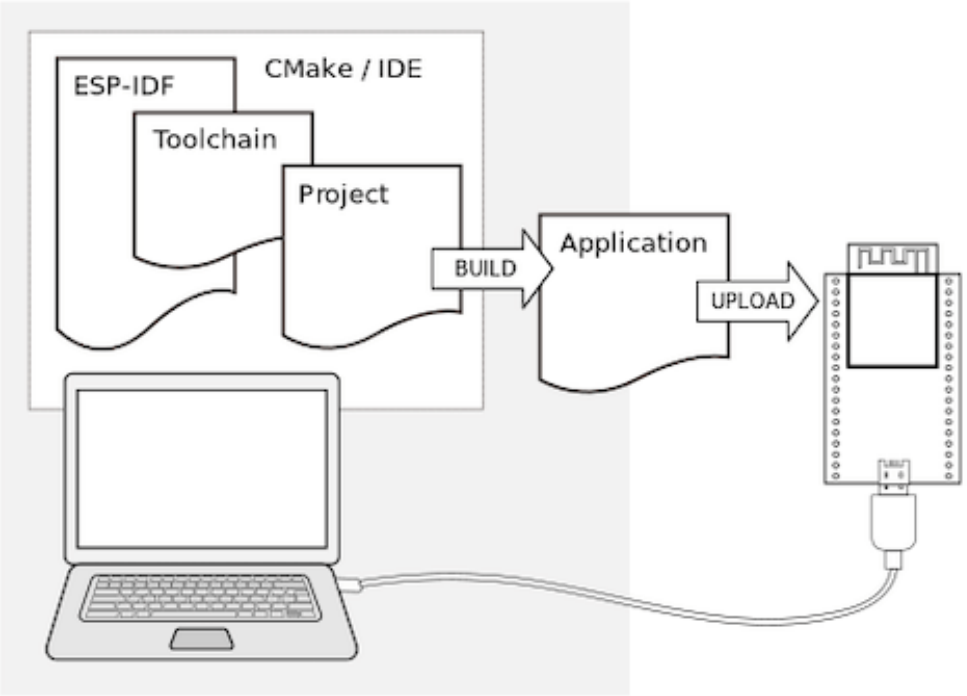
다음과 같은 경우 Arduino IDE를 사용하세요.

- 당신은 임베디드 시스템의 초보자입니다.
- 센서, LED 또는 IoT 프로젝트의 프로토타입을 빠르게 제작 하고 싶습니다.
- 복잡한 멀티태스킹이나 정밀한 하드웨어 제어가 필요하지 않습니다.

실용적인 조언

- ESP32에 익숙해지려면 Arduino IDE부터 시작하세요 .
- 자신감이 생기면 ESP-IDF 로 전환하여 더욱 심도 있는 제어와 전문가급 프로젝트를 수행하세요.
- 많은 개념(GPIO, I2C, SPI, Wi-Fi)이 Arduino에서 ESP-IDF로 이전되었습니다.
- 전문적인 임베디드 개발에 가까운 IDE 경험을 위해 ESP-IDF + VS Code를 결합하세요

ESP-IDF(Espressif IoT 개발 프레임워크)



- ESP32
- ESP32-S2, ESP32-S3
- ESP32-C2, ESP32-C3, ESP32-C5, ESP32-C6, ESP32-C61
- ESP32-H2
- ESP32-P4

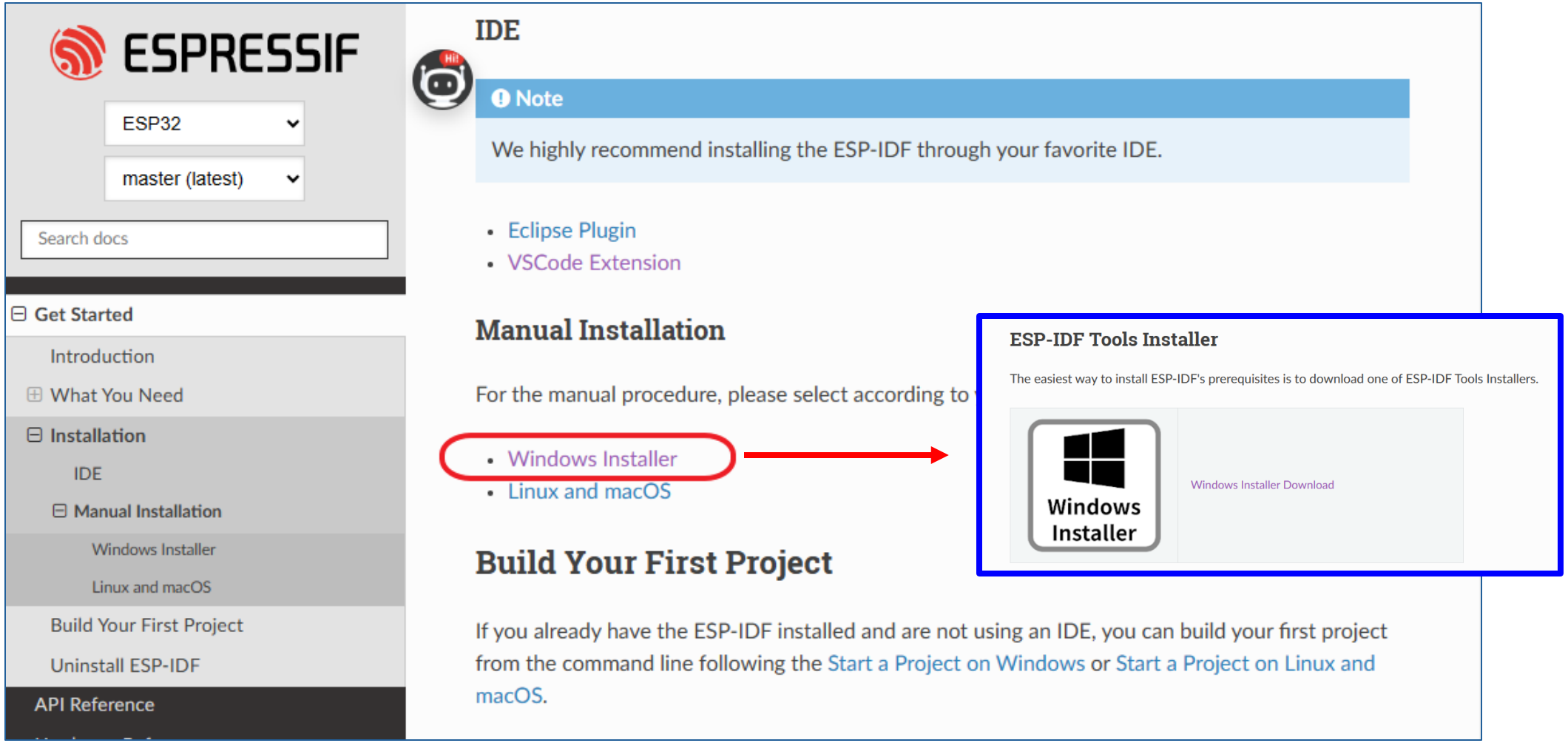
1. RTOS 🙂
2. 코어 수 🤔
3. 업데이트 🙂
4. 개발 난이도 🤔
5. 커뮤니티 🤔
6. 연산 벤치마킹 🤔
7. 이미지 처리 벤치마킹 🤔
8. 스케줄링 🙂

https://blog.naver.com/nanotoly/223322065269

ESP-IDF VS Code 한 방에 설치 1

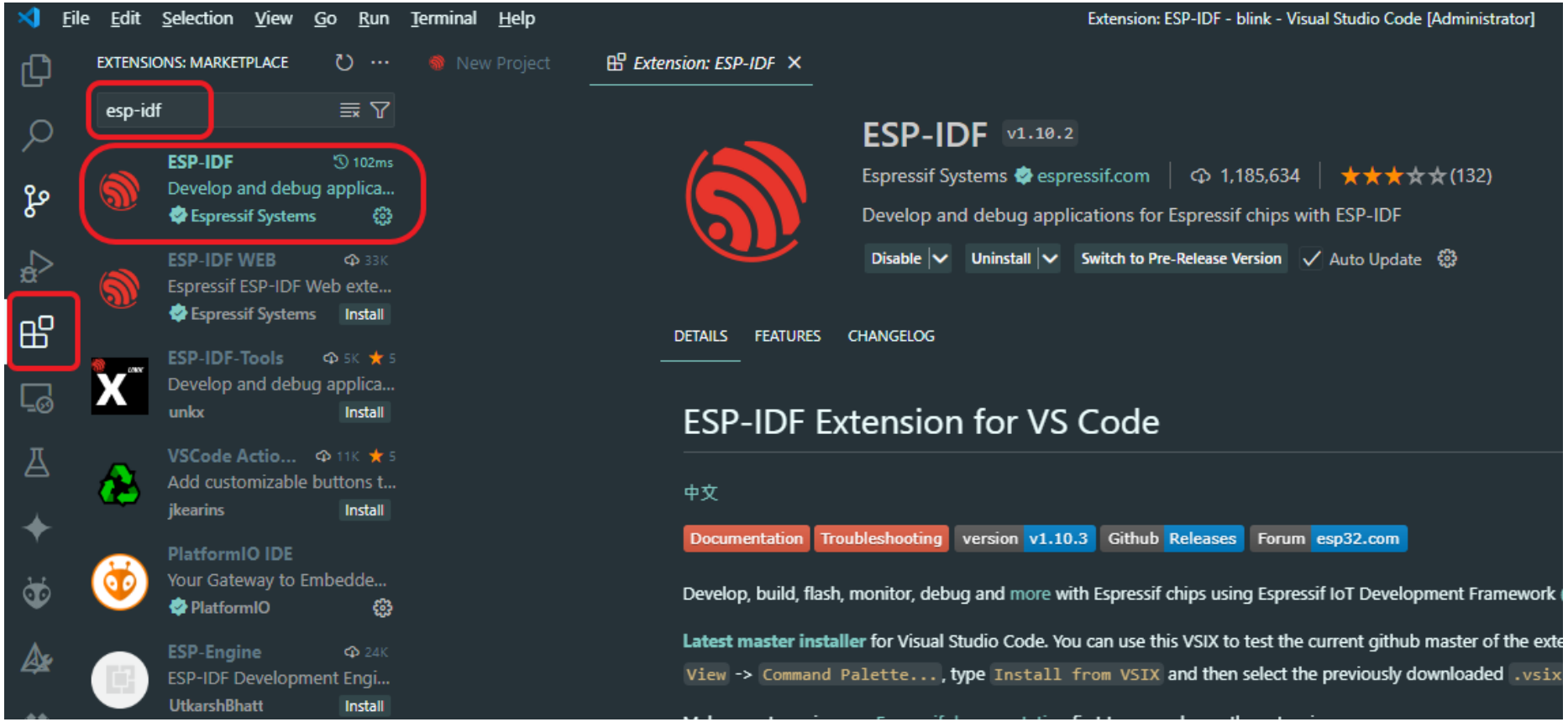
개발 블로그의 공식적인 설치방법으로 해결이 안되는 경우, 열 받으면 아래 방법을 사용한다.

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html#manual-installation



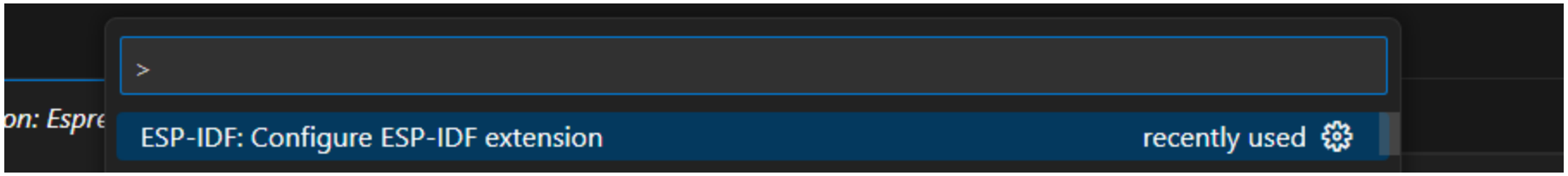
ESP-IDF VS Code 한 방에 설치 1

- 체크하는 부분에서 본인이 사용하고자 하는 ESP32 보드가 모두 체크되어있는지 정도는 확인
- 참고로 ESP-IDF 는 CMD 혹은 Powershell 창에서 개발환경을 제공한다. 이는 마치 파이썬의 IDLE 수준의 개발환경이므로 사실상 개발이 불가능하다 보면 됩니다.
- 용량이 1.2GB 정도 되서 10분 정도 기다리면 설치가 완료된다. 설치가 완료되면 VS 코드를 실행합니다.

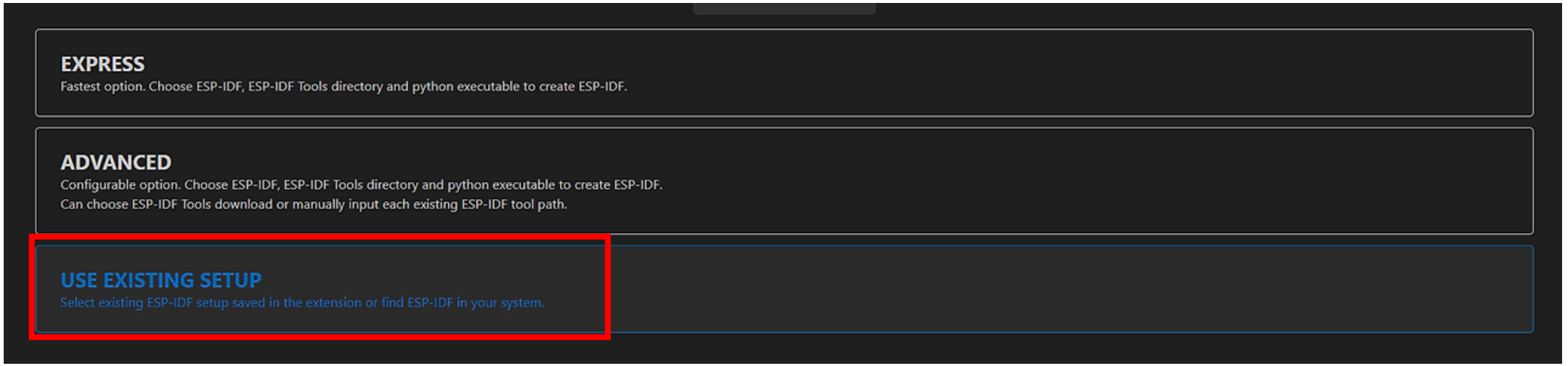


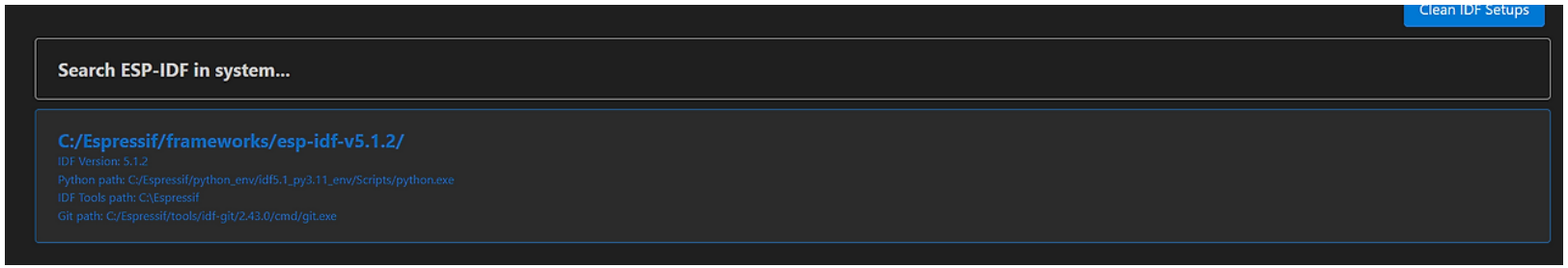
ESP-IDF VS Code 한 방에 설치 1

- View -> Command Pallete... 에서 Configure ESP-IDF extension을 클릭해주면,

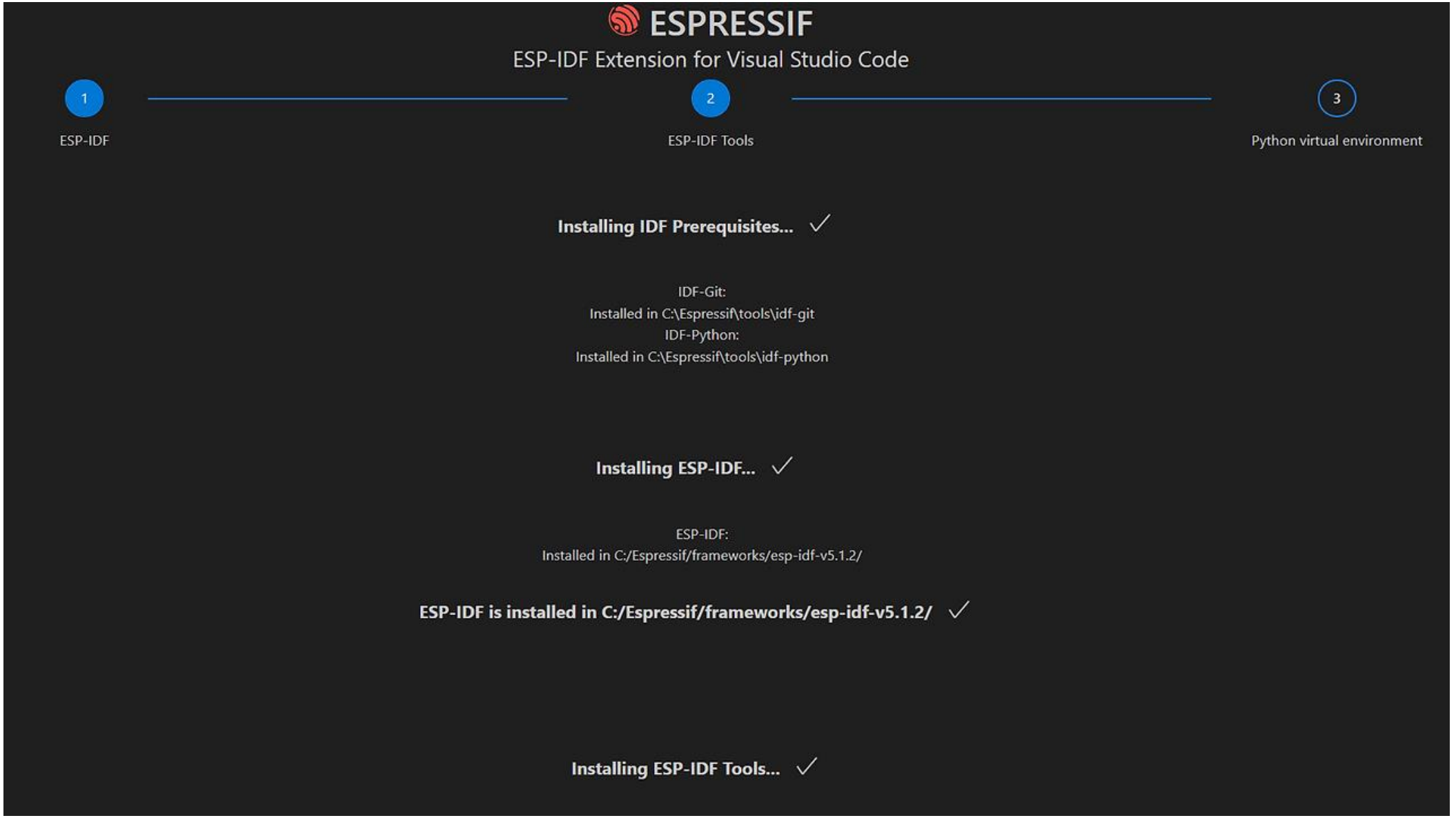


- 메세지가 뜨면서 ESP-IDF 가 설치되어있다고 뭐라뭐라 뜬다. 세번째 거를 클릭해준다.





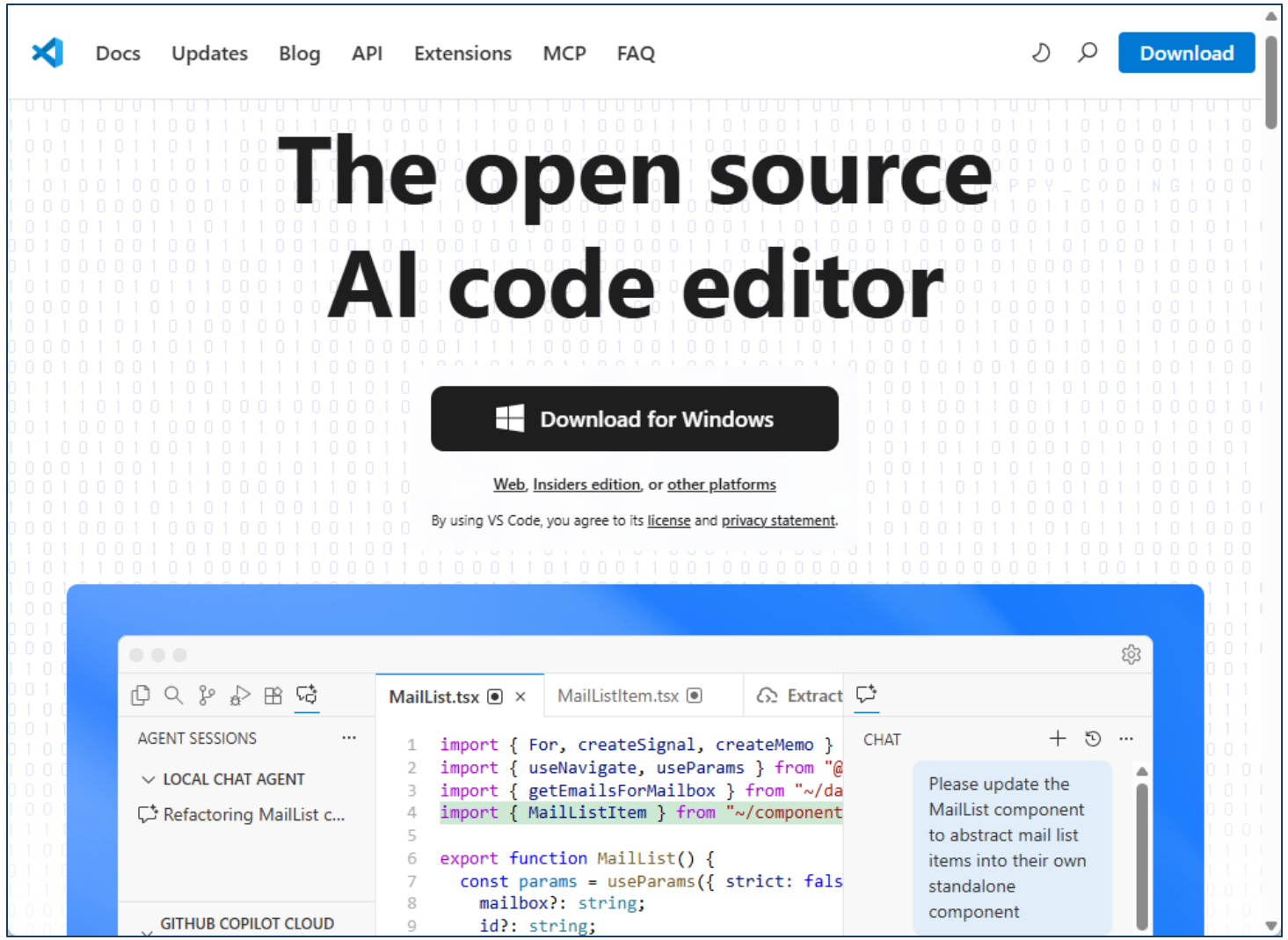
ESP-IDF VS Code 한 방에 설치 1



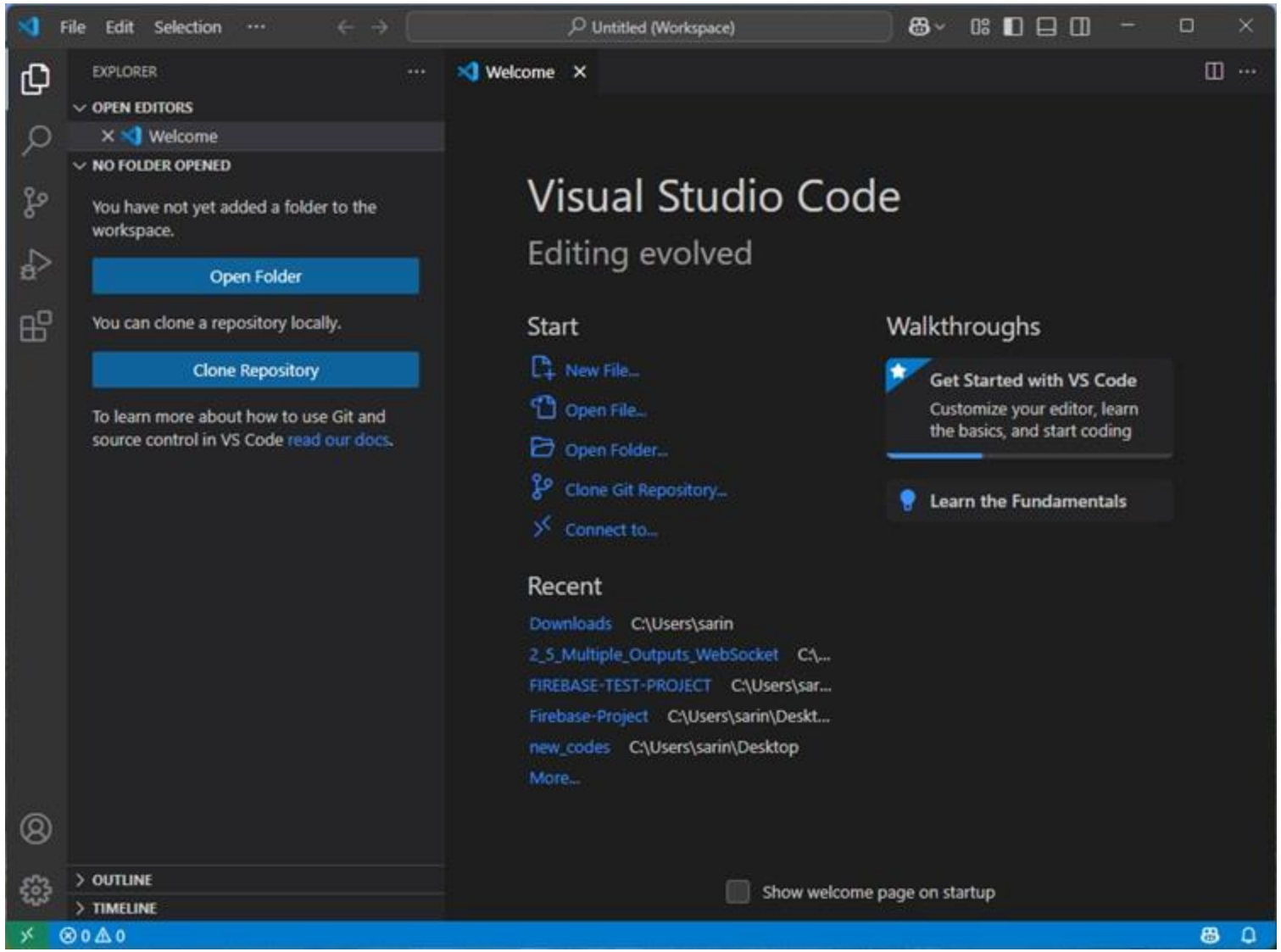
ESP-IDF Visual Studio Code 설치

Mac OS x, Linux Ubuntu 설치 참고
https://fishpoint.tistory.com/12218

https://code.visualstudio.com/



ESP-IDF Visual Studio Code 설치 완료



ESP-IDF 파이선 설치

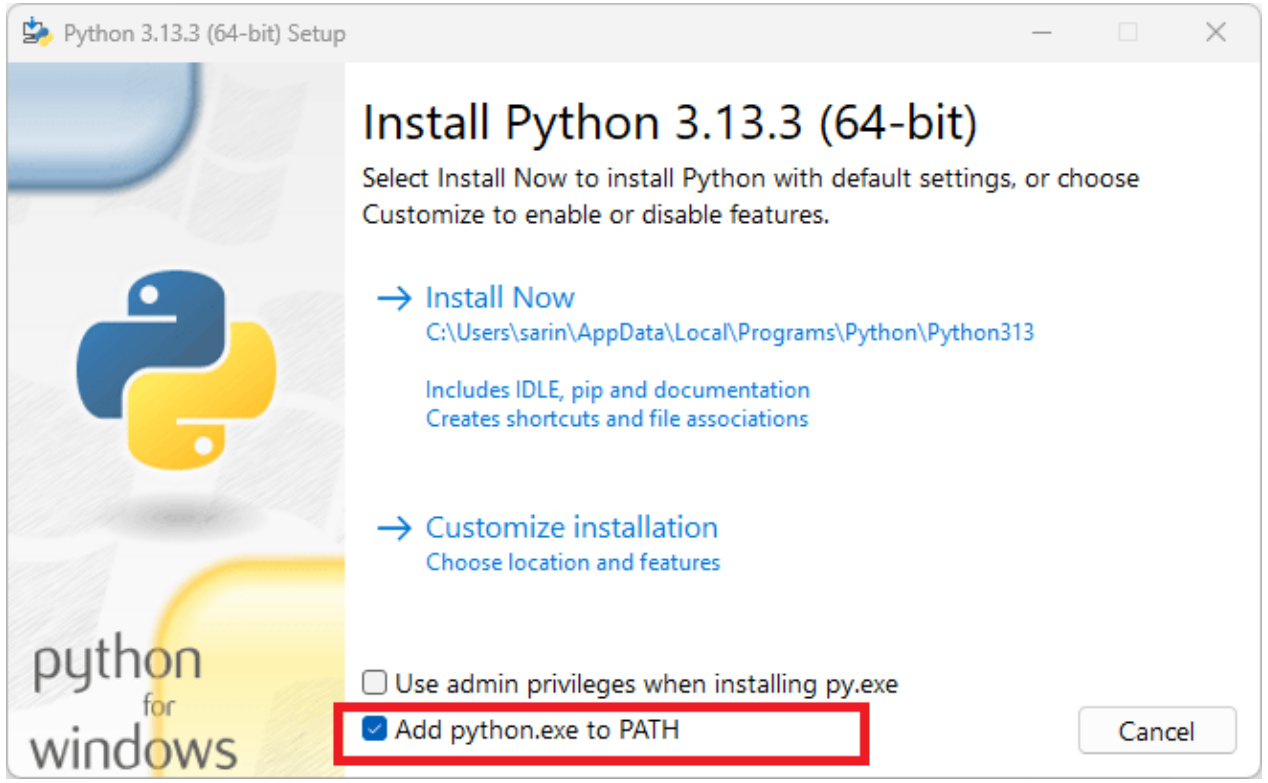
- ESP-IDF로 ESP32를 프로그래밍하려면 컴퓨터에 Python 3.5 이상 버전이 설치되어 있어야 합니다.

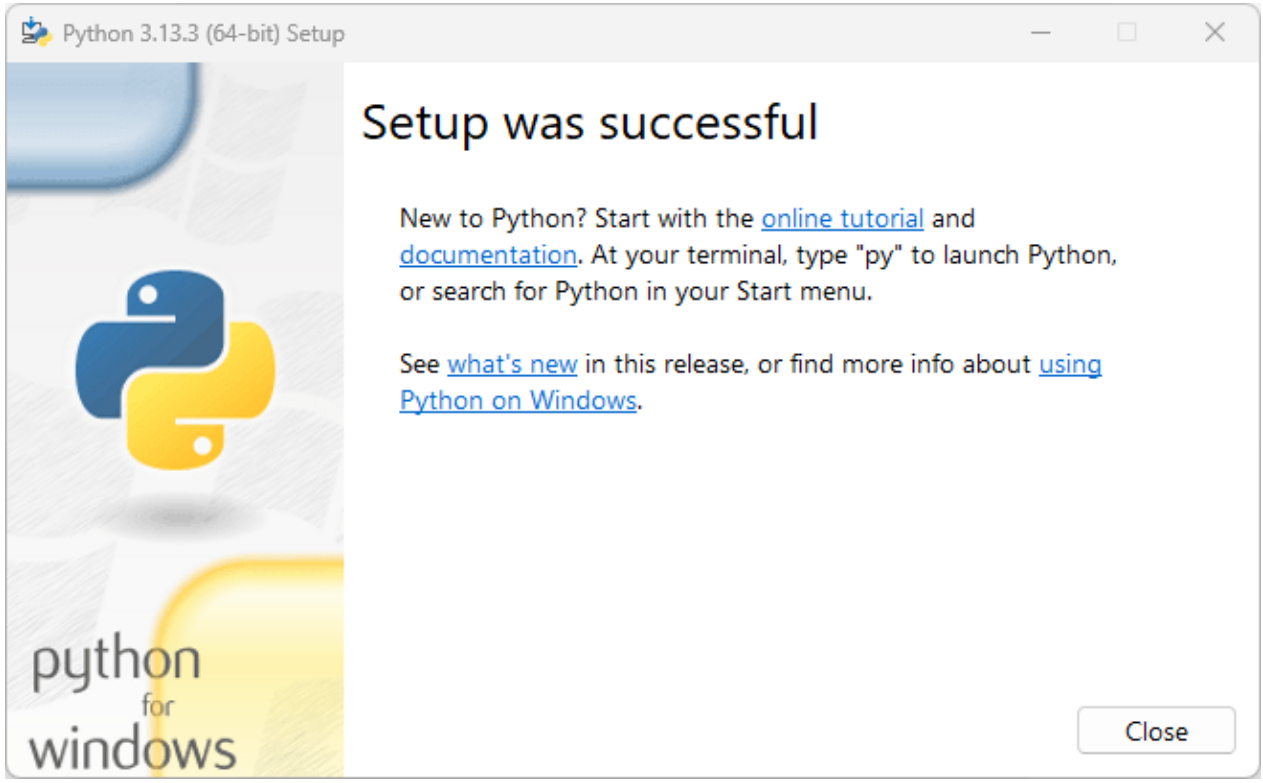
  https://www.python.org/downloads/

ESP-IDF 파이선 설치

- ESP-IDF로 ESP32를 프로그래밍하려면 컴퓨터에 Python 3.5 이상 버전이 설치되어 있어야 합니다.

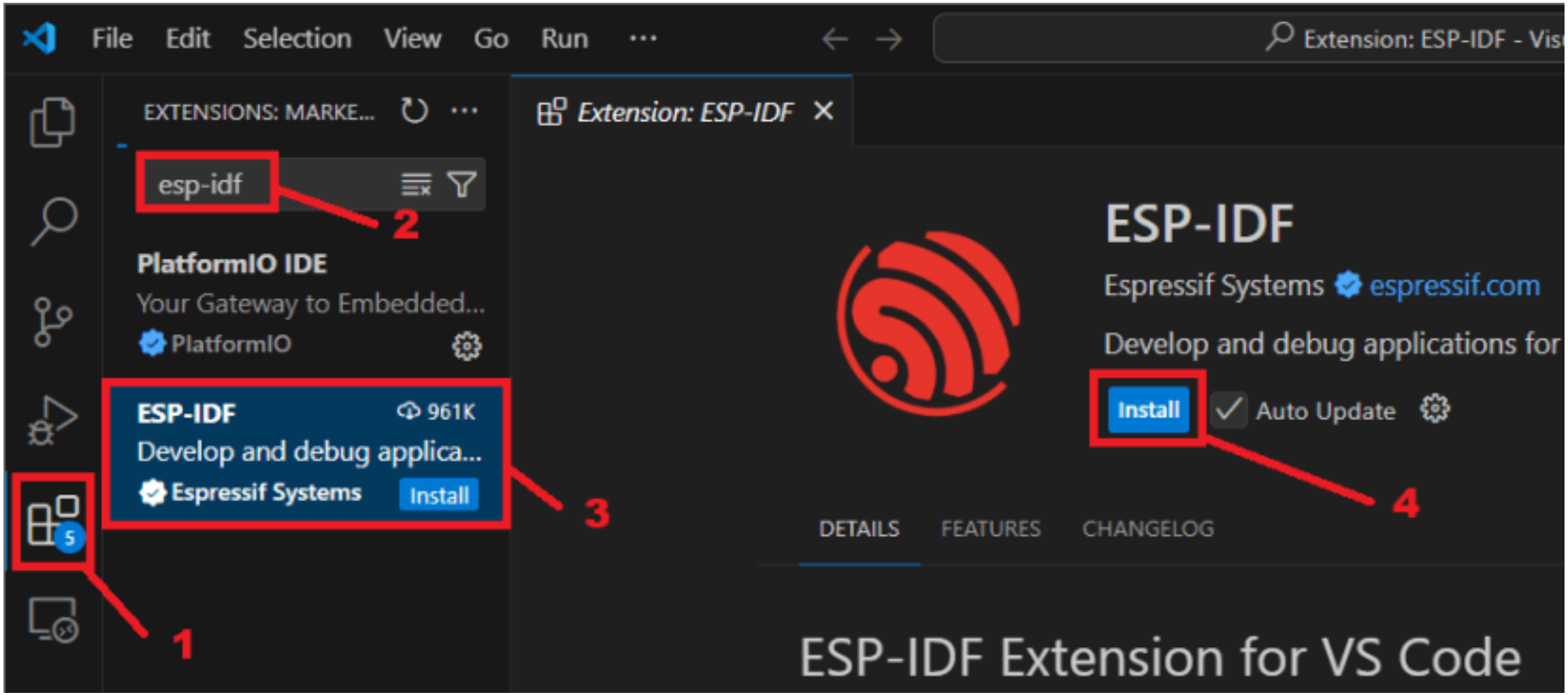
  https://www.python.org/downloads/





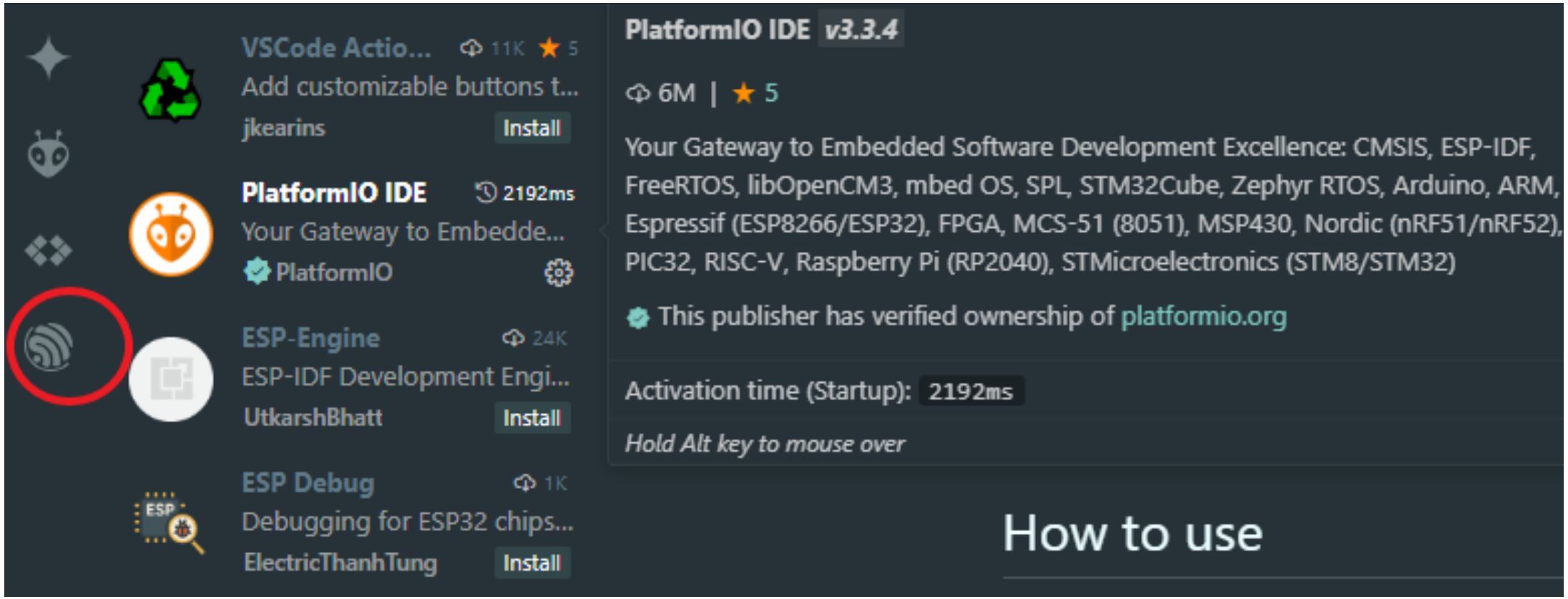
VS Code용 ESP-IDF 확장 프로그램 설치

- "ESP-IDF"를 검색하고 Espressif Systems의 ESP-IDF를 선택한 후 설치 버튼을 클릭합니다.



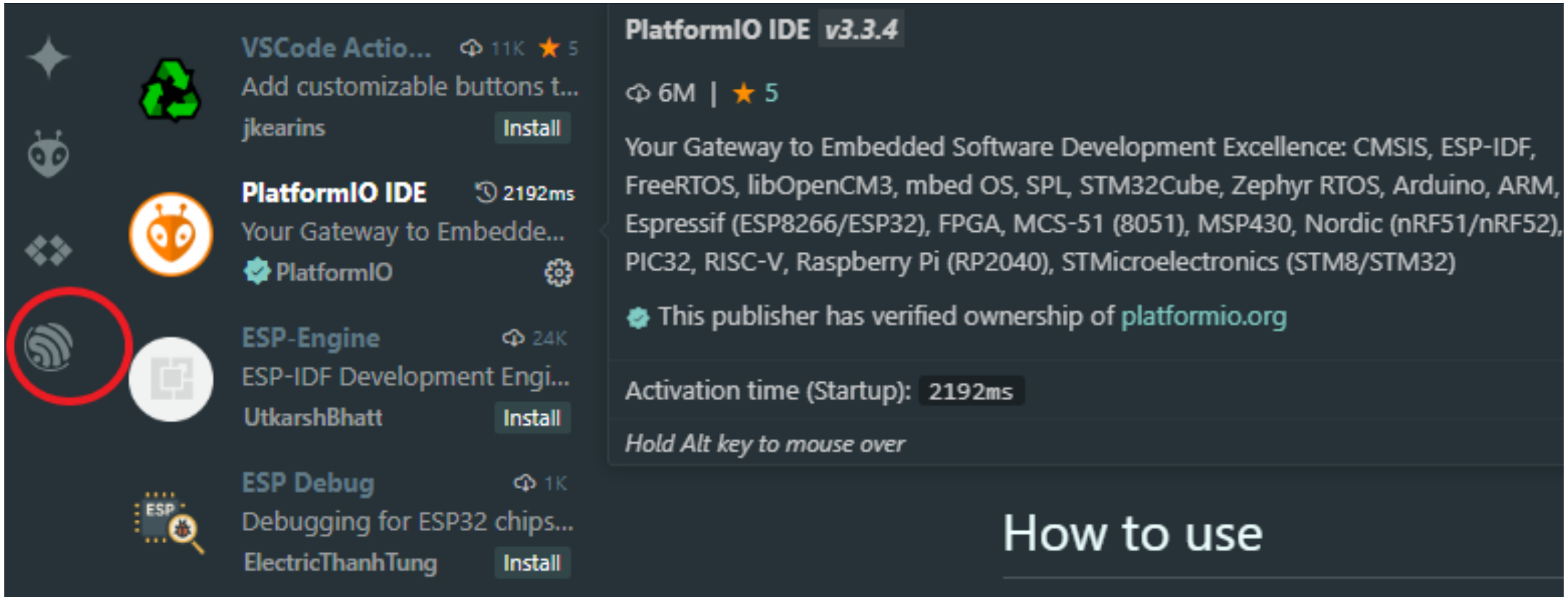
VS Code용 ESP-IDF 확장 프로그램 설치

- 이제 Espressif Systems의 ESP-IDF 확장 프로그램이 IDE에 설치되었습니다. 설치 완료 후 왼쪽 아래에 espressif 아이콘이 생기고 이것을 클릭하면 아래와 같은 화면이 나온다.

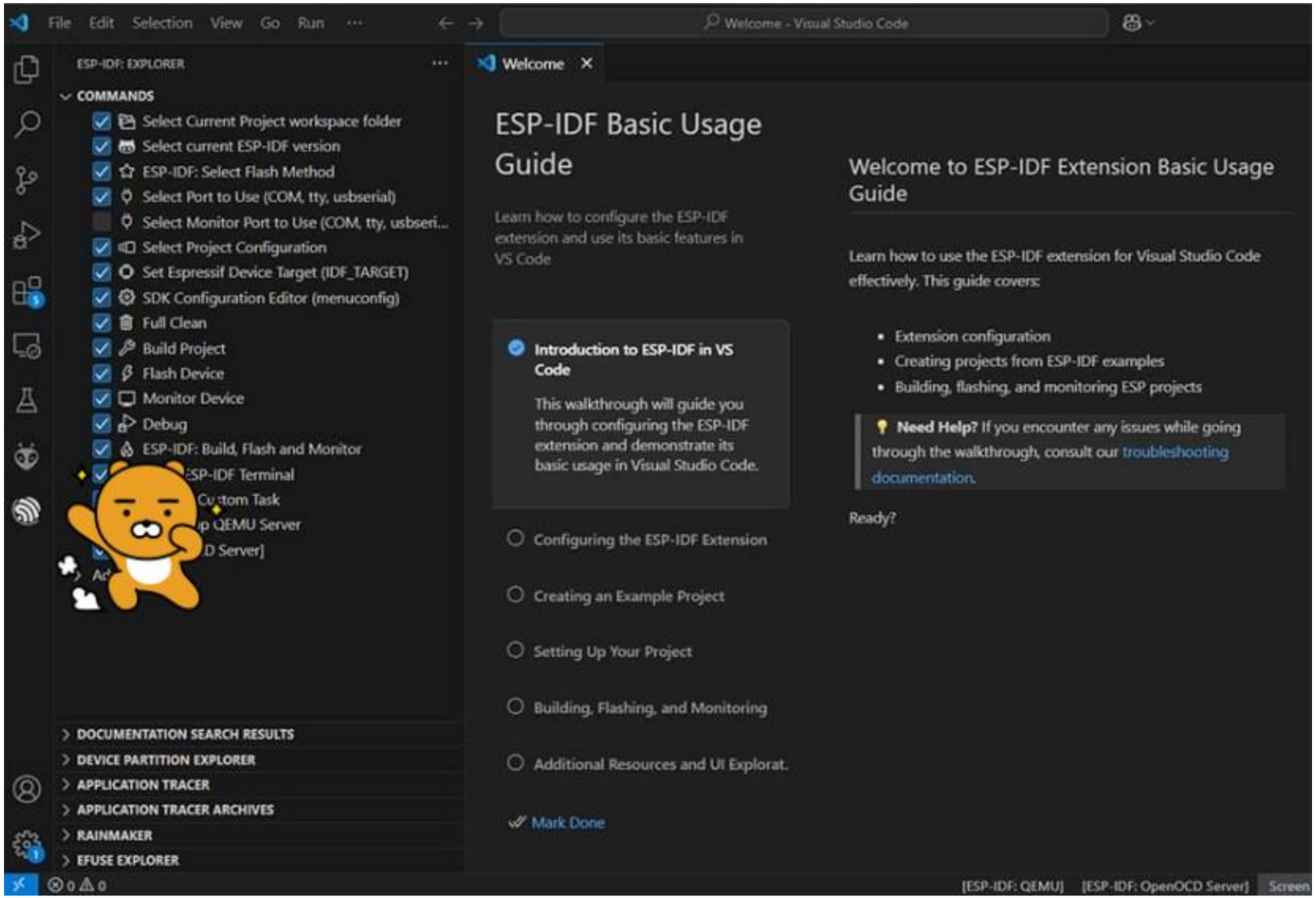


VS Code용 ESP-IDF 확장 프로그램 설치

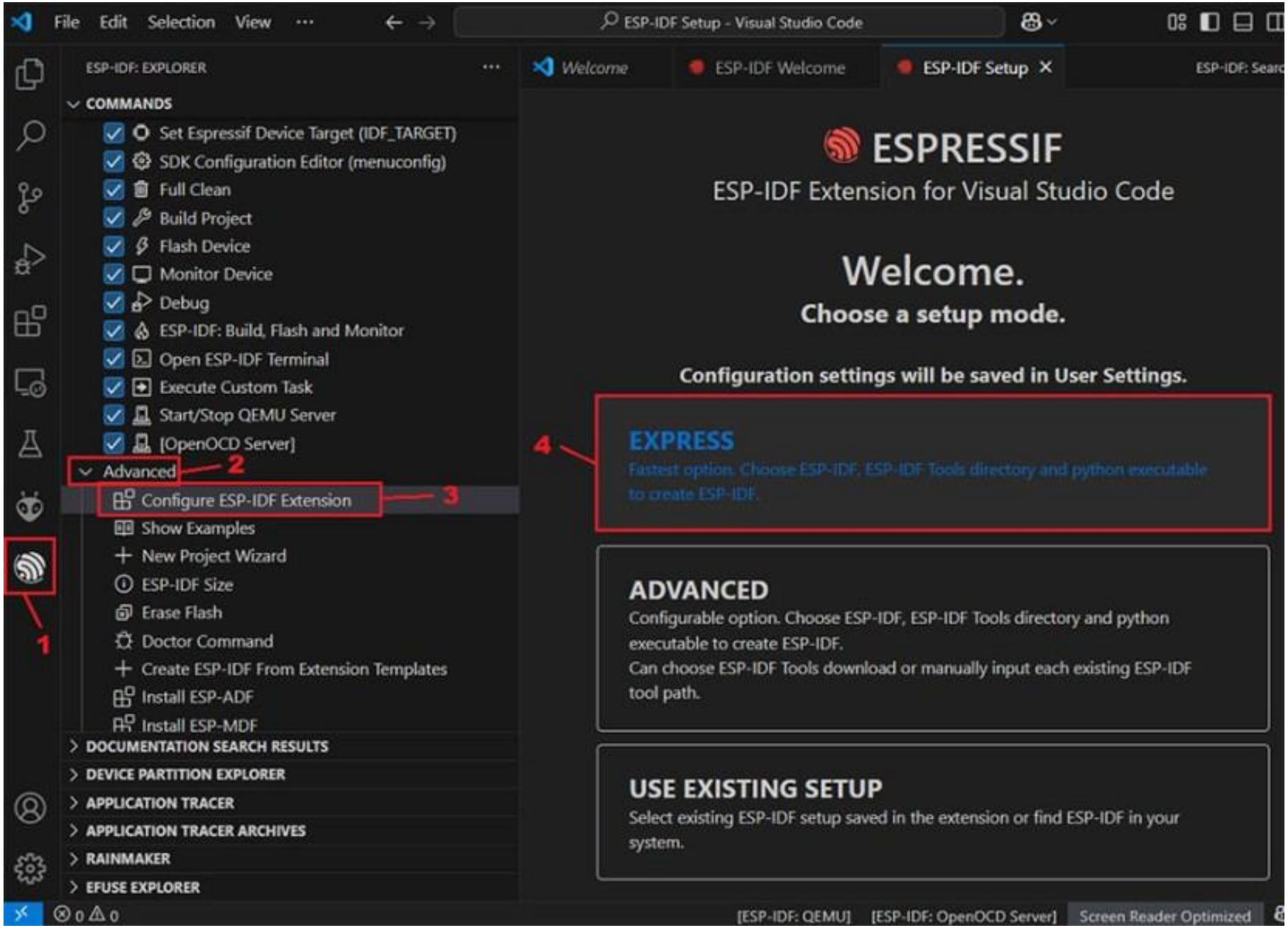
- 이제 Espressif Systems의 ESP-IDF 확장 프로그램이 IDE에 설치되었습니다. 설치 완료 후 왼쪽 아래에 espressif 아이콘이 생기고 이것을 클릭하면 아래와 같은 화면이 나온다.



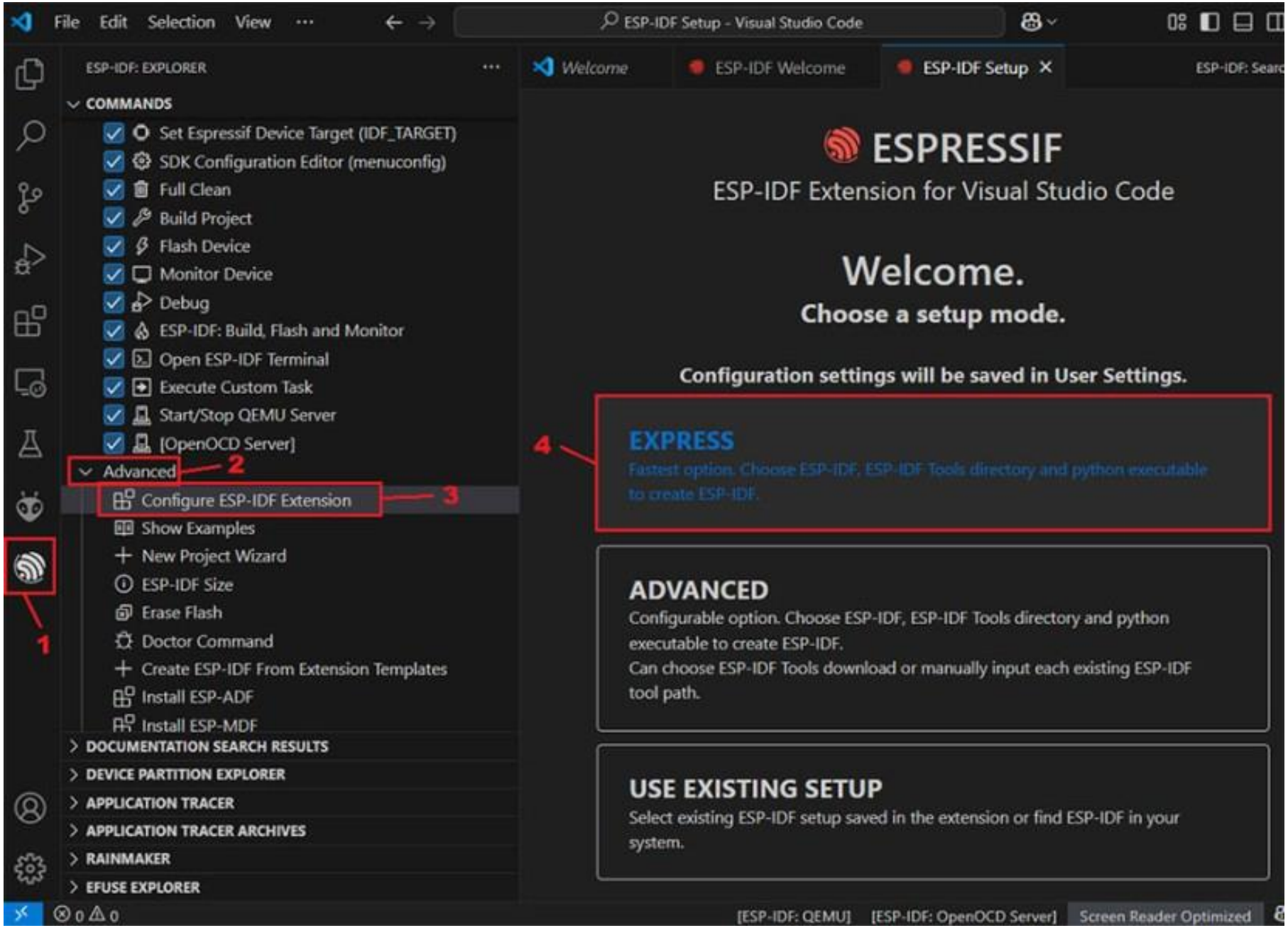
Espressif 사용 가이드



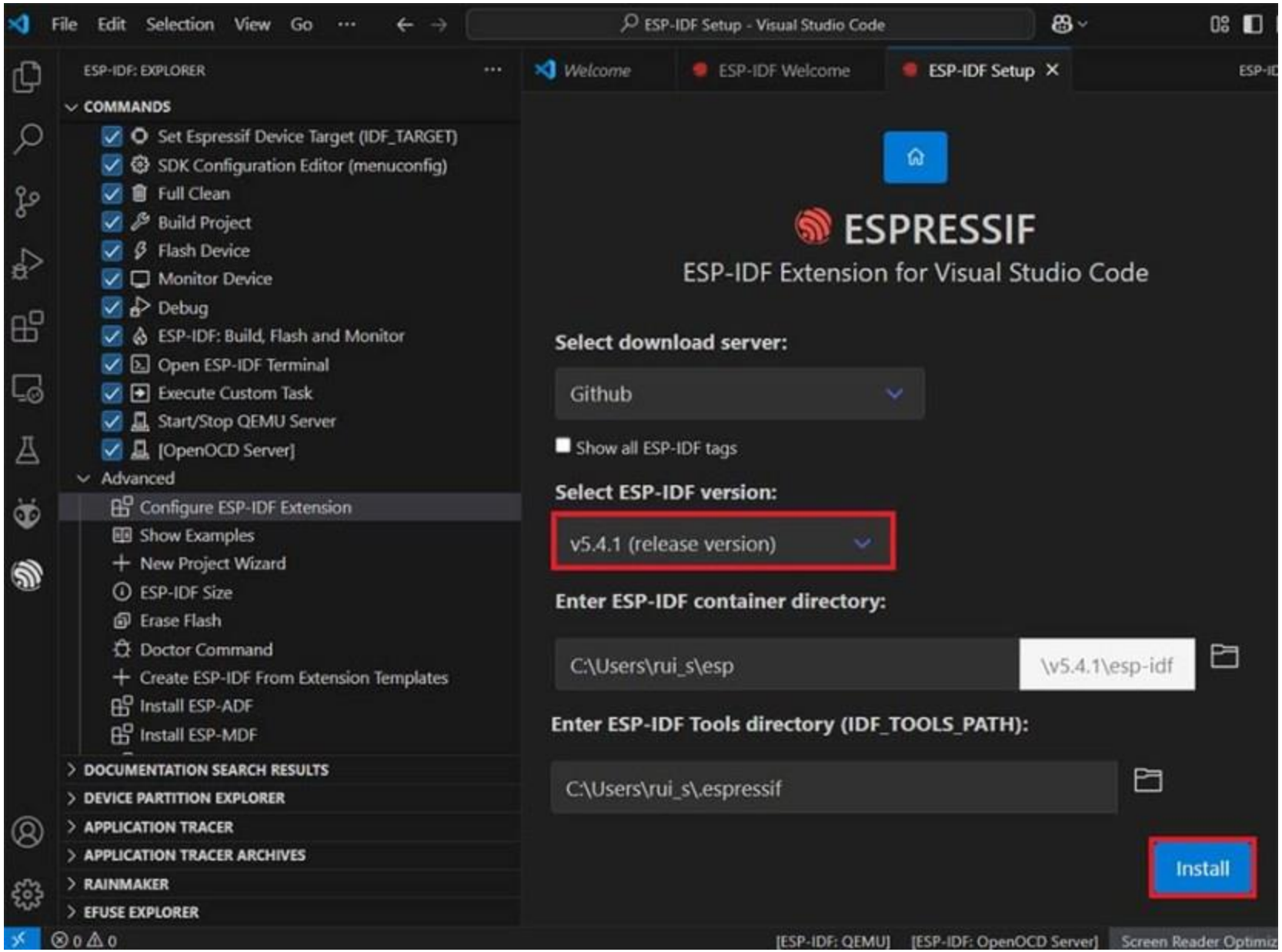
ESP-IDF 확장 구성


File Edit Selection View ···
ESP-IDF Setup - Visual Studio Code
ESP-IDF: EXPLORER
COMMANDS
Set Espressif Device Target (IDF_TARGET)
SDK Configuration Editor (menuconfig)
Full Clean
Build Project
Flash Device
Monitor Device
Debug
ESP-IDF: Build, Flash and Monitor
Open ESP-IDF Terminal
Execute Custom Task
Start/Stop QEMU Server
[OpenOCD Server]
Advanced
Configure ESP-IDF Extension
Show Examples
New Project Wizard
ESP-IDF Size
Erase Flash
Doctor Command
Create ESP-IDF From Extension Templates
Install ESP-ADF
Install ESP-MDF
DOCUMENTATION SEARCH RESULTS
DEVICE PARTITION EXPLORER
APPLICATION TRACER
APPLICATION TRACER ARCHIVES
RAINMAKER
EFUSE EXPLORER
1
2
3
4
Welcome
ESP-IDF Welcome
ESP-IDF Setup
ESPRESSIF
ESP-IDF Extension for Visual Studio Code
Welcome.
Choose a setup mode.
Configuration settings will be saved in User Settings.
EXPRESS
Fastest option. Choose ESP-IDF, ESP-IDF Tools directory and python executable to create ESP-IDF.
ADVANCED
Configurable option. Choose ESP-IDF, ESP-IDF Tools directory and python executable to create ESP-IDF.
Can choose ESP-IDF Tools download or manually input each existing ESP-IDF tool path.
USE EXISTING SETUP
Select existing ESP-IDF setup saved in the extension or find ESP-IDF in your system.
[ESP-IDF: QEMU]
[ESP-IDF: OpenOCD Server]
Screen Reader Optimized

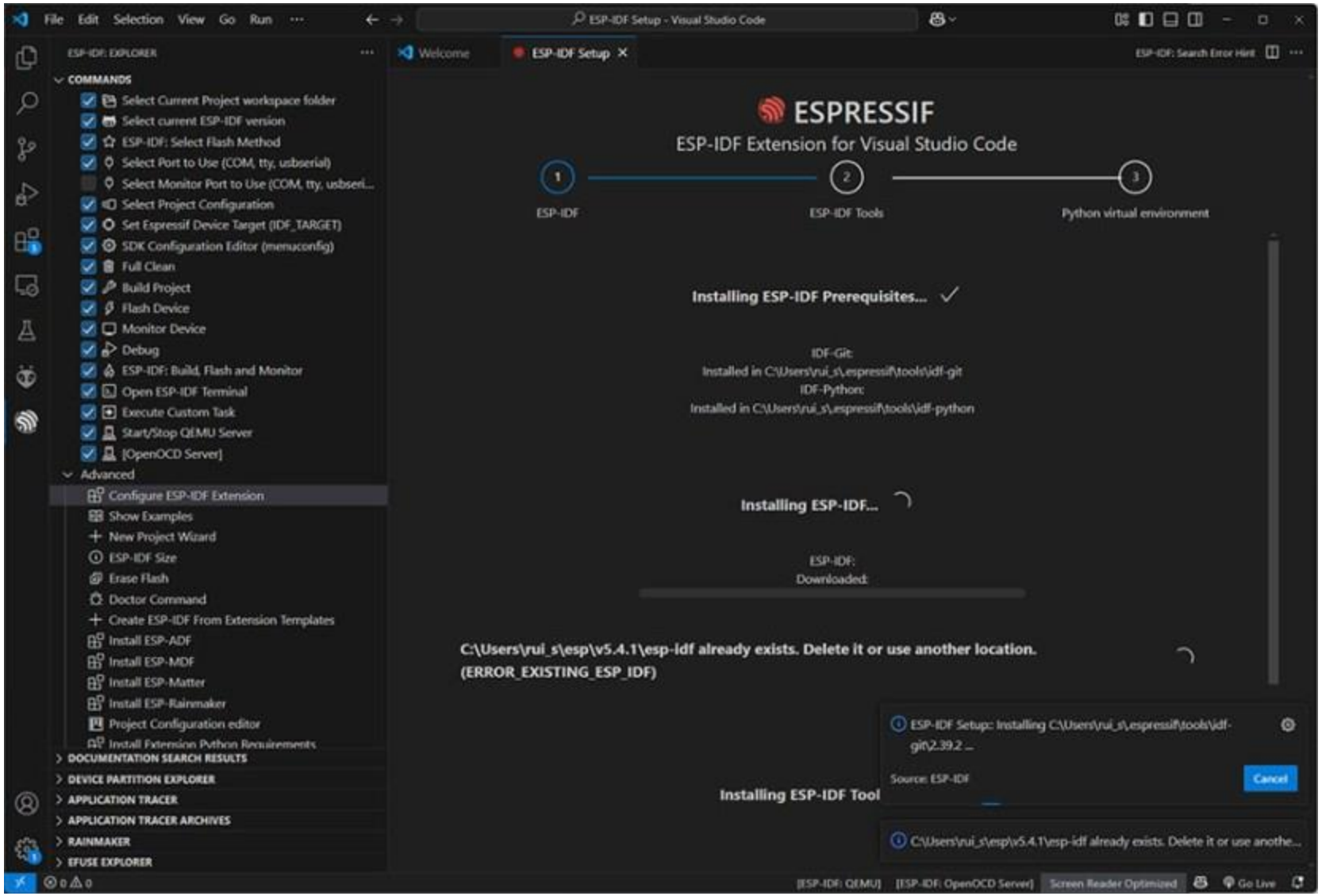
ESP-IDF 확장 구성


File Edit Selection View ···
ESP-IDF Setup - Visual Studio Code
ESP-IDF: EXPLORER
COMMANDS
Set Espressif Device Target (IDF_TARGET)
SDK Configuration Editor (menuconfig)
Full Clean
Build Project
Flash Device
Monitor Device
Debug
ESP-IDF: Build, Flash and Monitor
Open ESP-IDF Terminal
Execute Custom Task
Start/Stop QEMU Server
[OpenOCD Server]
Advanced
Configure ESP-IDF Extension
Show Examples
New Project Wizard
ESP-IDF Size
Erase Flash
Doctor Command
Create ESP-IDF From Extension Templates
Install ESP-ADF
Install ESP-MDF
DOCUMENTATION SEARCH RESULTS
DEVICE PARTITION EXPLORER
APPLICATION TRACER
APPLICATION TRACER ARCHIVES
RAINMAKER
EFUSE EXPLORER
1
2
3
4
Welcome
ESP-IDF Welcome
ESP-IDF Setup
ESPRESSIF
ESP-IDF Extension for Visual Studio Code
Welcome.
Choose a setup mode.
Configuration settings will be saved in User Settings.
EXPRESS
Fastest option. Choose ESP-IDF, ESP-IDF Tools directory and python executable to create ESP-IDF.
ADVANCED
Configurable option. Choose ESP-IDF, ESP-IDF Tools directory and python executable to create ESP-IDF.
Can choose ESP-IDF Tools download or manually input each existing ESP-IDF tool path.
USE EXISTING SETUP
Select existing ESP-IDF setup saved in the extension or find ESP-IDF in your system.
[ESP-IDF: QEMU]
[ESP-IDF: OpenOCD Server]
Screen Reader Optimized

ESP-IDF 버전을 선택 후 설치

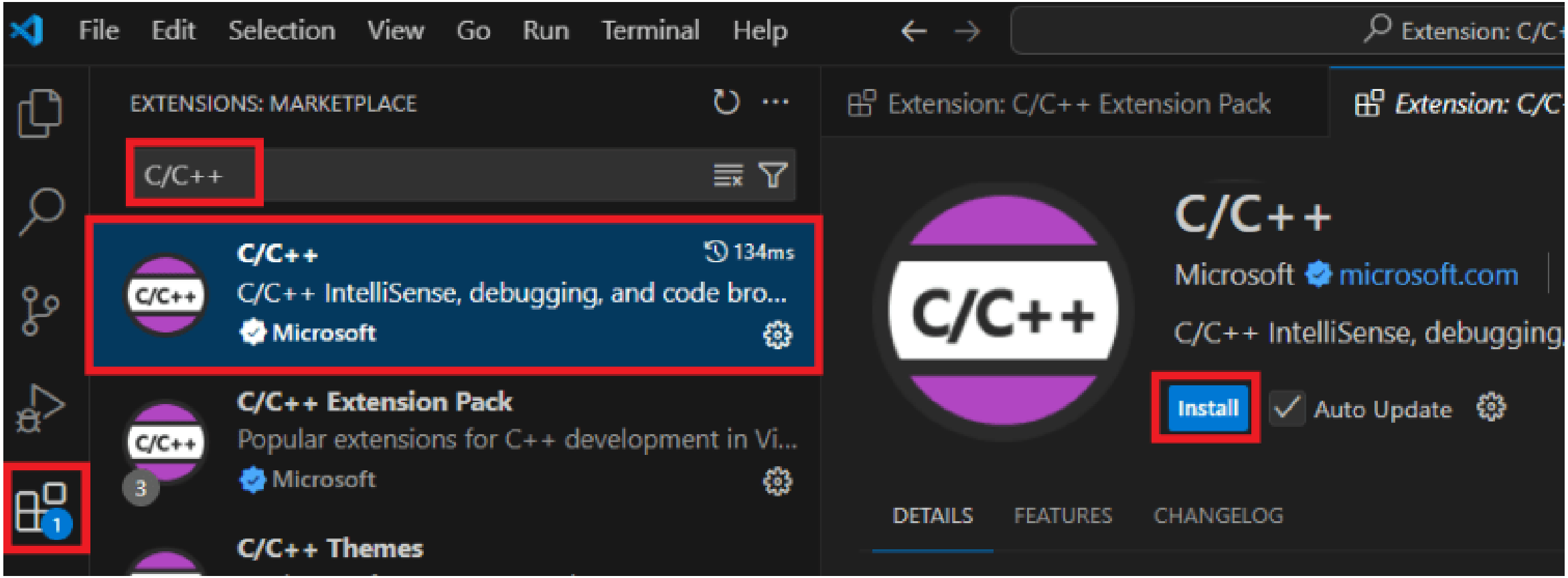


설치 완료


File Edit Selection View Go Run
ESP-IDF Setup - Visual Studio Code
ESP-IDF: EXPLORER
COMMANDS
Select Current Project workspace folder
Select current ESP-IDF version
ESP-IDF: Select Flash Method
Select Port to Use (COM, tty, usbserial)
Select Monitor Port to Use (COM, tty, usbseri...
Select Project Configuration
Set Espressif Device Target (IDF_TARGET)
SDK Configuration Editor (menuconfig)
Full Clean
Build Project
Flash Device
Monitor Device
Debug
ESP-IDF: Build, Flash and Monitor
Open ESP-IDF Terminal
Execute Custom Task
Start/Stop QEMU Server
[OpenOCD Server]
Advanced
Configure ESP-IDF Extension
Show Examples
New Project Wizard
ESP-IDF Size
Erase Flash
Doctor Command
Create ESP-IDF From Extension Templates
Install ESP-ADF
Install ESP-MDF
Install ESP-Matter
Install ESP-Rainmaker
Project Configuration editor
DOCUMENTATION SEARCH RESULTS
DEVICE PARTITION EXPLORER
APPLICATION TRACER
APPLICATION TRACER ARCHIVES
RAINMAKER
EFUSE EXPLORER
Welcome
ESP-IDF Setup
ESP-IDF: Search Error Hint
ESPRESSIF
ESP-IDF Extension for Visual Studio Code
1
ESP-IDF
2
ESP-IDF Tools
3
Python virtual environment
Installing ESP-IDF Prerequisites...
IDF-Git:
Installed in C:\Users\rui_s\.espressif\tools\idf-git
IDF-Python:
Installed in C:\Users\rui_s\.espressif\tools\idf-python
Installing ESP-IDF...
ESP-IDF:
Downloaded:
C:\Users\rui_s\esp\v5.4.1\esp-idf already exists. Delete it or use another location.
(ERROR_EXISTING_ESP_IDF)
Installing ESP-IDF Tool
ESP-IDF Setup: Installing C:\Users\rui_s\.espressif\tools\idf-git\2.39.2 ...
Source: ESP-IDF
Cancel
C:\Users\rui_s\esp\v5.4.1\esp-idf already exists. Delete it or use anothe...
[ESP-IDF: QEMU]
[ESP-IDF: OpenOCD Server]
Screen Reader Optimized
Go Live

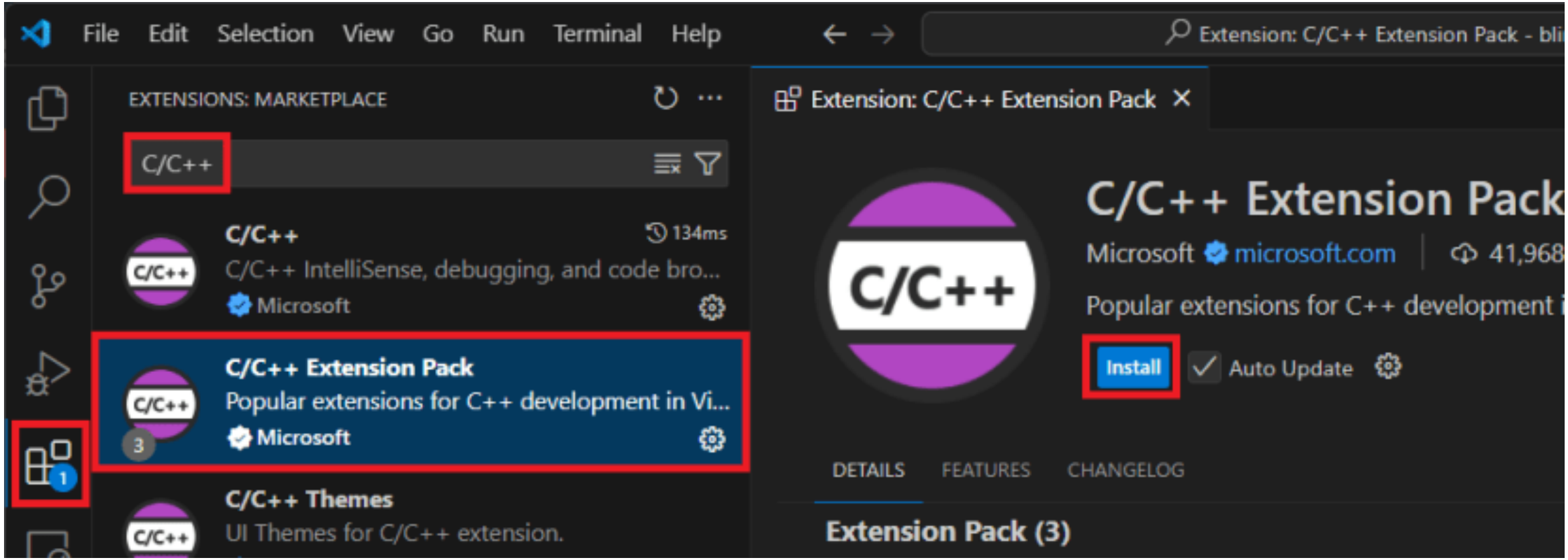
VS Code용 C/C++ 확장 프로그램 설치(선택 사항)

- " C/C++ "를 검색하고 첫 번째 옵션을 선택한 후 설치 버튼을 클릭합니다.

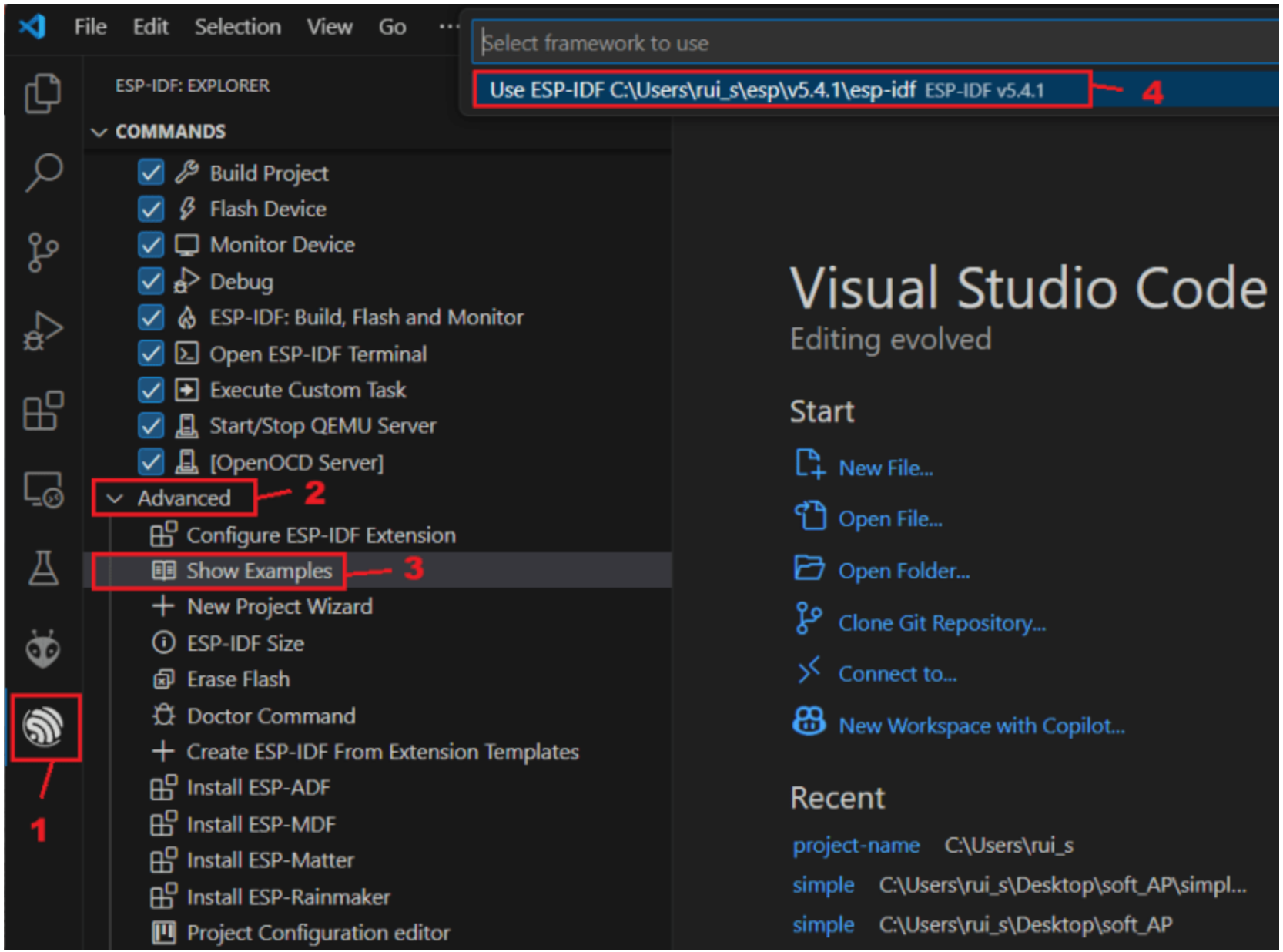


VS Code용 C/C++ 확장 프로그램 설치(선택 사항)

- C/C++ Extension Pack " C/C++ 확장 팩 "을 설치할 수도 있습니다 .

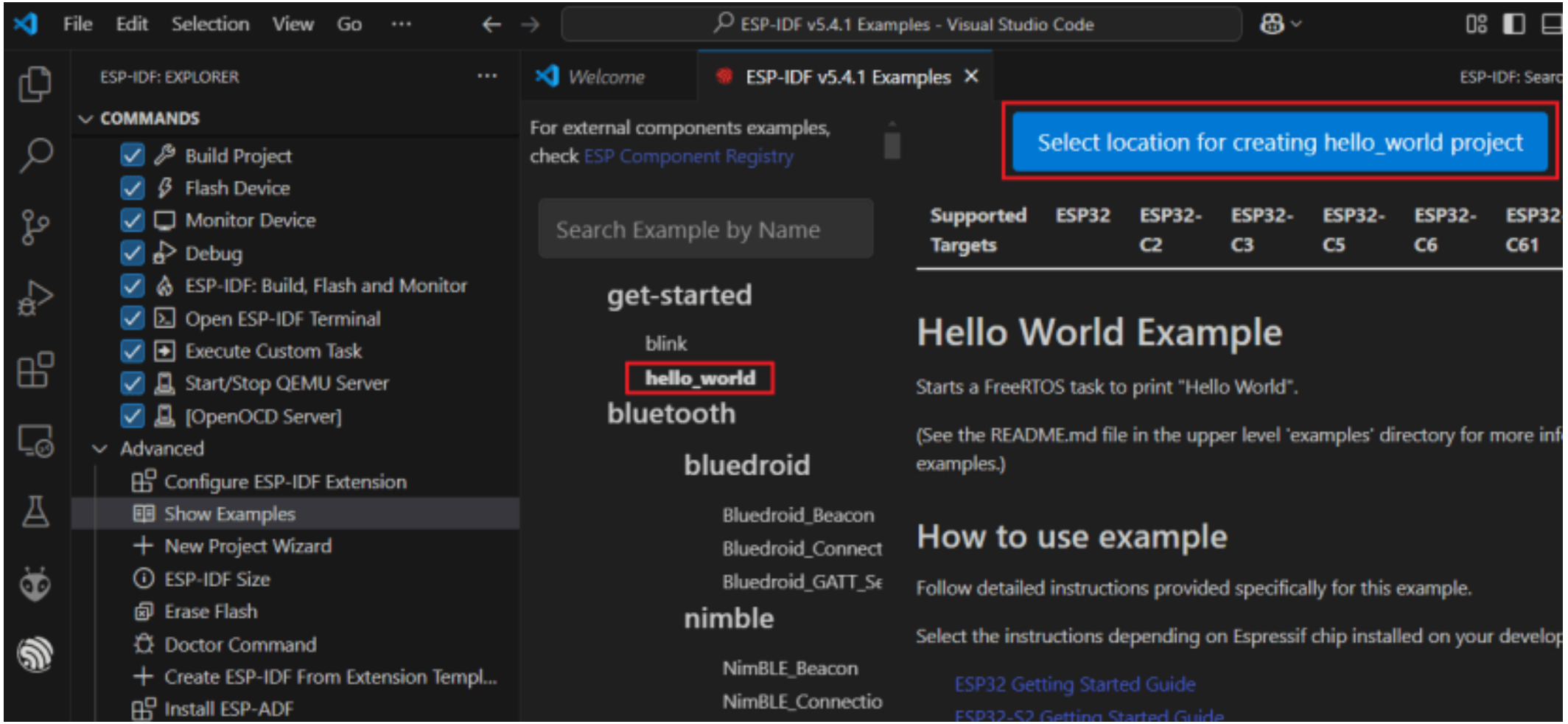


ESP-IDF ESP32 프로그래밍하기 – Hello World Code



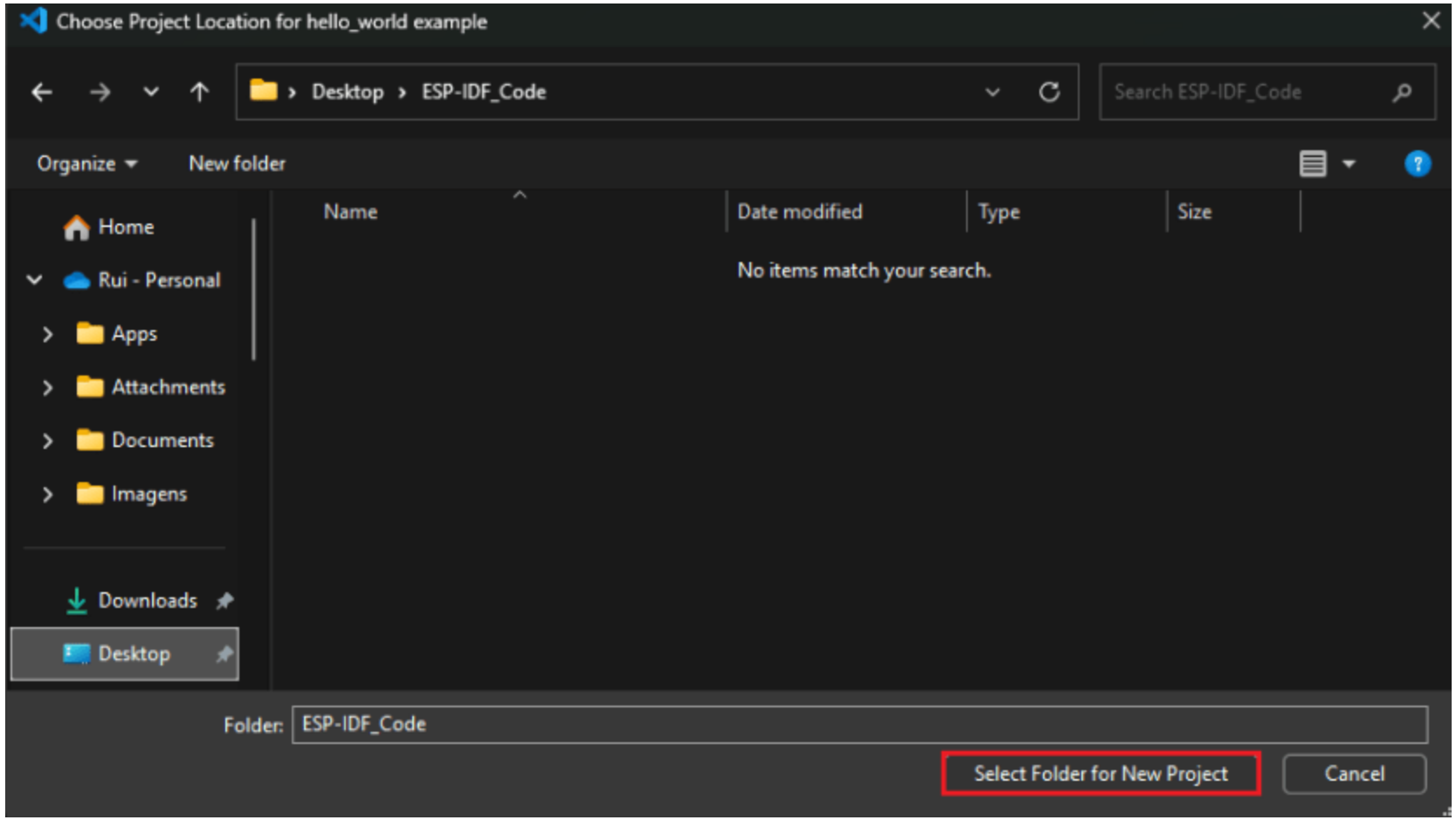
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

페이지 상단에서 "hello_world 프로젝트를 생성할 위치 선택" 버튼을 클릭합니다.



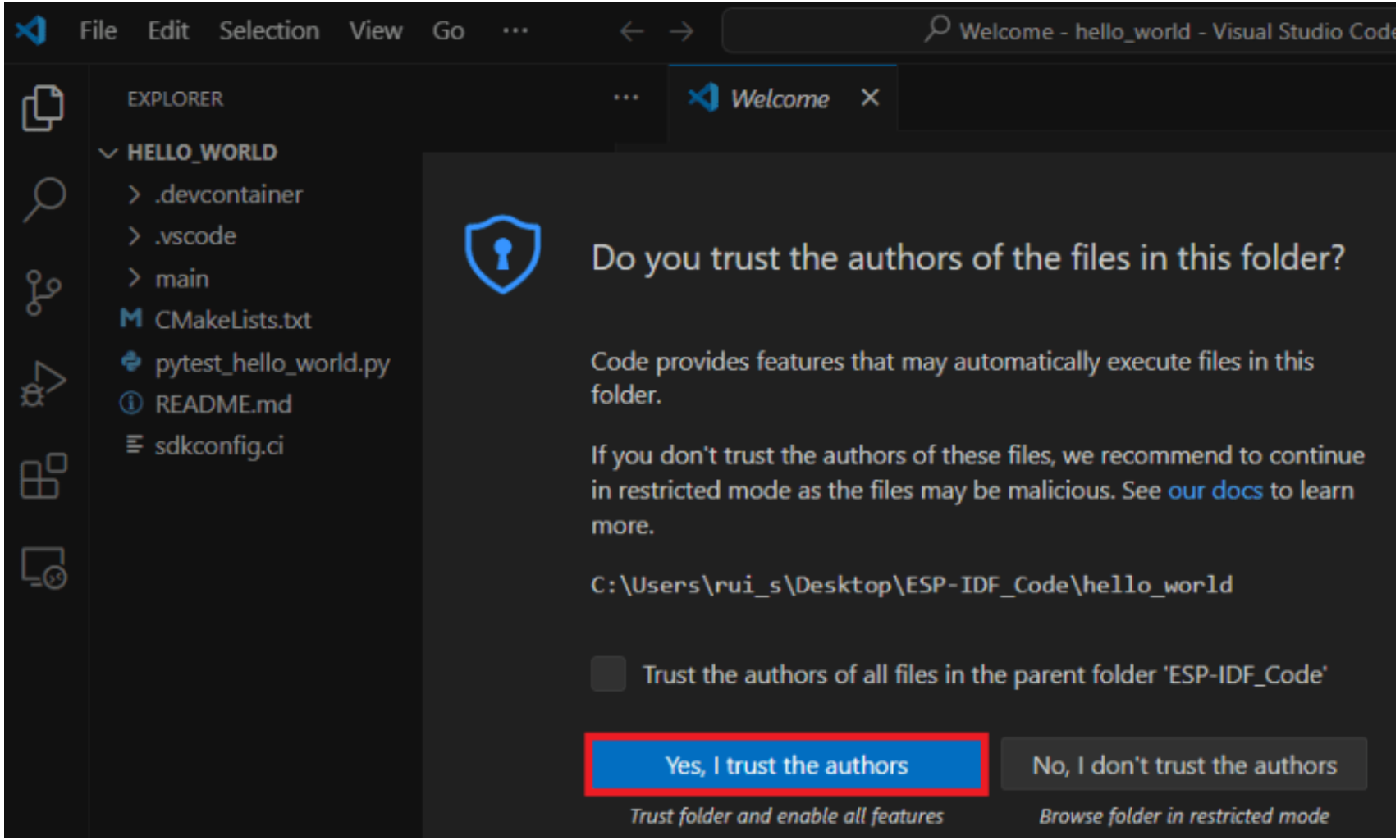
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

참고: Google Drive/One Drive/Dropbox 폴더는 사용하지 마세요 . 코드가 빌드되는 동안 많은 파일을 작성/생성하게 되며, 클라우드 폴더에 있는 경우 이 프로세스가 매우 느릴 수 있습니다.



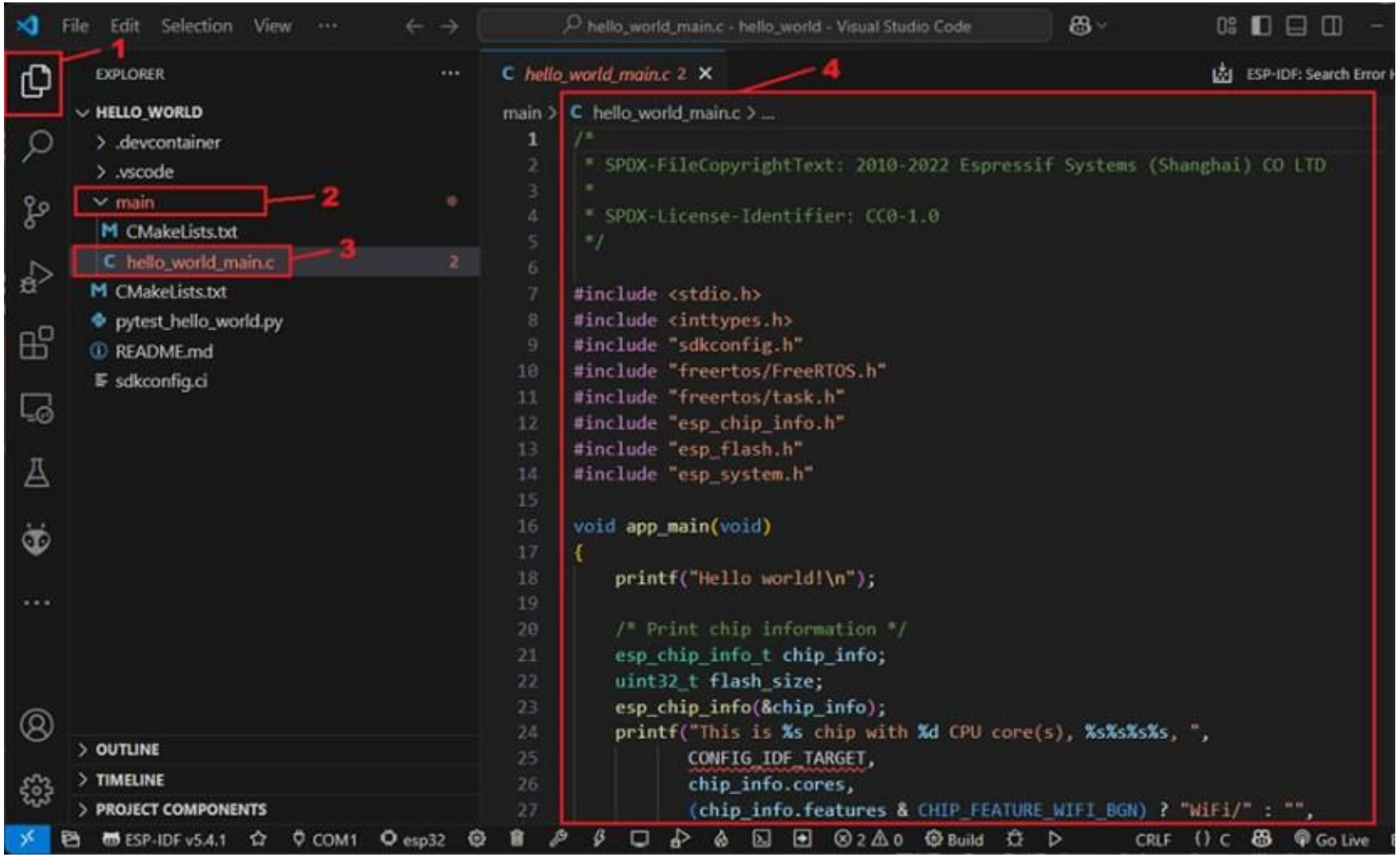
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

새 Hello World 프로젝트 의 폴더를 선택한 후 해당 파일에 대한 접근을 허용해야 합니다. " 예, 작성자를 신뢰합니다 " 버튼을 눌러 올바른 권한으로 계속 진행하세요.



ESP-IDF ESP32 프로그래밍하기 – Hello World Code

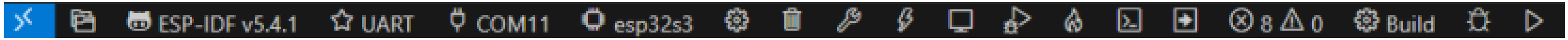
코드 나오지 않을 경우 여기서 카피 https://fishpoint.tistory.com/12218



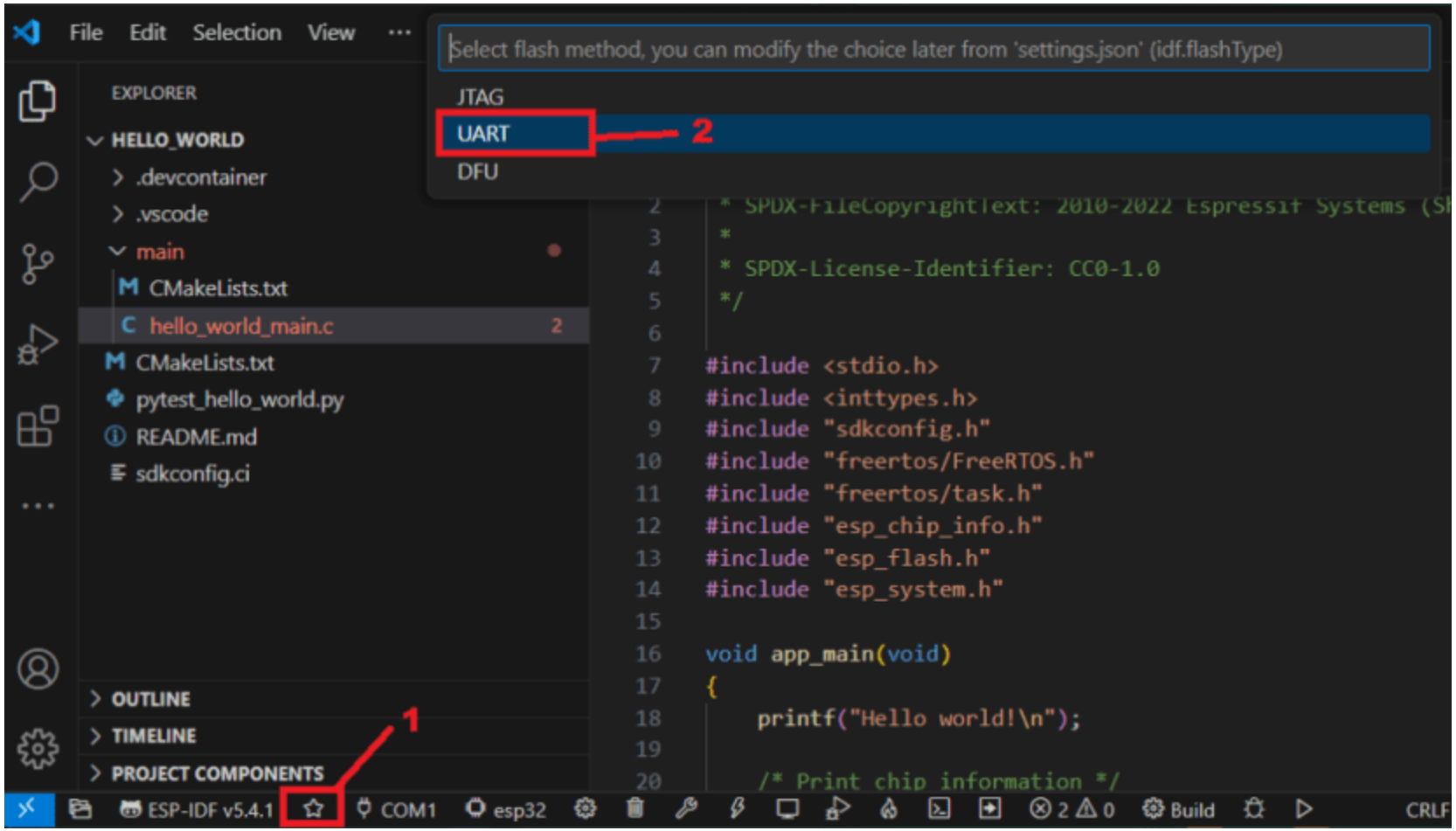
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

플래싱 방식(UART), COM 포트 번호, 대상 장치(ESP32)를 선택하고, 코드를 빌드한 후 보드에 플래싱해야 합니다.

이 모든 명령은 VS Code 하단 메뉴 막대에서 사용할 수 있습니다.

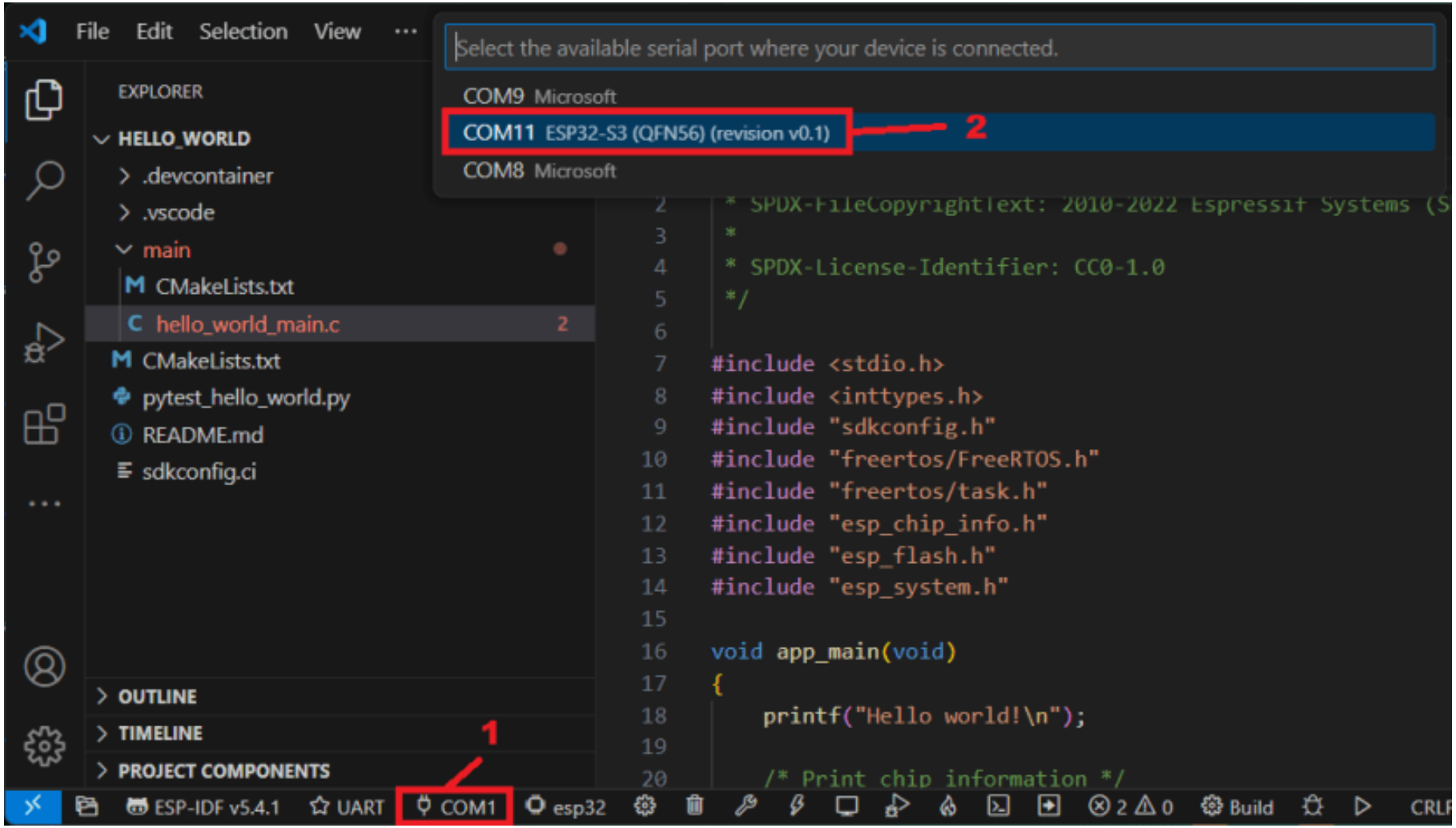


" 별 "아이콘을 클릭하고 플래시 방식을 UART 로 선택합니다 .



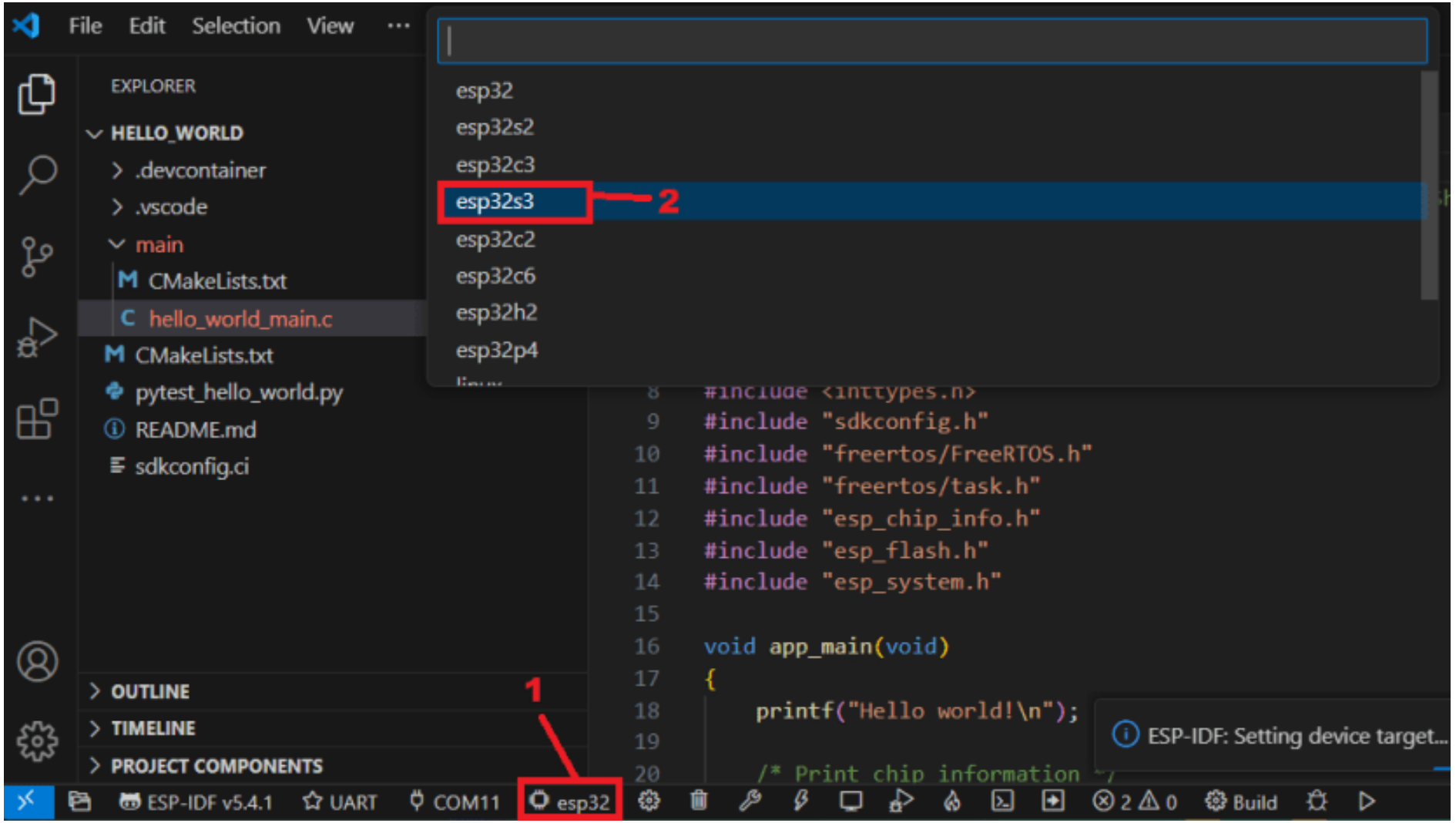
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

ESP32 보드를 컴퓨터에 연결한 상태에서 COM 포트(플러그 아이콘)를 클릭하고 ESP32를 나타내는 올바른 포트 번호를 선택합니다.



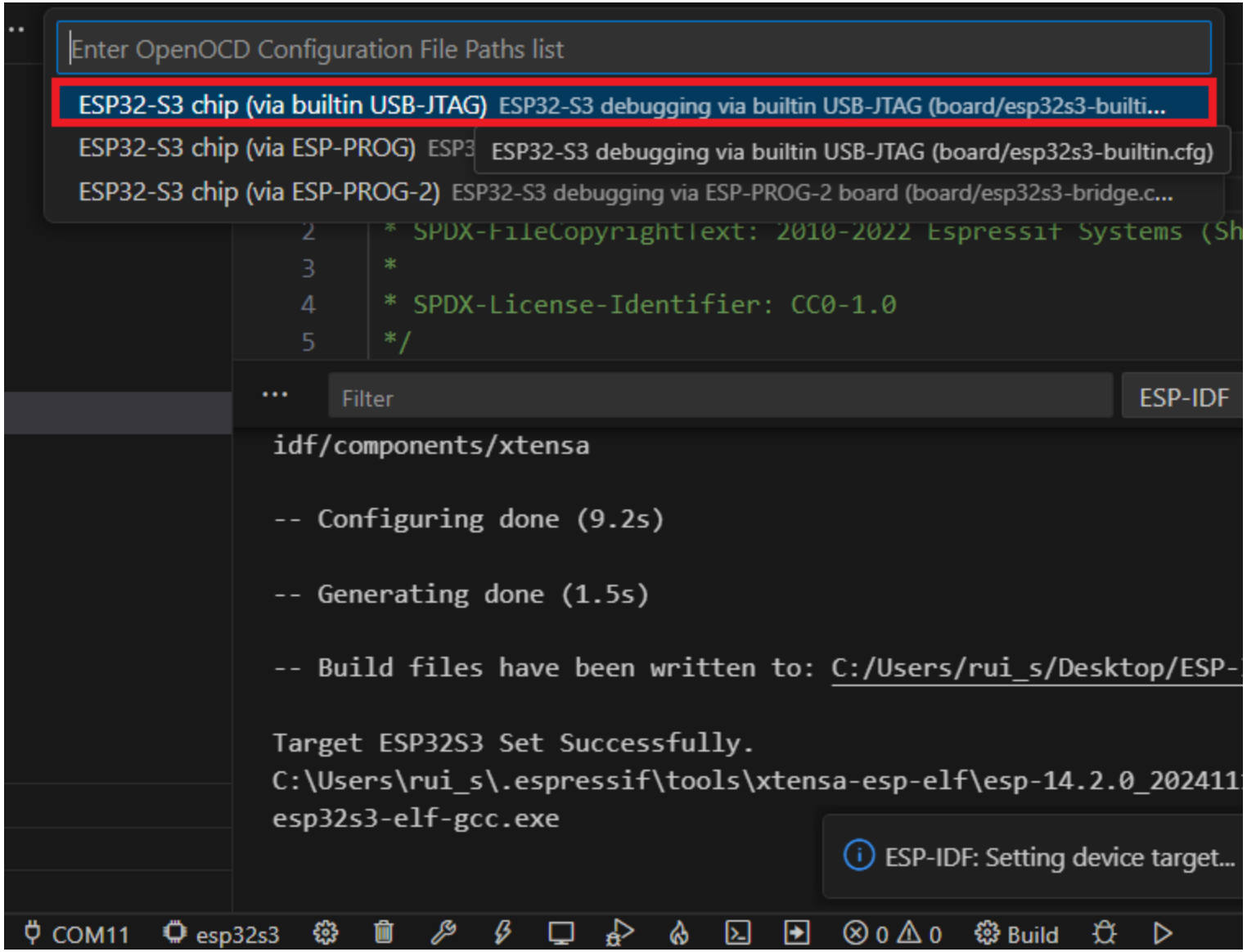
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

대상 장치도 선택해야 합니다. 하단 바의 칩 아이콘을 클릭하세요. 제 경우에는 esp32s3 칩이 장착된 ESP32를 사용하고 있습니다.



ESP-IDF ESP32 프로그래밍하기 – Hello World Code

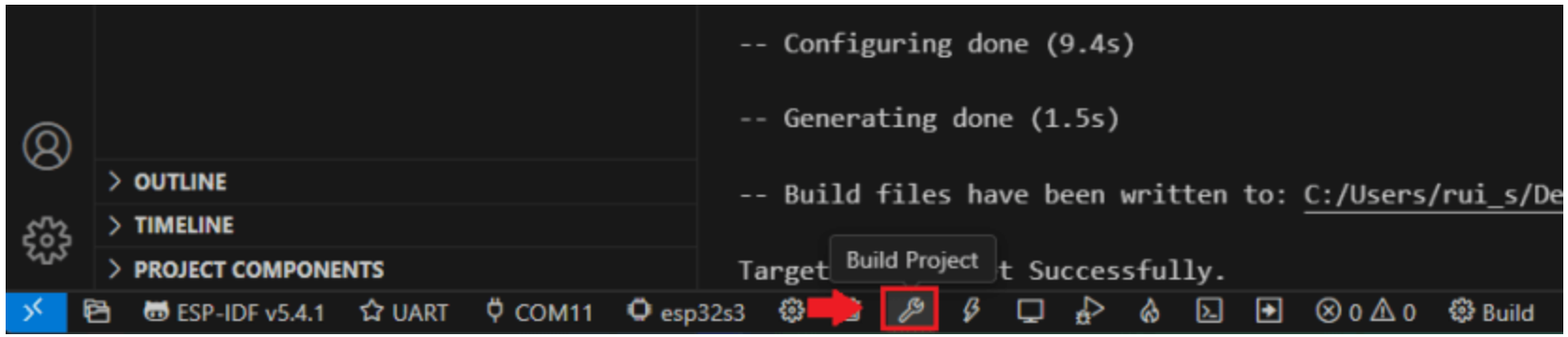
ESP-IDF 내장 USB JTAG 대상 장치를 통해 ESP32 S3 칩을 선택합니다.



ESP-IDF ESP32 프로그래밍하기 – Hello World Code

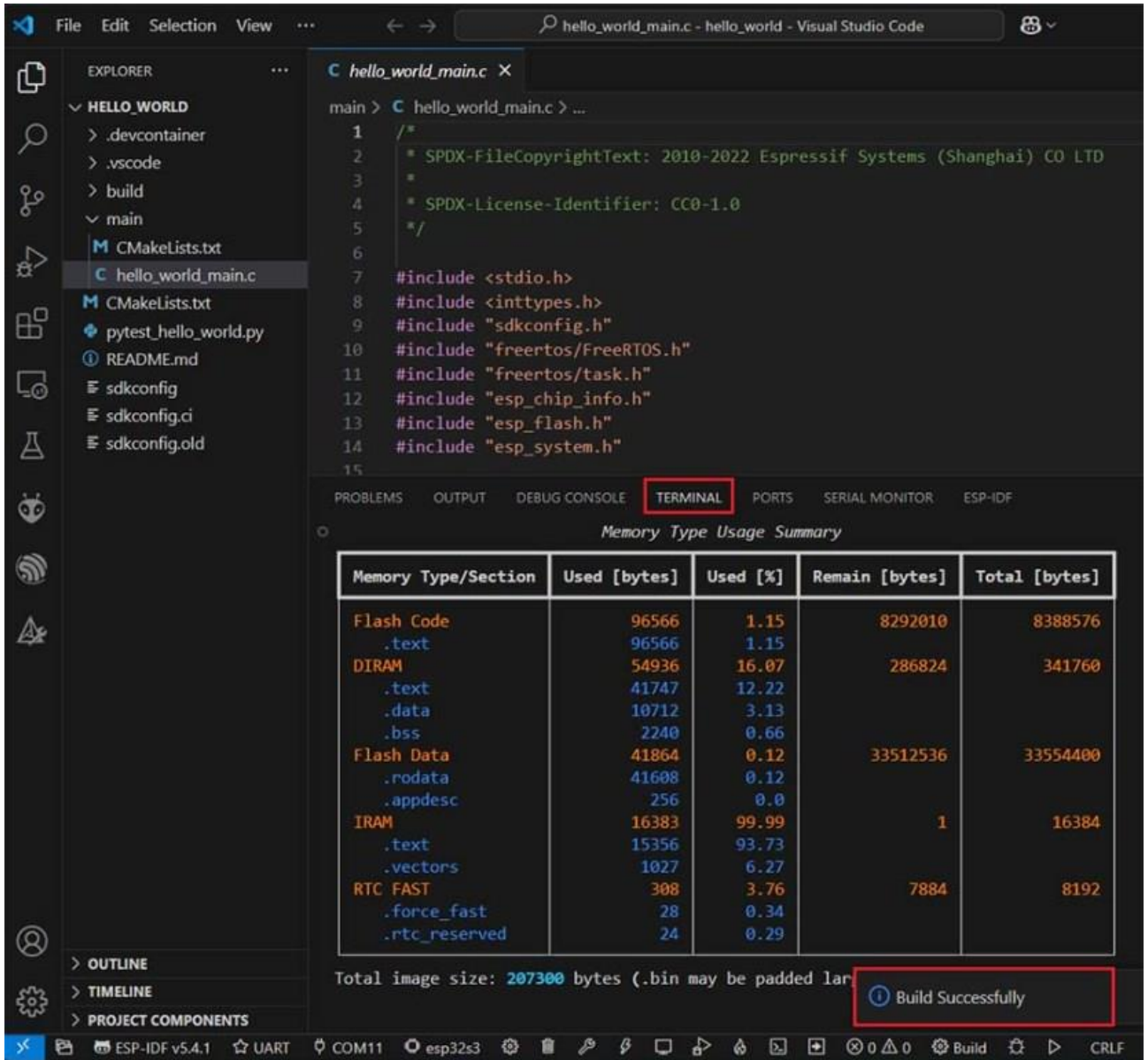
VS Code 하단의 명령 모음에도 비슷한 옵션이 선택되어 있어야 합니다.

아래 이미지에 표시된 것처럼 렌치 아이콘을 클릭하여 프로젝트를 빌드



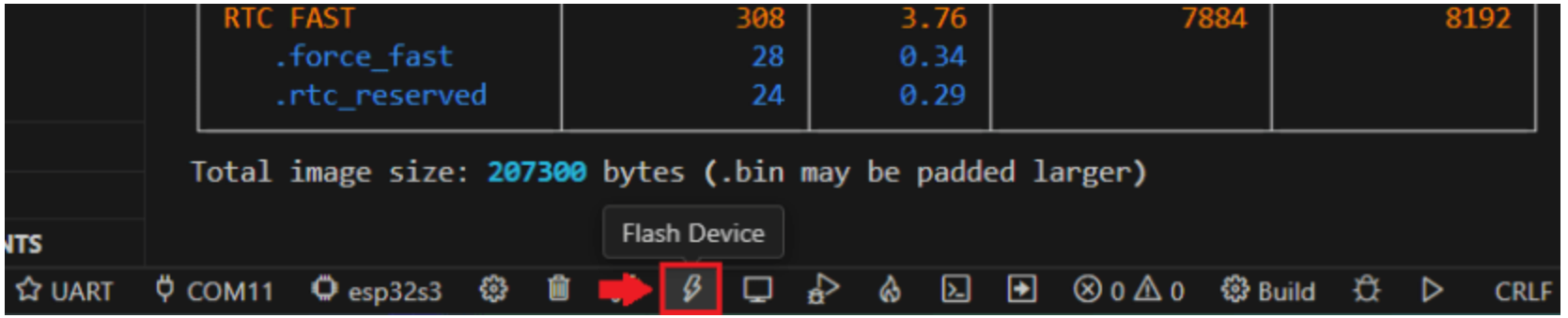
ESP-IDF ESP32 프로그래밍하기 – Hello World Code

VS Code 빌드 Hello World 프로젝트 ESP32 ESP-IDF 성공 메시지

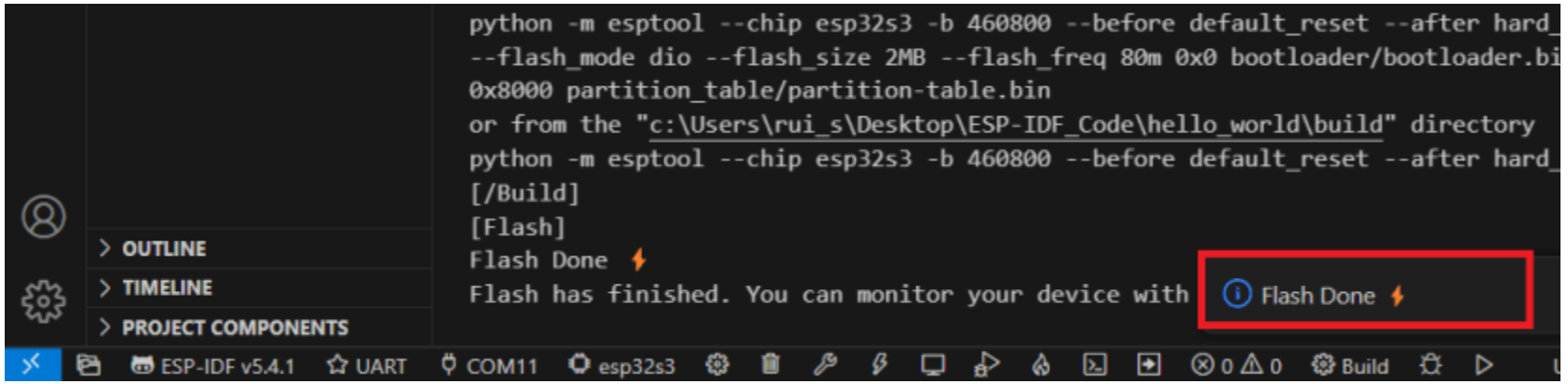


ESP-IDF ESP32 프로그래밍하기 – Hello World Code

천둥 아이콘이 있는 "장치 플래시 " 버튼을 클릭하여 Hello World ESP-IDF 프로젝트를 ESP32에 플래시



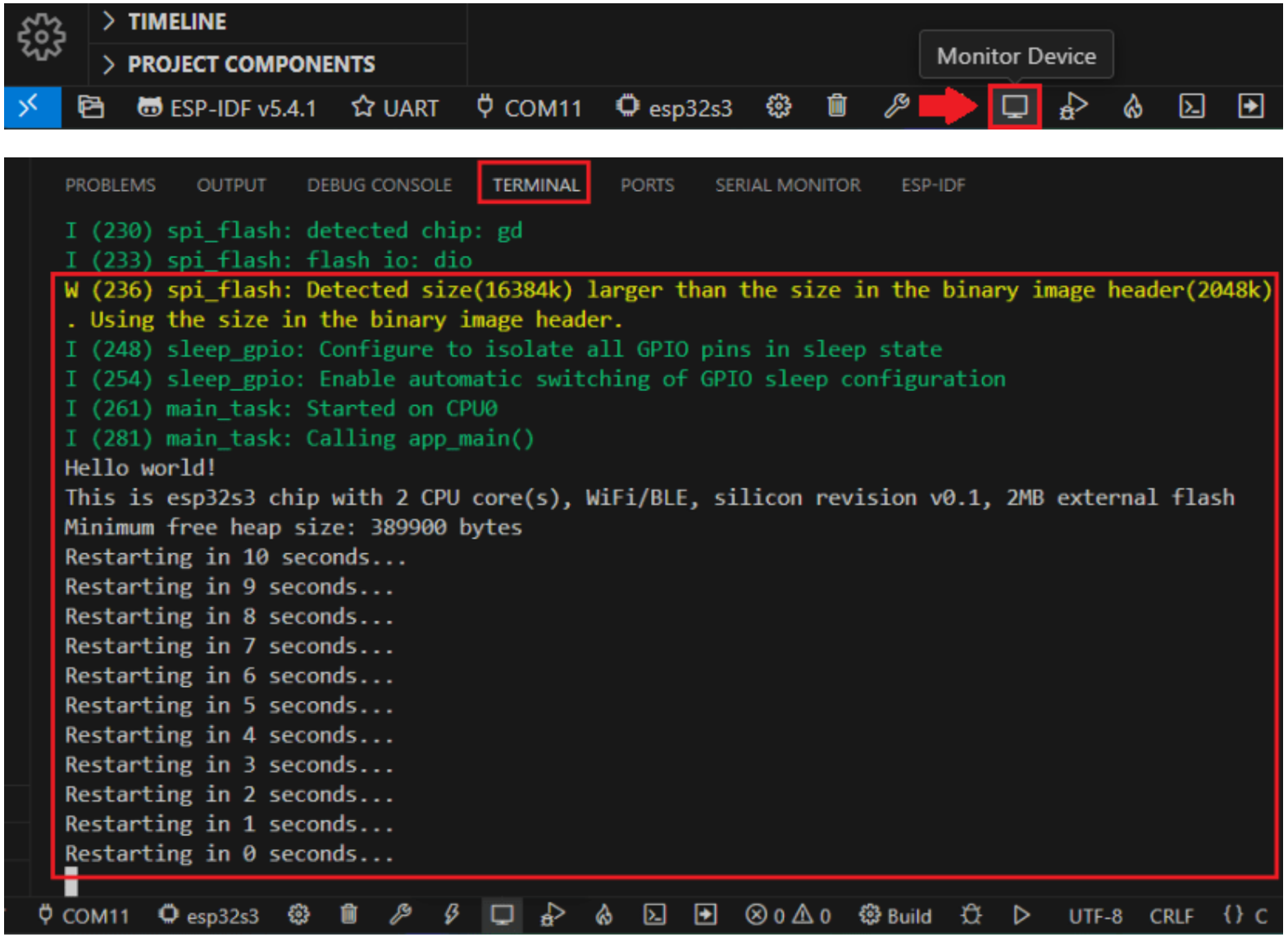
보드에 따라 ESP32의 온보드 BOOT 버튼을 눌러 플래싱 모드로 전환해야 할 수도 있습니다. 프로세스가 완료되면 " 플래시 완료 "라는 정보 메시지가 팝업됩니다.



ESP-IDF ESP32 프로그래밍하기 – Hello World Code

화면 아이콘으로 표시된 " 장치 모니터링 " 도구를 클릭

https://fishpoint.tistory.com/12218



ESP32 Wi-Fi

Care-lab.kr

유용한 Wi-Fi 라이브러리

Wi-Fi 라이브러리 포함하기 - #include <WiFi.h>

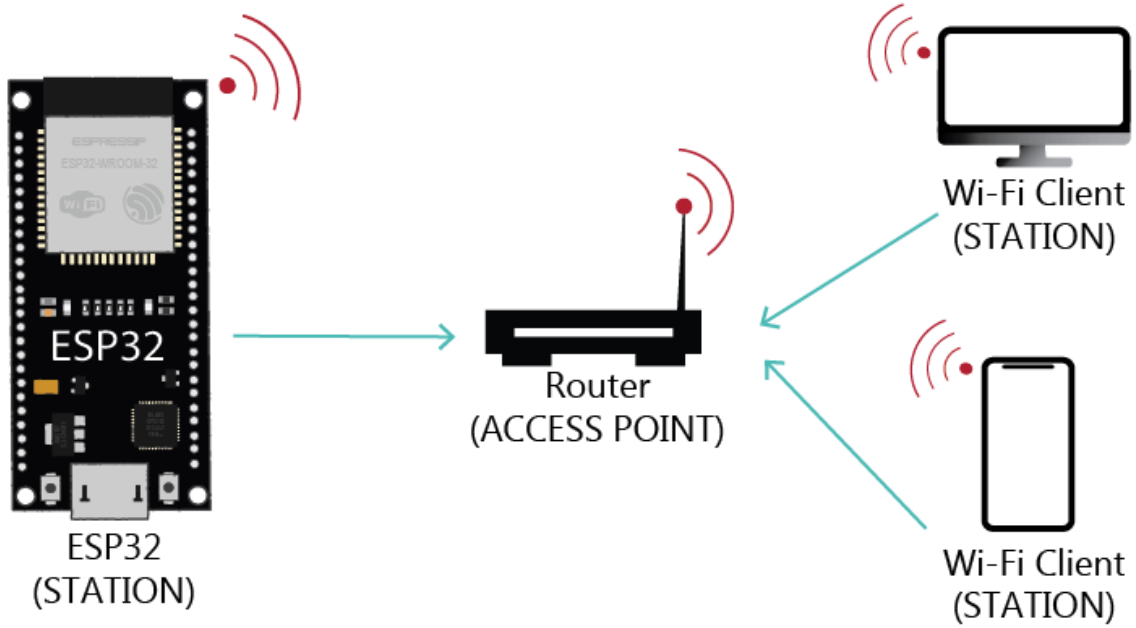
Wi-Fi 모드: 스테이션, 액세스 포인트 및 둘 다(스테이션 + 액세스 포인트);
Wi-Fi 네트워크 스캔;
Wi-Fi 네트워크 연결;
Wi-Fi 연결 상태 확인;
Wi-Fi 연결 강도 확인;
ESP32 IP 주소 확인;
ESP32 고정 IP 주소 설정;
Wi-Fi 연결 해제;
연결 끊김 후 Wi-Fi 다시 연결;
ESP32 Wi-Fi 이벤트;
연결 끊김 후 Wi-Fi 네트워크 다시 연결(Wi-Fi 이벤트);
ESP32 WiFiMulti
ESP32 호스트 이름 변경

참고 문서
https://fishpoint.tistory.com/11170
https://fishpoint.tistory.com/10981

유용한 Wi-Fi 라이브러리

Wi-Fi 모드: 스테이션, 액세스 포인트 및 둘 다(스테이션 + 액세스 포인트);

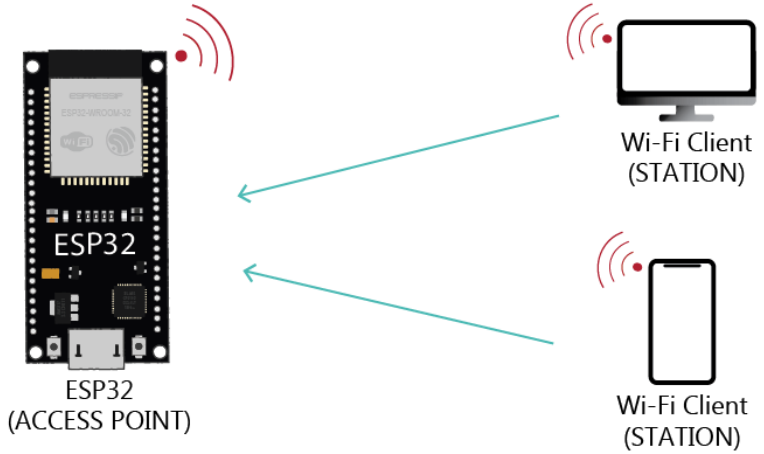
WiFi.mode(WIFI_STA)	station mode: the ESP32 connects to an
WiFi.mode(WIFI_AP)	access point mode: stations can connect to the ESP32
WiFi.mode(WIFI_AP_STA)	access point and a station connected to another access point



네트워크가 없어도 ESP에 연결하여 제어해야 하는 경우에는 이 구성이 적합하지 않을 수 있습니다. 이런 경우는 ESP 보드를 액세스 포인트로 설정해야 합니다.

유용한 Wi-Fi 라이브러리

액세스 포인트



WiFi.mode(WIFI_AP);

WiFi.softAP(ssid, password);

WiFi.softAP(const char* ssid, const char* password, int channel, int ssid_hidden, int max_connection)

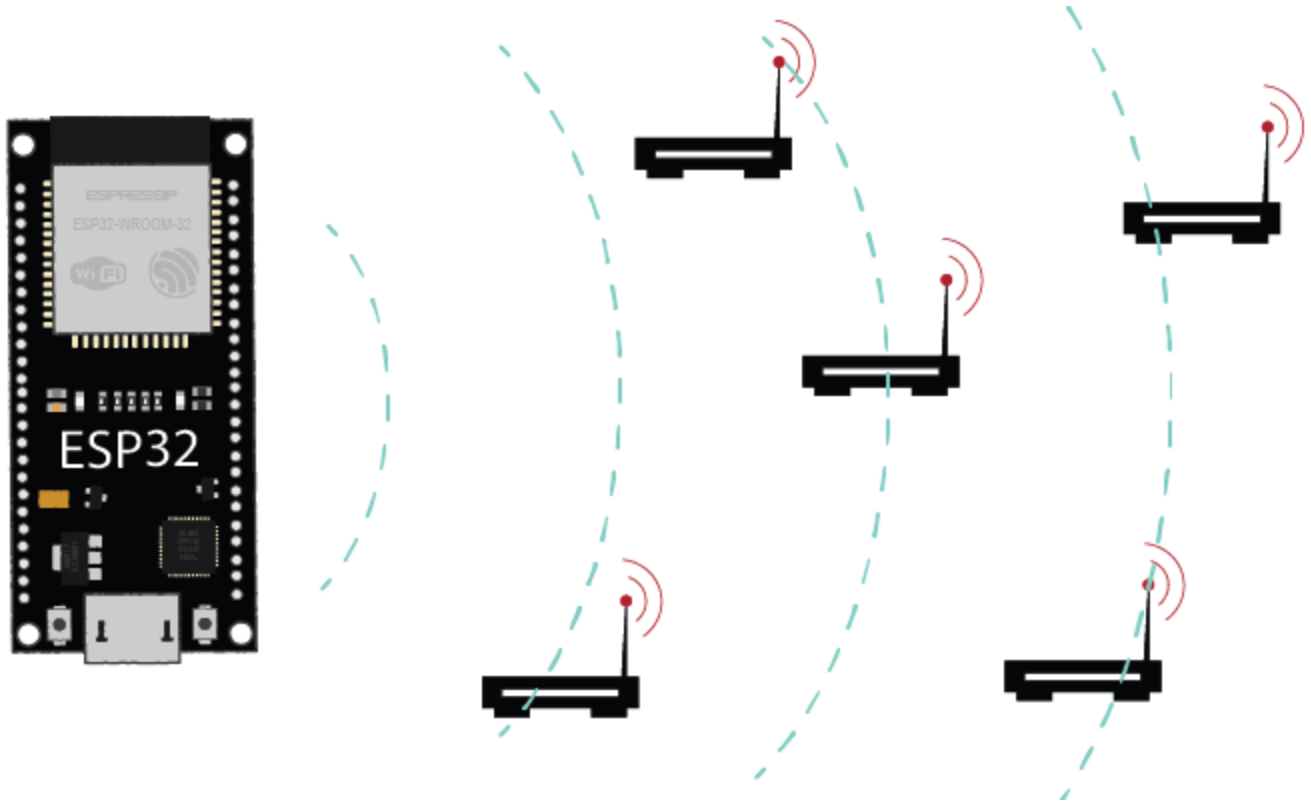
- ssid: 액세스 포인트 이름 - 최대 63자;
- password: 최소 8자; 액세스 포인트를 개방하려면 NULL로 설정;
- channel: Wi-Fi 채널 번호(1-13)
- ssid_hidden: (0 = 브로드캐스트 SSID, 1 = SSID 숨기기)
- max_connection: 최대 동시 연결 클라이언트 수(1-4)

유용한 Wi-Fi 라이브러리

Wi-Fi 스테이션 + 액세스 포인트

WiFi.mode(WIFI_AP_STA);

ESP32는 Wi-Fi 범위 내에서 근처 Wi-Fi 네트워크를 스캔할 수 있습니다. Arduino IDE에서 파일 > 예제 > WiFi > WiFiScan으로 이동하세요. 그러면 ESP32 보드 범위 내의 Wi-Fi 네트워크를 스캔하는 스케치가 로드됩니다.



유용한 Wi-Fi 라이브러리

Wi-Fi 네트워크 스캔;

int n = WiFi.scanNetworks();

WiFi.SSID()는 특정 네트워크의 SSID를 출력

WiFi.RSSI()는 해당 네트워크의 RSSI를 반환 - 절댓값이 낮을수록 Wi-Fi 연결이 가장 강력함을 의미합니다.

WiFi.encryptionType()은 네트워크 암호화 유형을 반환

- WIFI_AUTH_OPEN
- WIFI_AUTH_WEP
- WIFI_AUTH_WPA_PSK
- WIFI_AUTH_WPA2_PSK
- WIFI_AUTH_WPA_WPA2_PSK
- WIFI_AUTH_WPA2_ENTERPRISE

유용한 Wi-Fi 라이브러리

Wi-Fi 네트워크 연결;

```
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}
```

유용한 Wi-Fi 라이브러리

Wi-Fi 연결 상태 확인;

while (WiFi.status() != WL_CONNECTED)

Value	Constant	Meaning
0	WL_IDLE_STATUS	temporary status assigned when WiFi.begin() is called
1	WL_NO_SSID_AVAIL	when no SSID are available
2	WL_SCAN_COMPLETED	scan networks is completed
3	WL_CONNECTED	when connected to a WiFi network
4	WL_CONNECT_FAILED	when the connection fails for all the attempts
5	WL_CONNECTION_LOST	when the connection is lost
6	WL_DISCONNECTED	when disconnected from a network

0 WL_IDLE_STATUS: WiFi.begin()이 호출될 때 할당되는 임시 상태입니다.
1 WL_NO_SSID_AVAIL: 사용 가능한 SSID가 없는 경우입니다.
2 WL_SCAN_COMPLETED: 네트워크 스캔이 완료된 경우입니다.
3 WL_CONNECTED: WiFi 네트워크에 연결된 경우입니다.
4 WL_CONNECT_FAILED: 모든 시도에도 연결이 실패한 경우입니다.
5 WL_CONNECTION_LOST: 연결이 끊어진 경우입니다.
6 WL_DISCONNECTED: 네트워크 연결이 끊어진 경우입니다.

유용한 Wi-Fi 라이브러리

ESP32 IP 주소 확인;

```
Serial.println(WiFi.localIP());
```

ESP32 고정 IP 주소 설정;

```
// Set your Static IP address
IPAddress local_IP(192, 168, 1, 184);

// Set your Gateway IP address
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);    // optional
IPAddress secondaryDNS(8, 8, 4, 4); // optional
```

setup()에서 WiFi.config() 메서드를 호출하여 ESP32에 구성을 할당

```
// Configures static IP address
if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
  Serial.println("STA Failed to configure");
}
```

유용한 Wi-Fi 라이브러리

Wi-Fi 연결 해제;

```
WiFi.disconnect()
```

연결 끊김 후 Wi-Fi 다시 연결;

```
WiFi.reconnect() 혹은


WiFi.disconnect();
WiFi.begin(ssid, password);
```

연결되었는지 주기적으로 확인

```
unsigned long currentMillis = millis();
// if WiFi is down, try reconnecting
if ((WiFi.status() != WL_CONNECTED) && (currentMillis - previousMillis >=interval)) {
  Serial.print(millis());
  Serial.println("Reconnecting to WiFi...");
  WiFi.disconnect();
  WiFi.reconnect();
  previousMillis = currentMillis;
}d
```

유용한 Wi-Fi 라이브러리

ESP32 Wi-Fi 이벤트;

이벤트를 사용하는 방법에 대한 전체 예제를 보려면 Arduino IDE에서 파일 > 예제 > WiFi > WiFiClientEvents로 이동하세요

연결 끊김 후 Wi-Fi 네트워크 다시 연결(Wi-Fi 이벤트);

Wi-Fi 이벤트는 연결이 끊어졌음을 감지하고 바로 재연결을 시도하는 데 유용할 수 있습니다
(SYSTEM_EVENT_AP_STADISCONNECTED 이벤트 사용)

ESP32가 연결될 때, IP 주소를 받을 때, 연결이 끊어질 때의 세 가지 Wi-Fi 이벤트 추가

```
WiFi.onEvent(WiFiStationConnected, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_CONNECTED);
WiFi.onEvent(WiFiGotIP, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_GOT_IP);
WiFi.onEvent(WiFiStationDisconnected, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_DISCONNECTED);
```

유용한 Wi-Fi 라이브러리

ESP32 WiFiMulti;

ESP32 WiFiMulti를 사용하면 여러 네트워크(SSID/비밀번호 조합)를 등록할 수 있습니다. ESP32는 가장 강한 신호(RSSI)를 가진 Wi-Fi 네트워크에 연결합니다. 연결이 끊어지면 목록의 다음 네트워크에 연결됩니다. 이를 위해 WiFiMulti.h 라이브러리를 포함해야 합니다(설치할 필요는 없으며 ESP32 패키지에 기본적으로 포함되어 있습니다).

ESP32 호스트 이름 변경

보드에 사용자 지정 호스트 이름을 설정하려면 WiFi.begin(); 전에 WiFi.setHostname(YOUR_NEW_HOSTNAME);을 호출합니다.

기본 ESP32 호스트 이름은 espressif입니다.

먼저 새 호스트 이름을 정의합니다. 예: String hostname = "ESP32 Node Temperature";

그런 다음 WiFi.begin()을 호출하기 전에 WiFi.setHostname() 함수를 호출합니다. 아래와 같이 WiFi.config()도 호출해야 합니다.

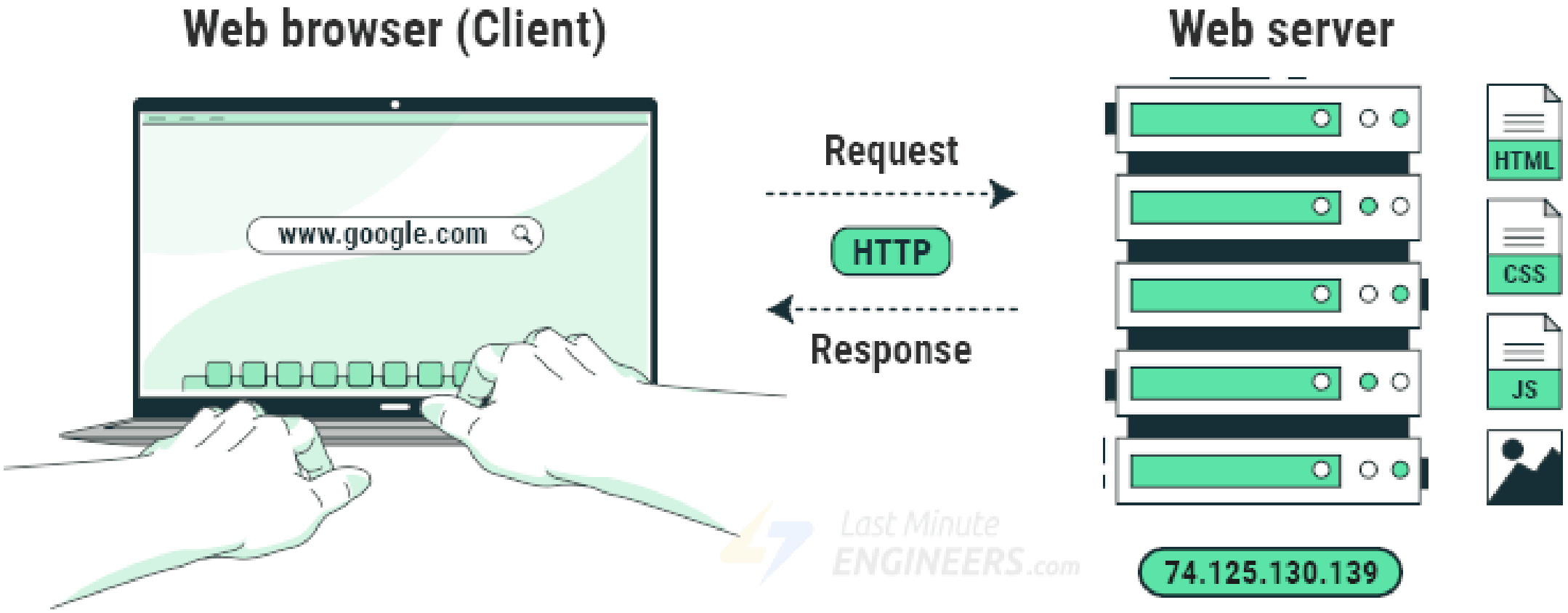
```
WiFi.config(INADDR_NONE, INADDR_NONE, INADDR_NONE, INADDR_NONE);
WiFi.setHostname(hostname.c_str()); //define hostname
```

유용한 Wi-Fi 라이브러리 기타

- 오픈 네트워크(비밀번호 없음)인 경우 코드가 더 간단해집니다. WiFi.begin(ssid);
- 코드에 로그인과 비밀번호를 일반 텍스트로 저장하는 것은 나쁜 관행입니다. 따로 헤더 파일을 만들어 보관하세요.
- Wi-Fi 연결 문제 디버깅 - 연결 상태 모니터링, WiFi.status() 함수를 사용
- WiFi 채널 변경
- 일시적으로 MAC 주소 변경
- 연결에 실패할 경우 전력 소모를 최소화하기 위해 ESP32를 딥 슬립 모드로 설정하는 것이 좋습니다. 10초 간격
- WiFi 함수를 폴링하는 코드에서 이벤트 기반 프로그래밍을 사용

ESP32 HTTP Web Server

웹 서버는 기본적으로 웹사이트를 저장하고 이를 요청하는 사용자에게 전달하는 단 하나의 주요 임무를 가진 특수한 컴퓨터입니다. 브라우저에 웹사이트 주소를 입력하거나 링크를 클릭할 때, 여러분은 본질적으로 웹 서버에 주문을 내는 것입니다.

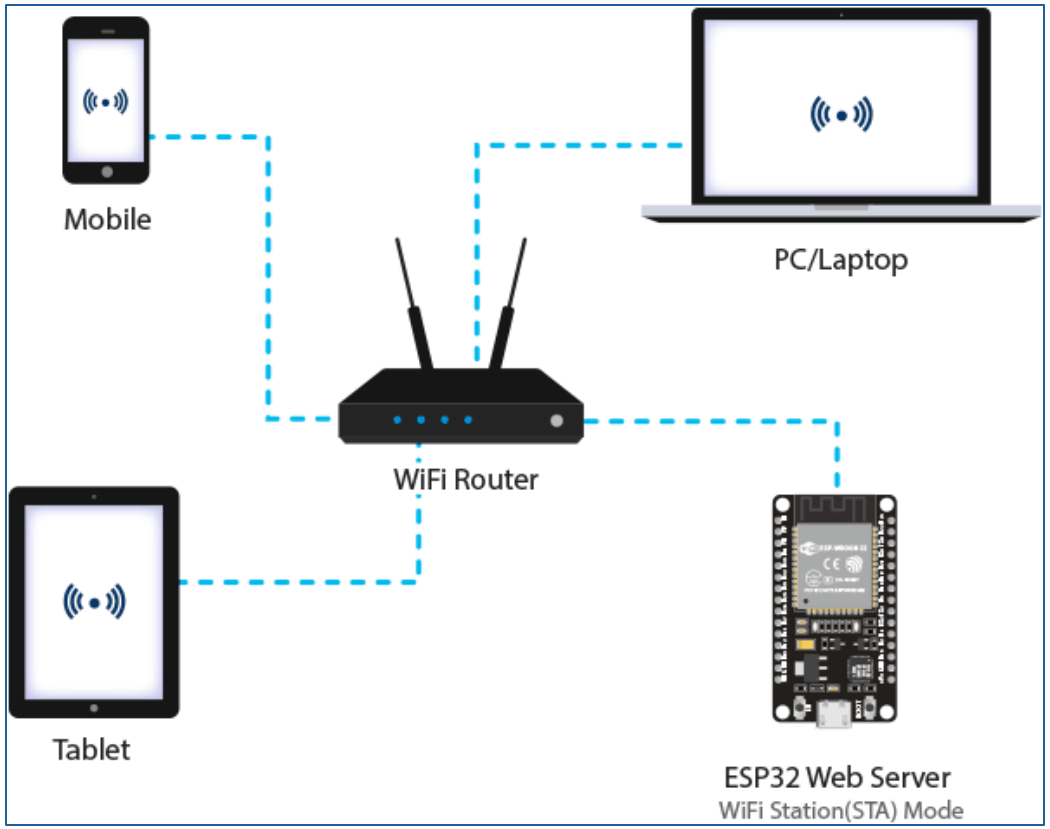


ESP32 Wi-Fi 작동 모드

ESP32가 웹 서버로 기능하여 웹 페이지를 제공하려면 먼저 세 가지 Wi-Fi 모드 중 하나를 사용하여 네트워크 연결을 설정해야 합니다: 스테이션(STA) 모드, 액세스 포인트(AP) 모드, 듀얼(AP+STA) 모드입니다.

Station (STA) Mode

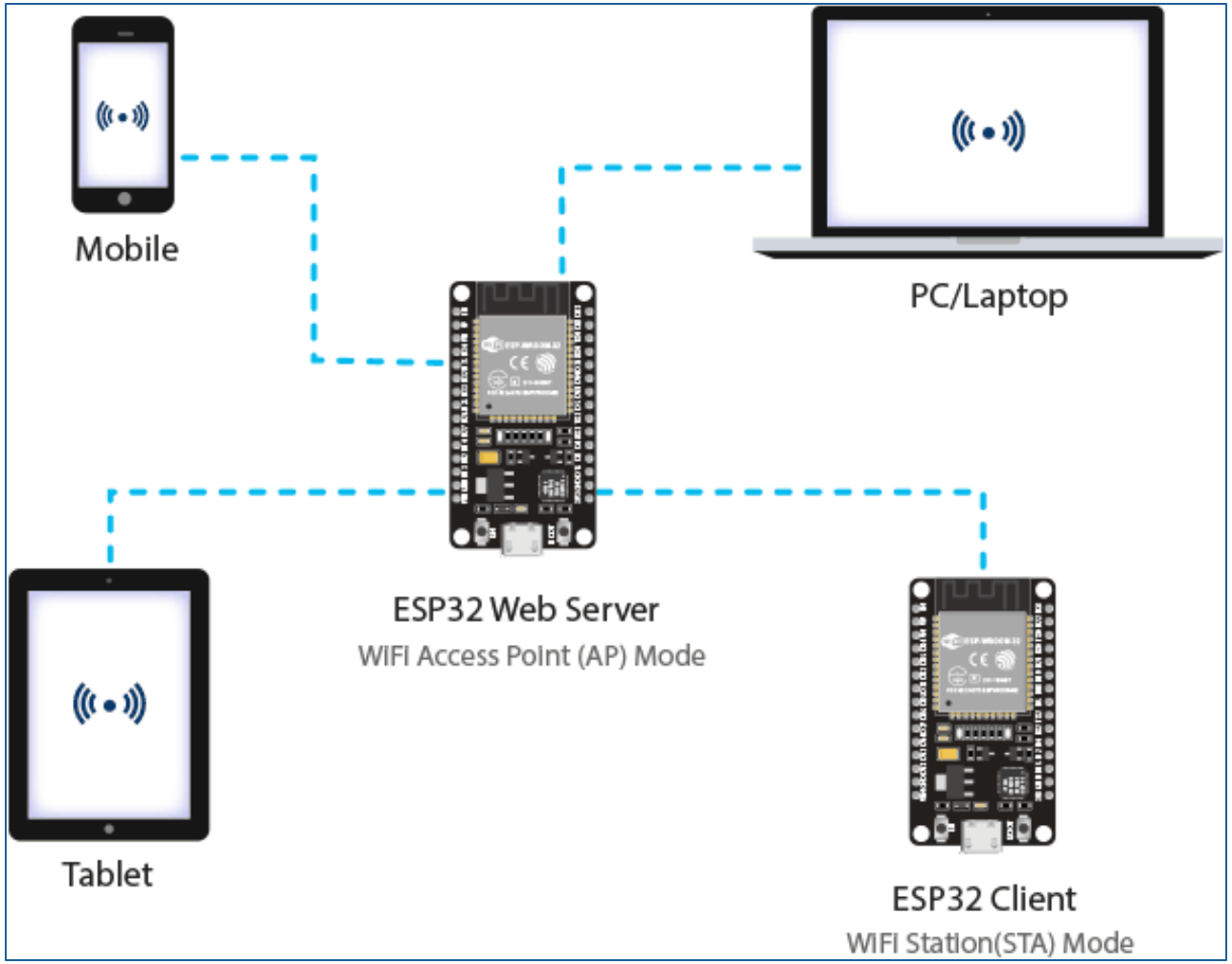
ESP32가 STA 모드일 때는 기존 Wi-Fi 네트워크에 연결하려는 노트북이나 스마트폰과 같은 다른 장치와 동일하게 작동합니다. Wi-Fi 네트워크를 검색하고, 네트워크 이름(SSID)과 비밀번호를 사용하여 연결한 후 해당 네트워크의 일부가 됩니다.



ESP32 Wi-Fi 작동 모드

Access Point (AP) Mode

ESP32가 AP 모드일 때 자체 Wi-Fi 네트워크를 생성하며, SSID(네트워크 이름), 비밀번호, IP 주소를 할당합니다. 스마트폰이나 컴퓨터 같은 다른 기기들은 이 네트워크를 검색한 후 설정한 비밀번호로 연결할 수 있습니다. 자체 할당된 IP 주소를 통해 ESP32는 새로 생성된 네트워크에 연결된 모든 기기에 웹 페이지를 직접 제공할 수 있습니다.



ESP32 Wi-Fi 작동 모드

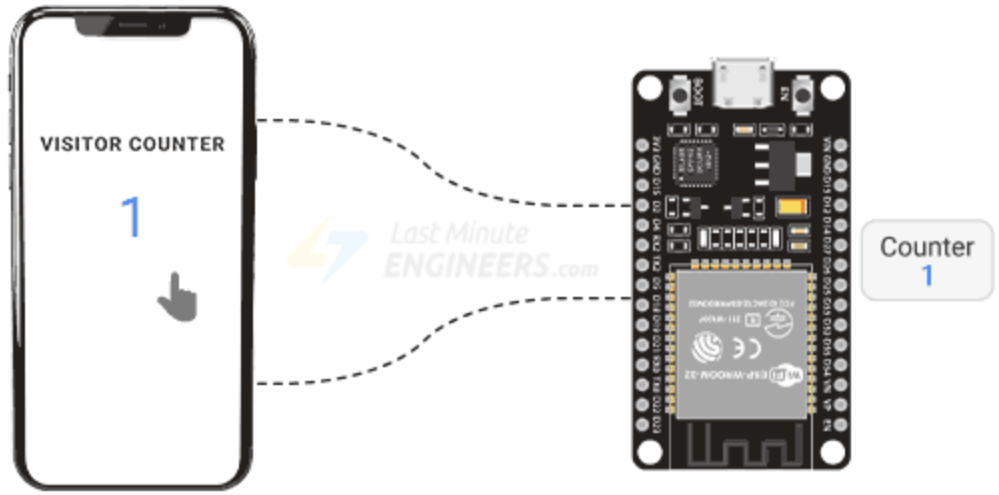
Dual Mode

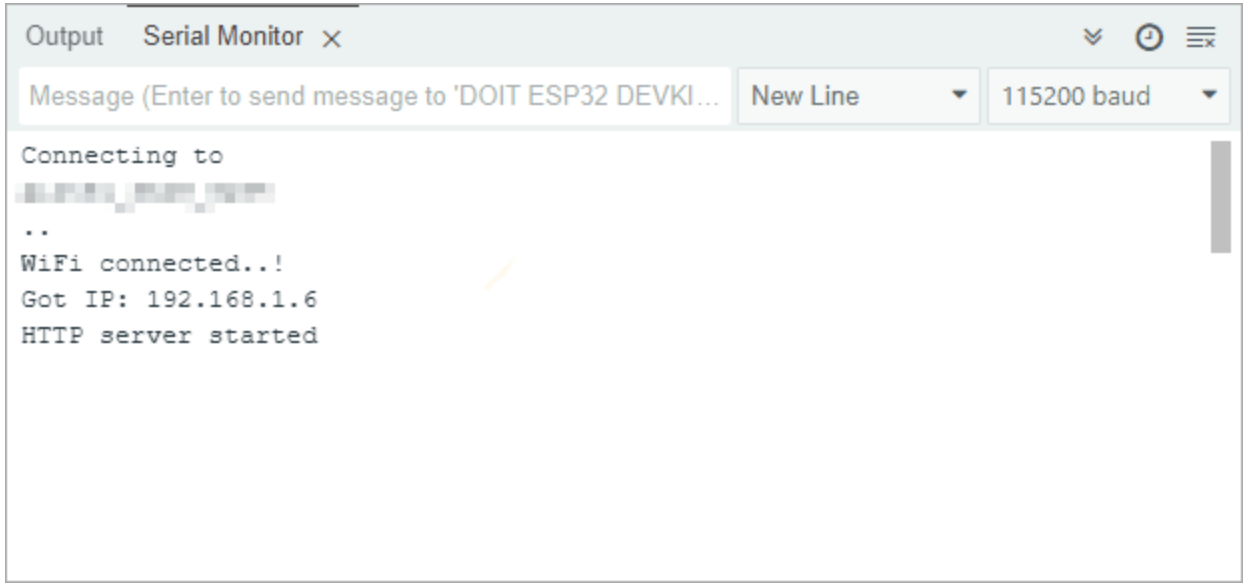
ESP32가 특히 다재다능한 이유는 실제로 두 모드를 동시에 작동할 수 있다는 점입니다.

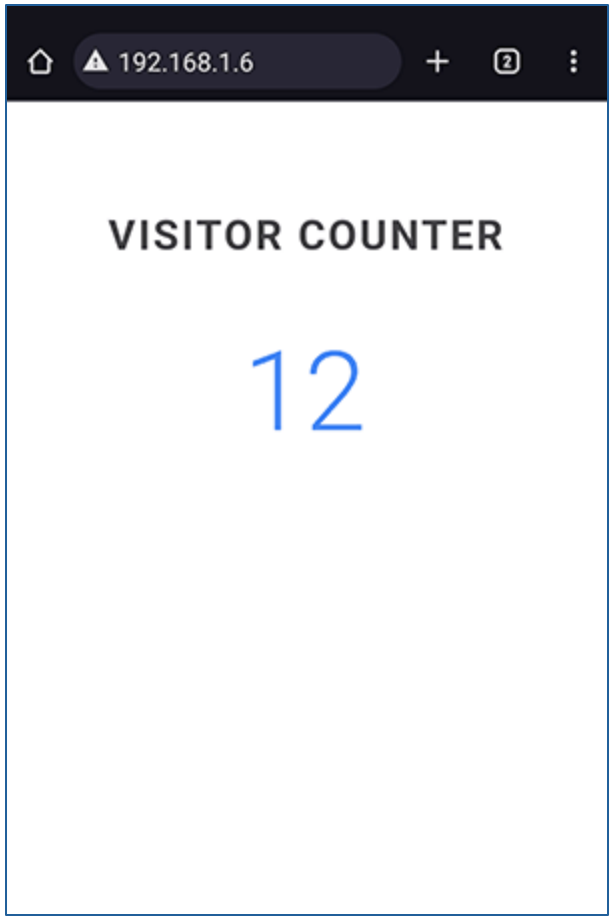
이를 AP+STA 모드 또는 듀얼 모드라고 합니다. 이 구성에서 ESP32는 스테이션으로 기존 Wi-Fi 네트워크에 연결하는 동시에 다른 기기가 연결할 수 있는 자체 액세스 포인트를 생성할 수 있습니다.

이는 Wi-Fi 커버리지를 확장하거나 서로 다른 네트워크 간에 브리지를 생성하는 기기를 구축하는 등 흥미로운 가능성을 열어줍니다.

Ex1. ESP32 Web Server in Station (STA) Mode

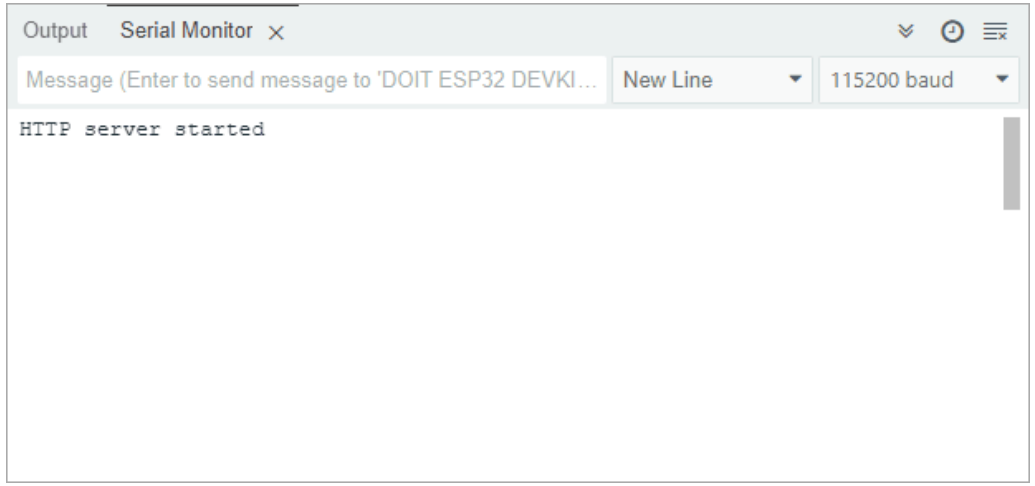


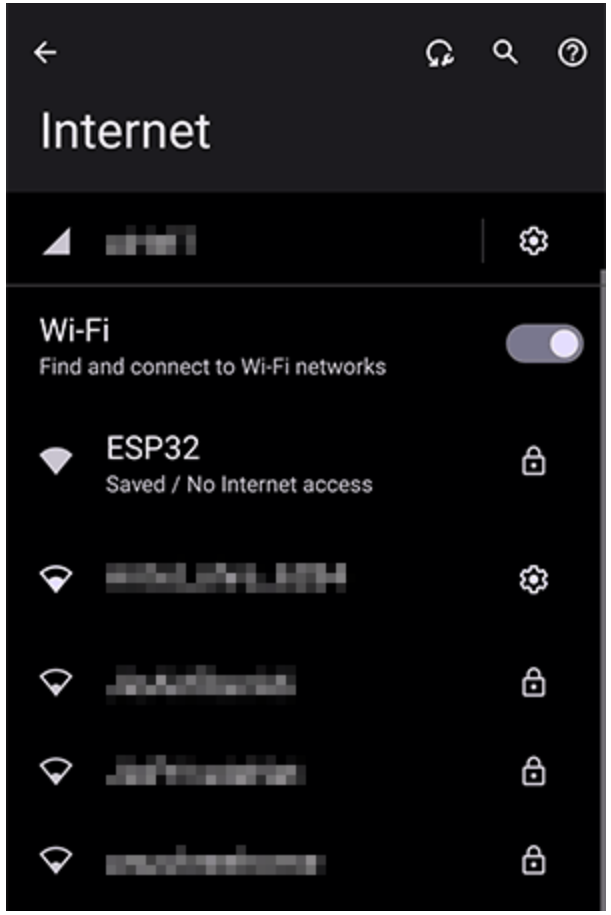


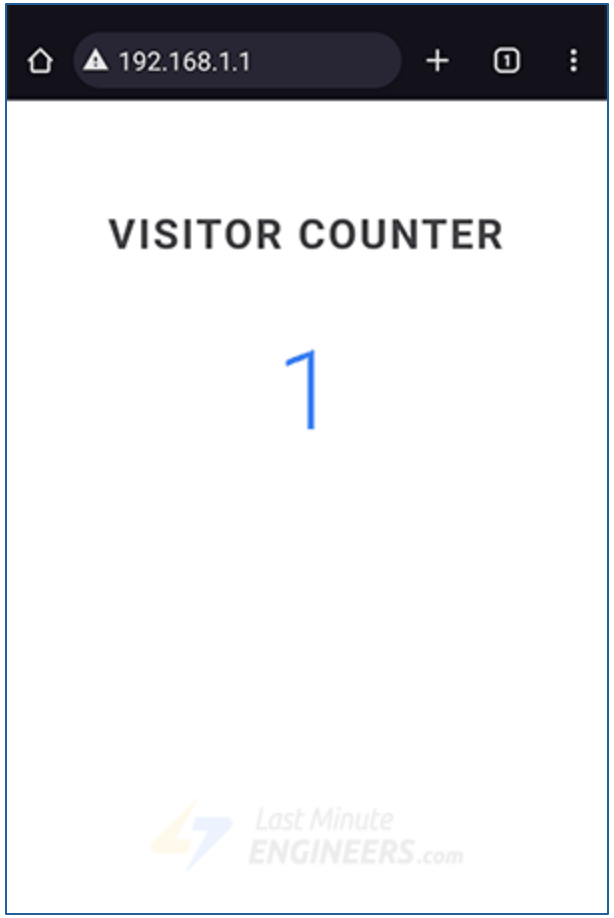


server-stationmode.ino

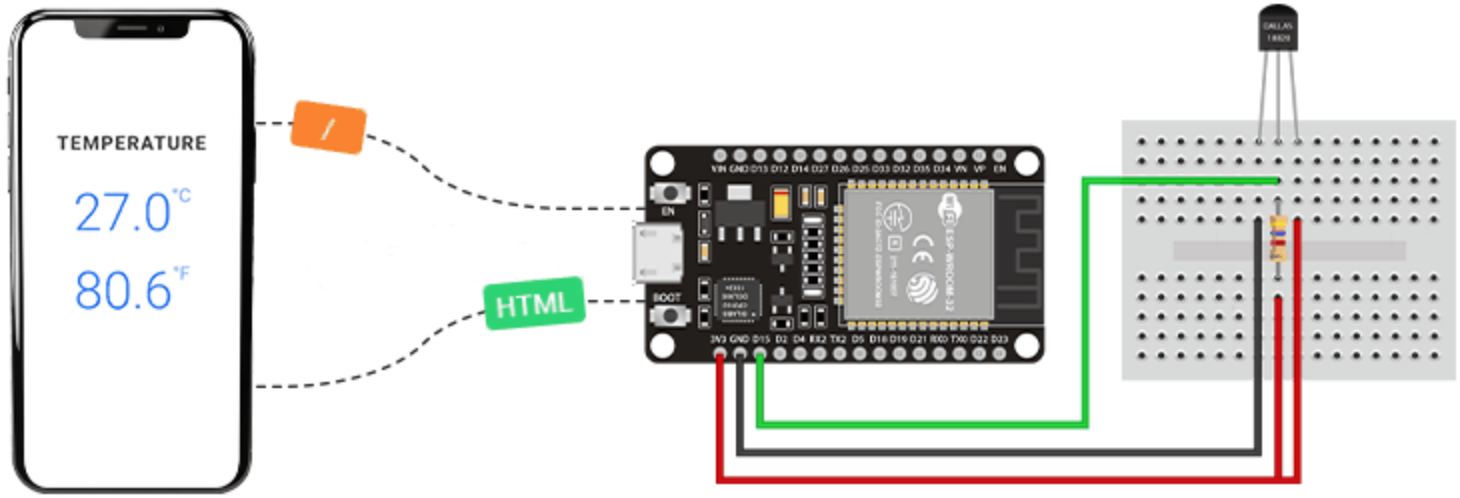
Ex2. ESP32 Web Server in Access Point (AP) mode

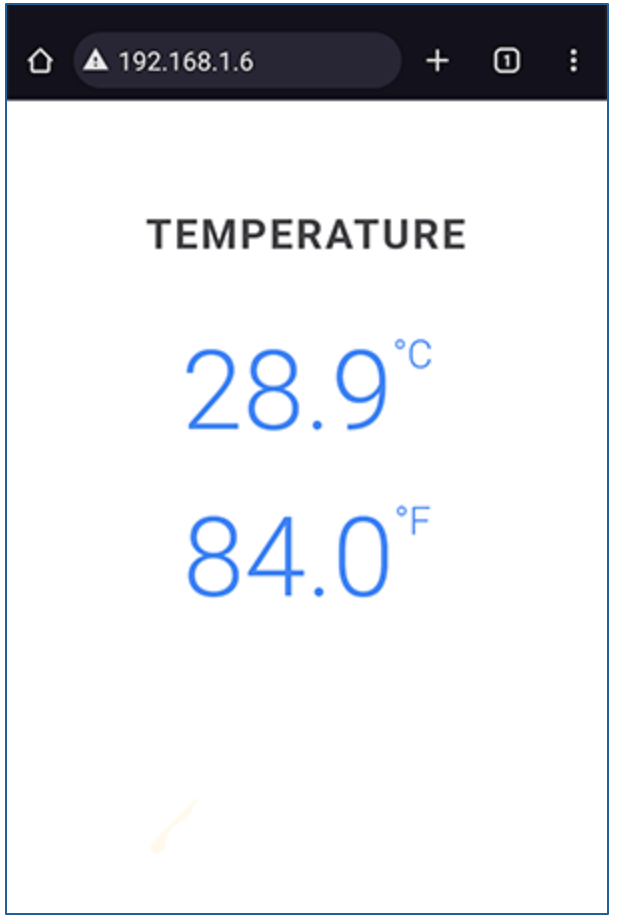




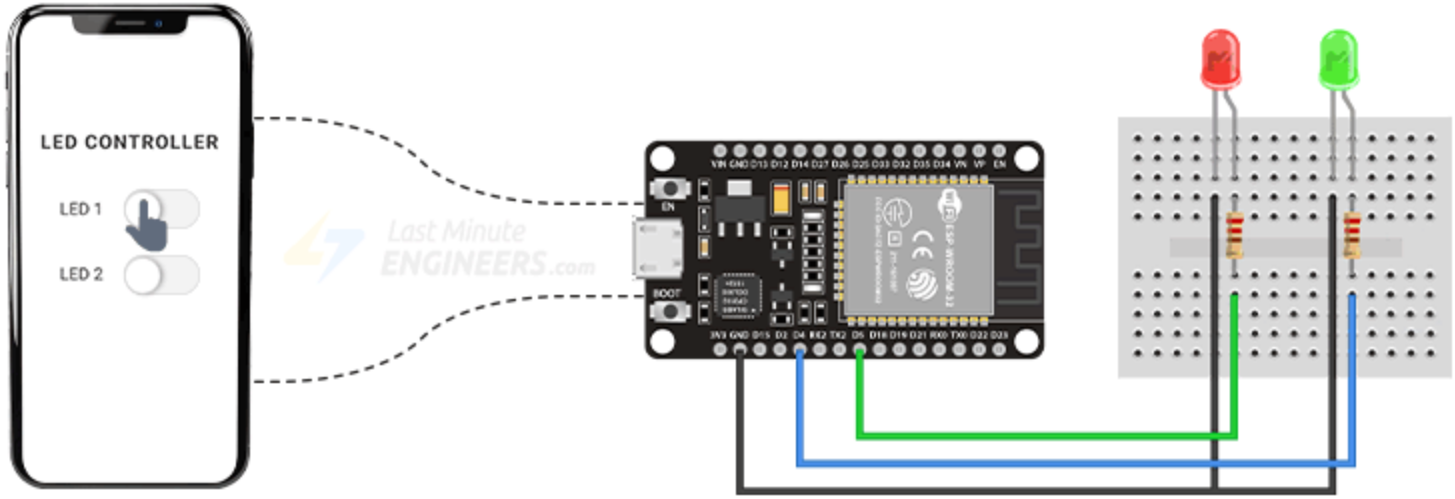


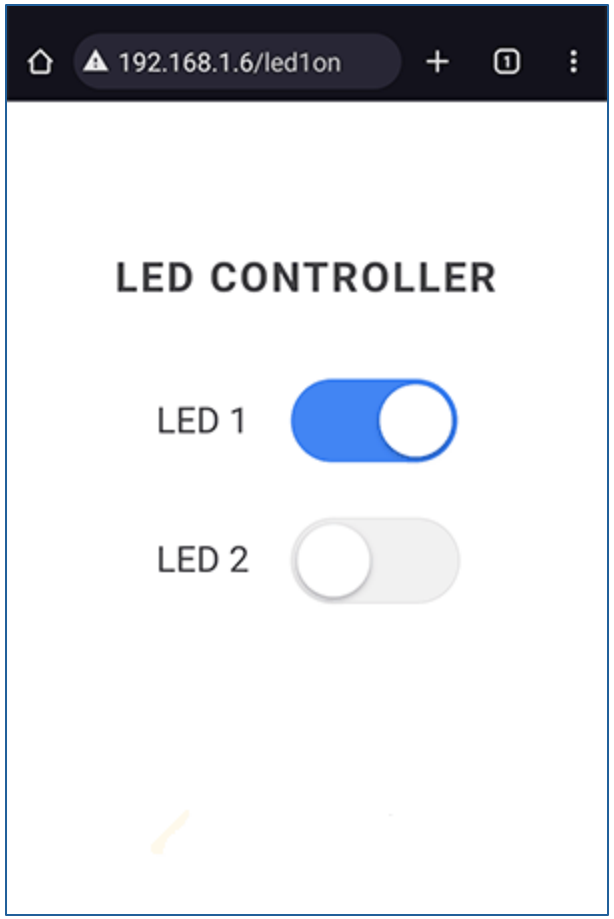
Ex3. Displaying Sensor Readings on an ESP32 Web Server





Ex4. Controlling Physical Things with a Web Server





Output Serial Monitor ×

Message (Enter to send message to 'DOIT ESP32 DEVKI... New Line 115200 baud

```
Connecting to
[redacted]
..
WiFi connected..!
Got IP: 192.168.1.6
HTTP server started
```

Output Serial Monitor ×

Message (Enter to ser New Line 115200 baud

```
Connecting to
[redacted]
..
WiFi connected..!
Got IP: 192.168.1.6
HTTP server started
LED1 Status: OFF | LED2 Status: OFF
LED1 Status: ON
```

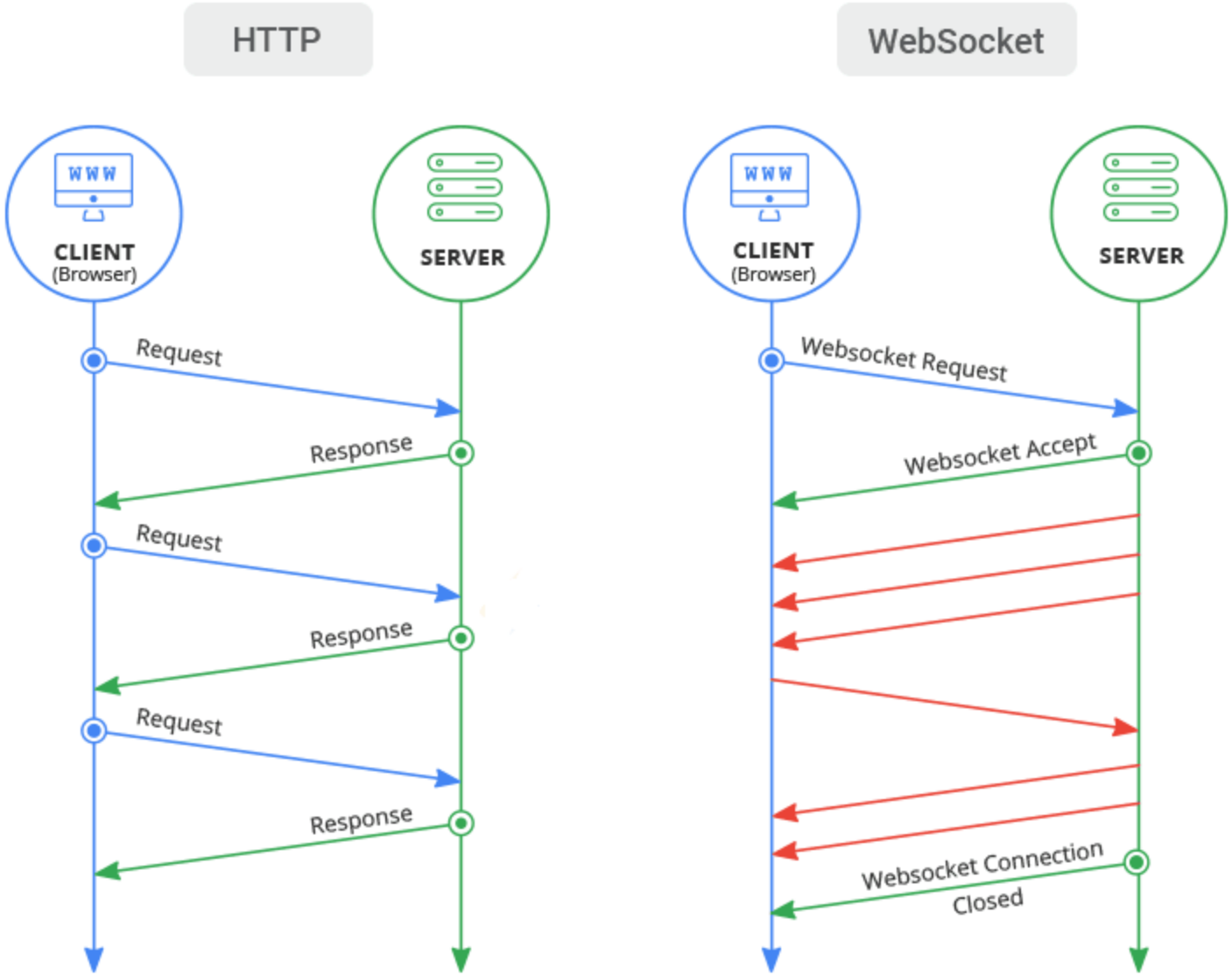
ESP32 Web Server with WebSocket

"WebSocket"이라는 이름은 언뜻 보기엔 이해하기 어려워 보일 수 있지만, 그 개념 자체는 사실 매우 간단하며 금방 기본적인 이해를 얻을 수 있습니다.

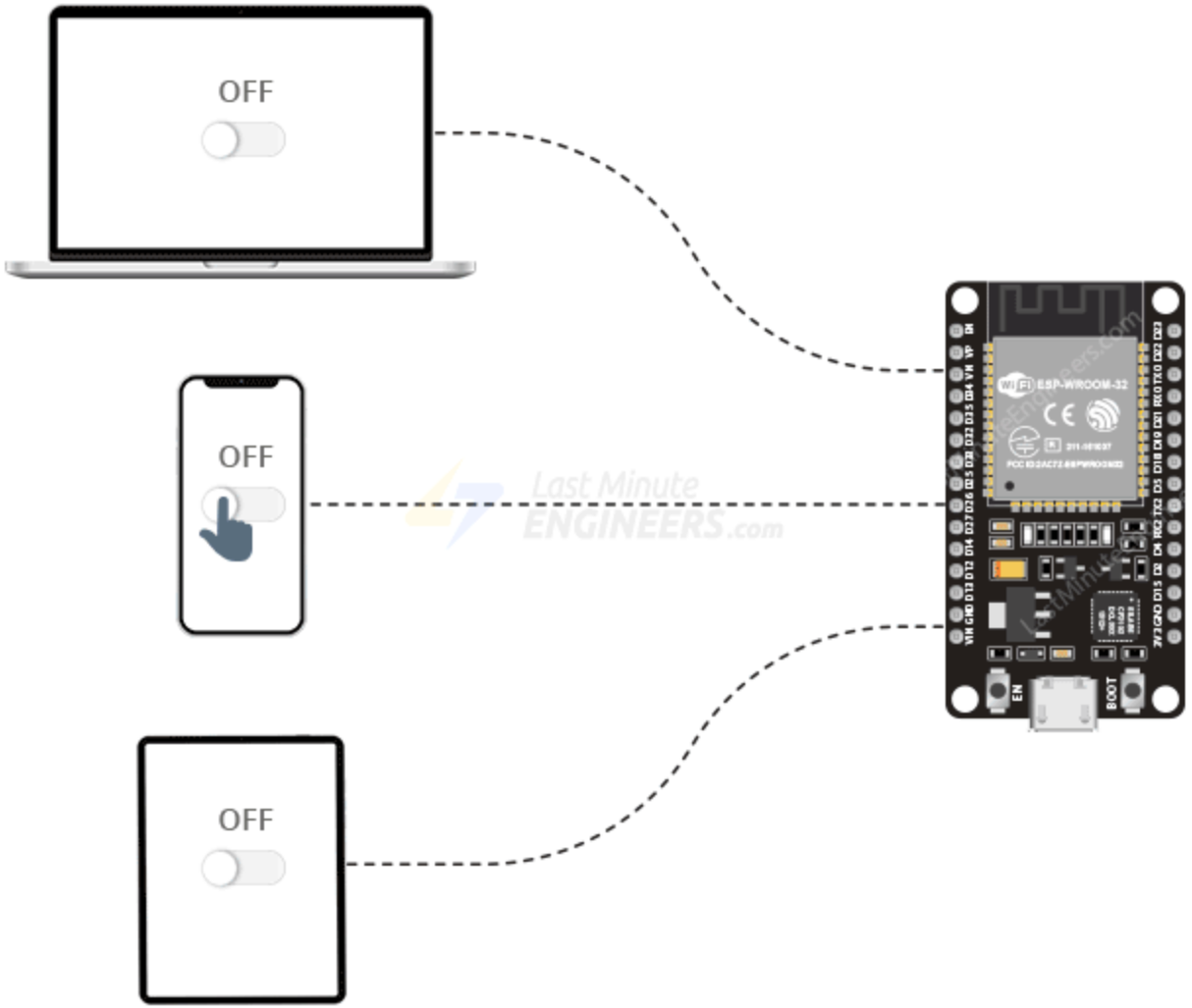
웹소켓은 클라이언트와 웹 서버 간 양방향(정확히는 전이중) 통신을 가능하게 하는 통신 프로토콜의 이름입니다. 간단히 말해, 웹소켓은 클라이언트와 서버 양측이 연결을 설정하여 언제든지 서로에게 메시지를 보낼 수 있게 하는 기술입니다.

이는 클라이언트가 요청을 시작하고 서버가 응답을 보낸 후 연결이 종료되는 일반적인 HTTP 연결과는 다릅니다. 사실 웹소켓은 완전히 다른 통신 프로토콜입니다. 클라이언트가 서버와 연결을 설정하면 연결 양 끝에서 데이터를 주고받을 수 있습니다. 따라서 서버가 새 메시지를 수신 대기하는 것처럼, 연결된 모든 클라이언트 역시 능동적으로 수신 대기합니다. 결과적으로 서버는 특정 클라이언트에게 데이터를 전송하거나 요청 없이도 모든 클라이언트에게 브로드캐스트할 수 있습니다. 게다가 클라이언트나 서버 중 한 쪽이 연결을 종료할 때까지 연결이 유지되므로 양측 간 지속적인 통신이 가능합니다.

ESP32 Web Server with WebSockets



ESP32 Web Server with WebSockets Code



라이브러리 설치

ESPAsyncWebServer

AsyncTCP

COM6

Send

```
Connecting to MiFi
.........
Connected..!
Got IP: 192.168.12.112
```

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

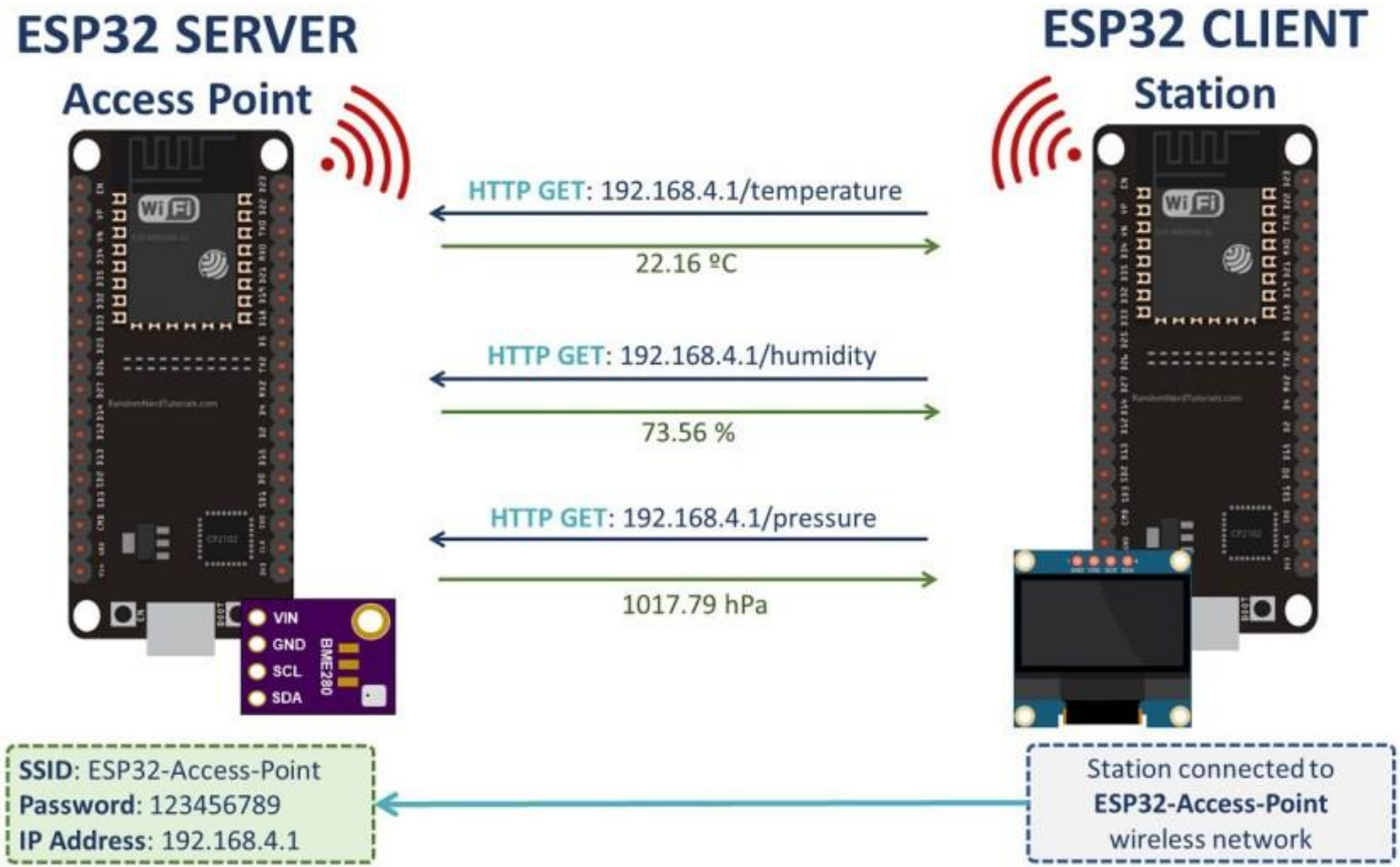
COM6

Send

```
Connecting to MiFi
.........
Connected..!
Got IP: 192.168.12.112
WebSocket client #1 connected from 192.168.12.221
WebSocket client #2 connected from 192.168.12.65
WebSocket client #1 disconnected
WebSocket client #2 disconnected
```

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

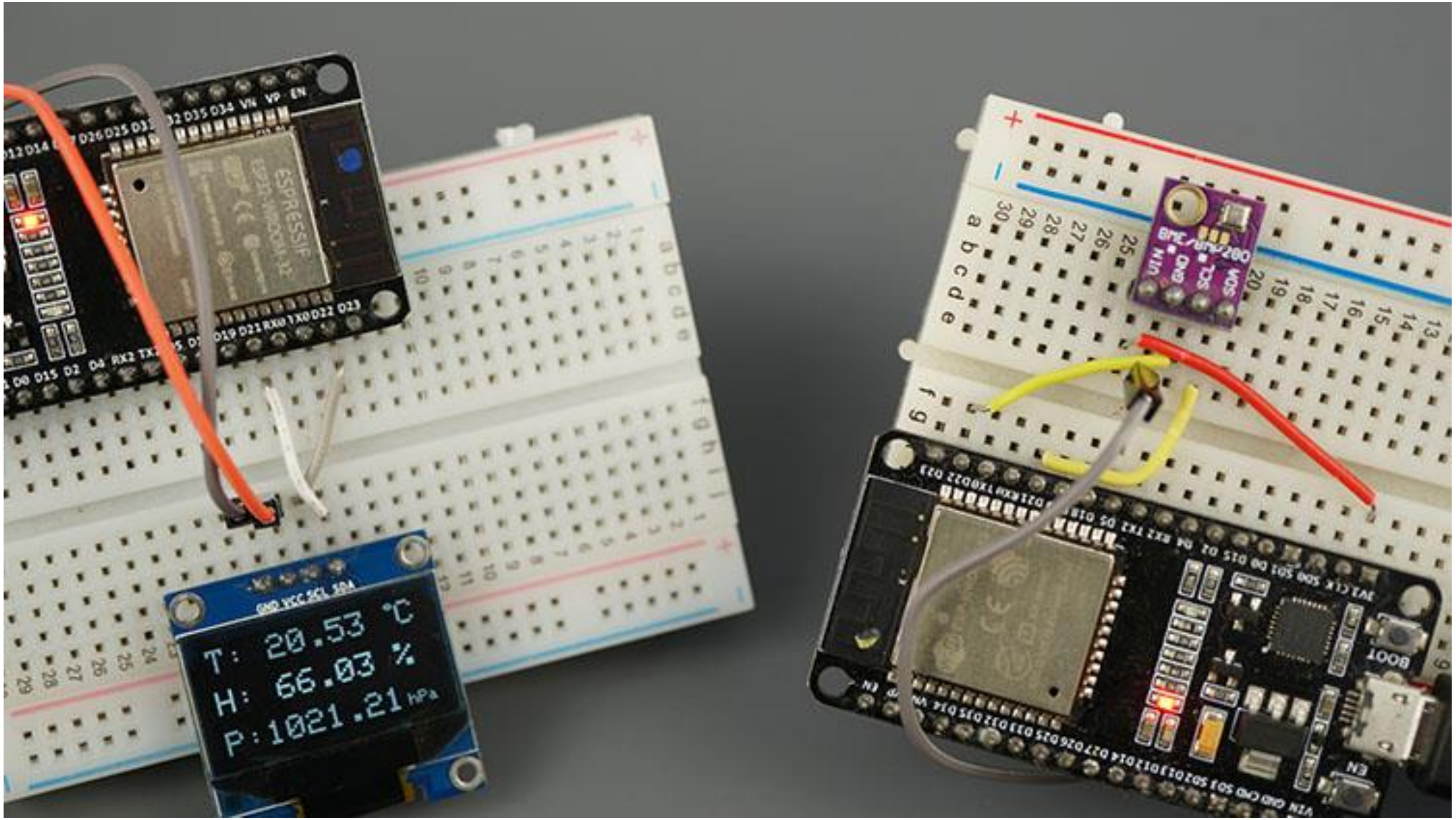
ESP32 두 대 서버 클라이언트 구조



https://fishpoint.tistory.com/9983

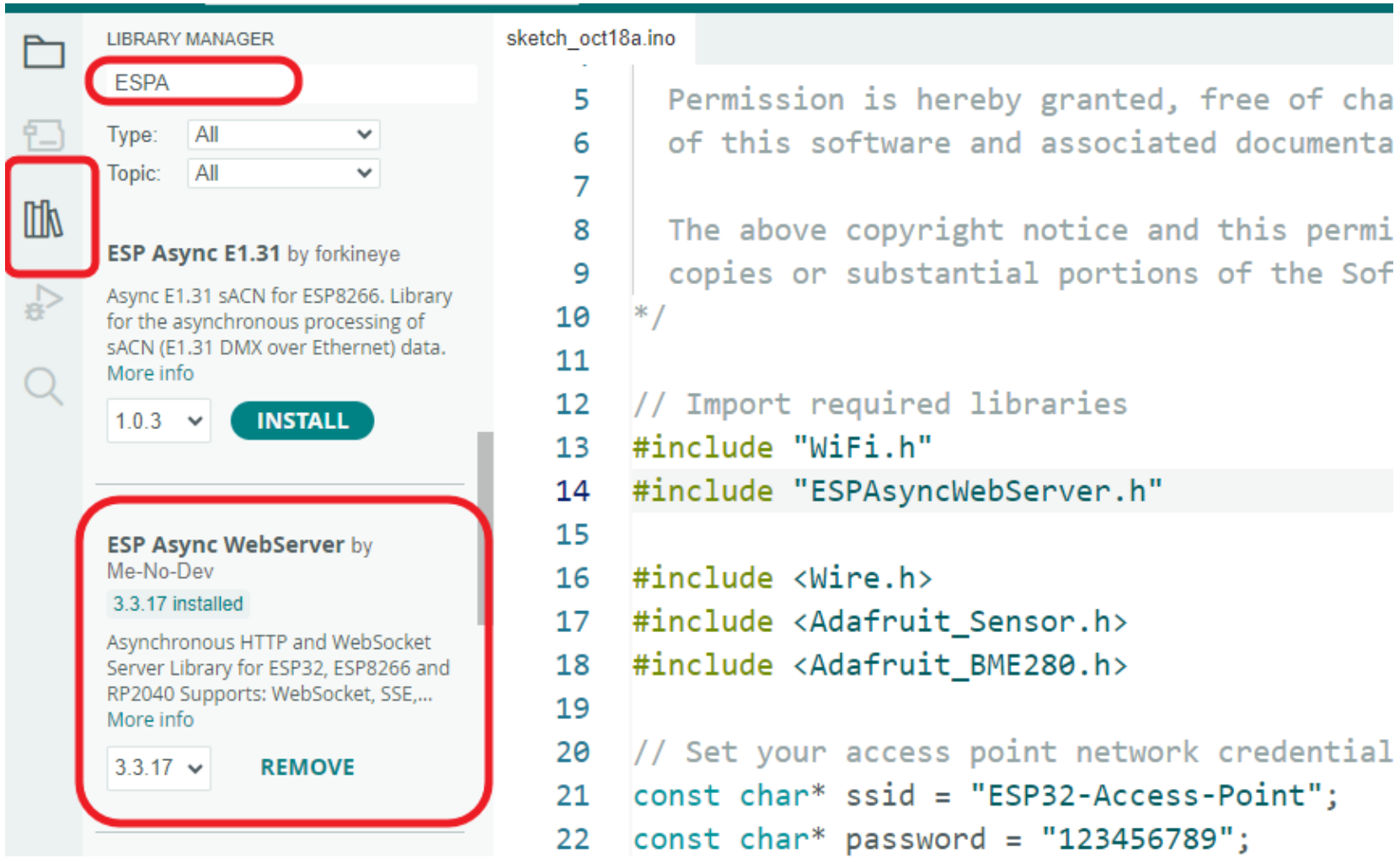
ESP32 Wi-Fi 두 대 서버 클라이언트

프로젝트 참고 https://fishpoint.tistory.com/9983



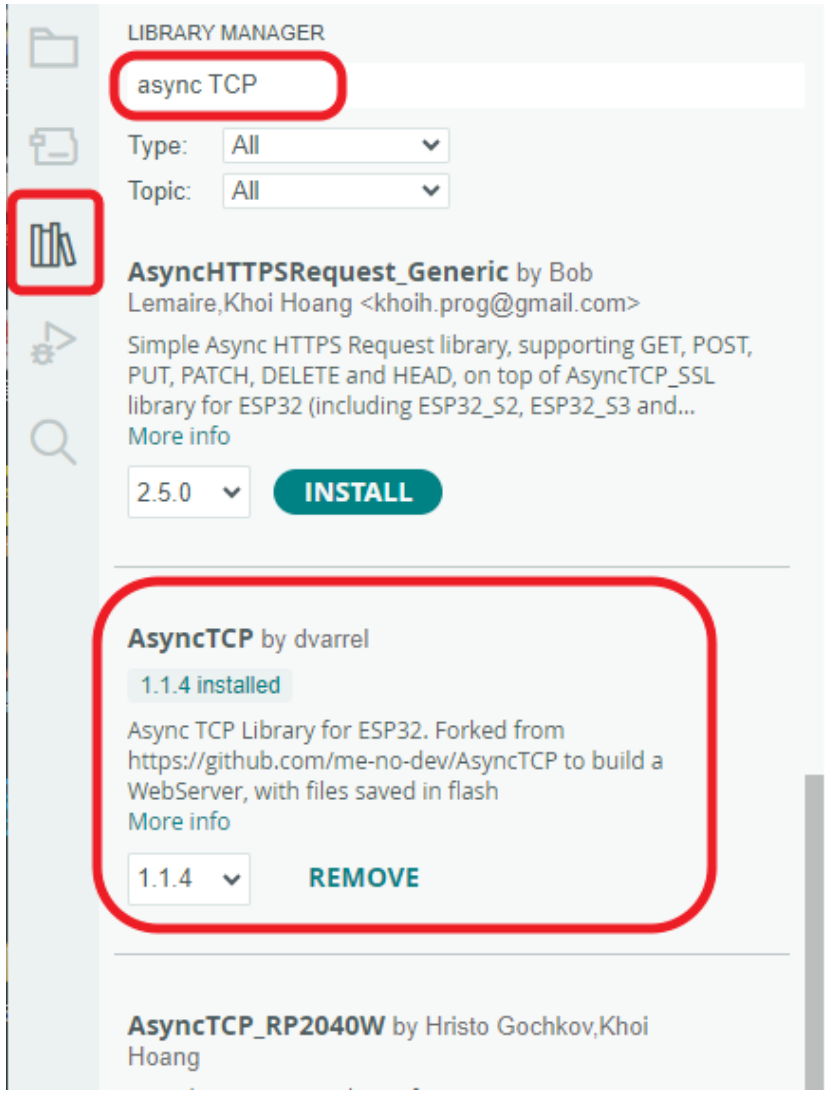
ESP32 Wi-Fi 두 대 서버 클라이언트

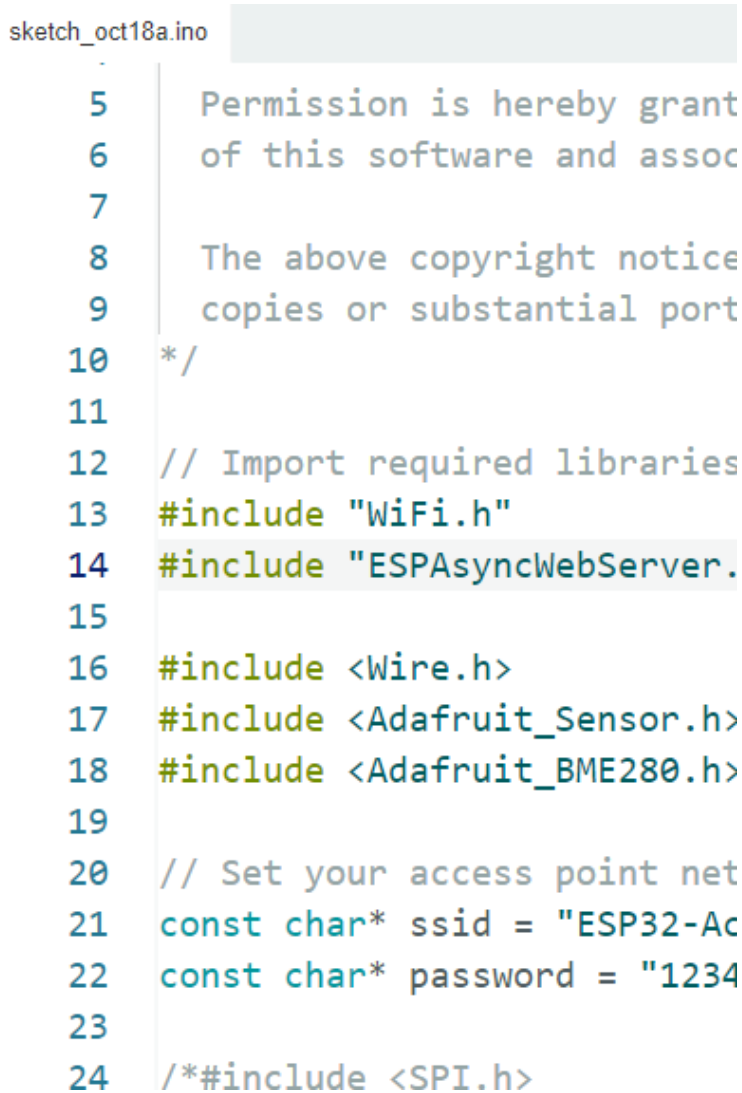
비동기 웹 서버 라이브러리 설치



ESP32 Wi-Fi 두 대 서버 클라이언트

비동기 TCP 라이브러리



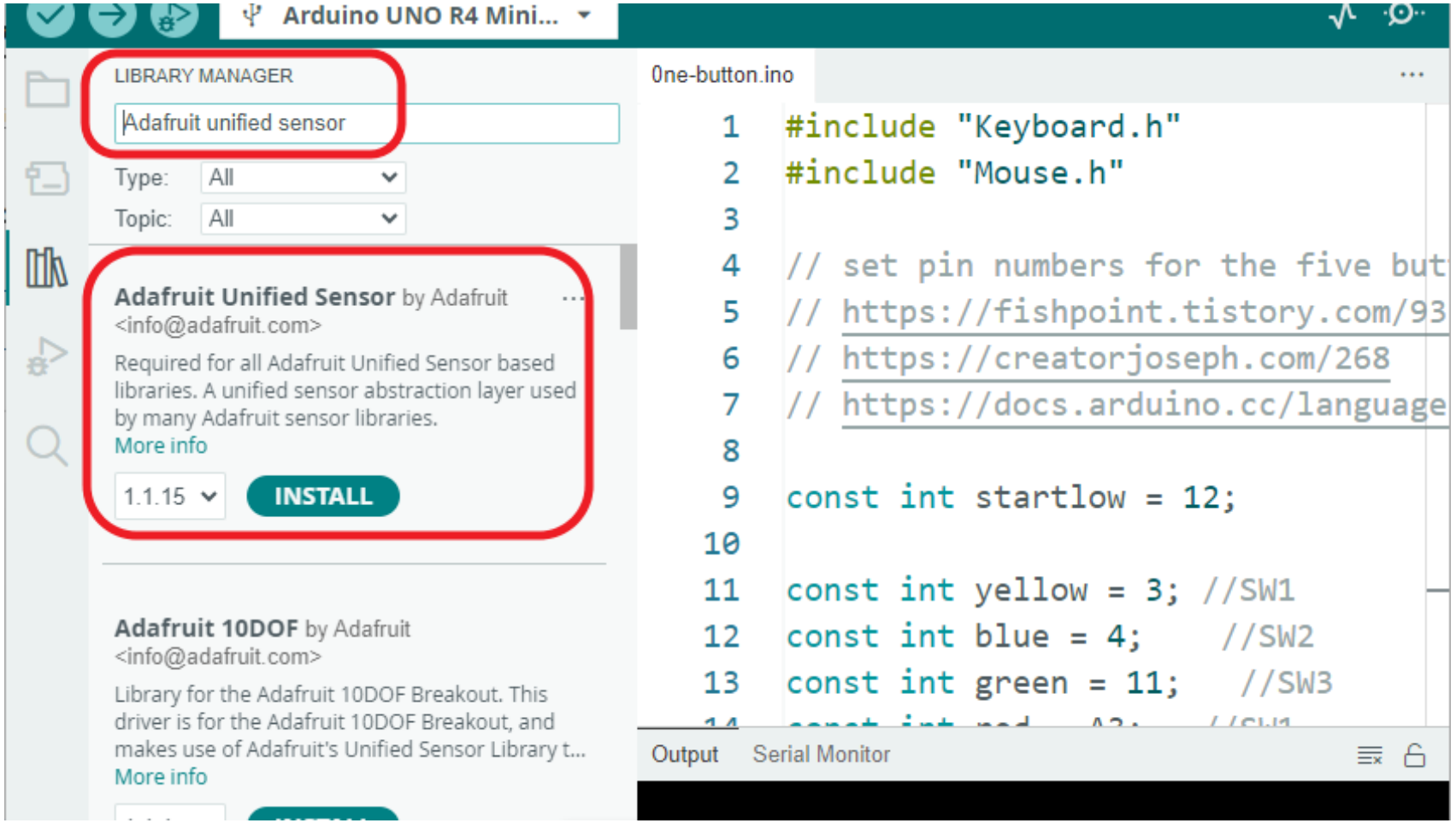


ESP32 Wi-Fi 두 대 서버 클라이언트

- Adafruit_BME280 라이브러리
- Adafruit unified sensor 라이브러리

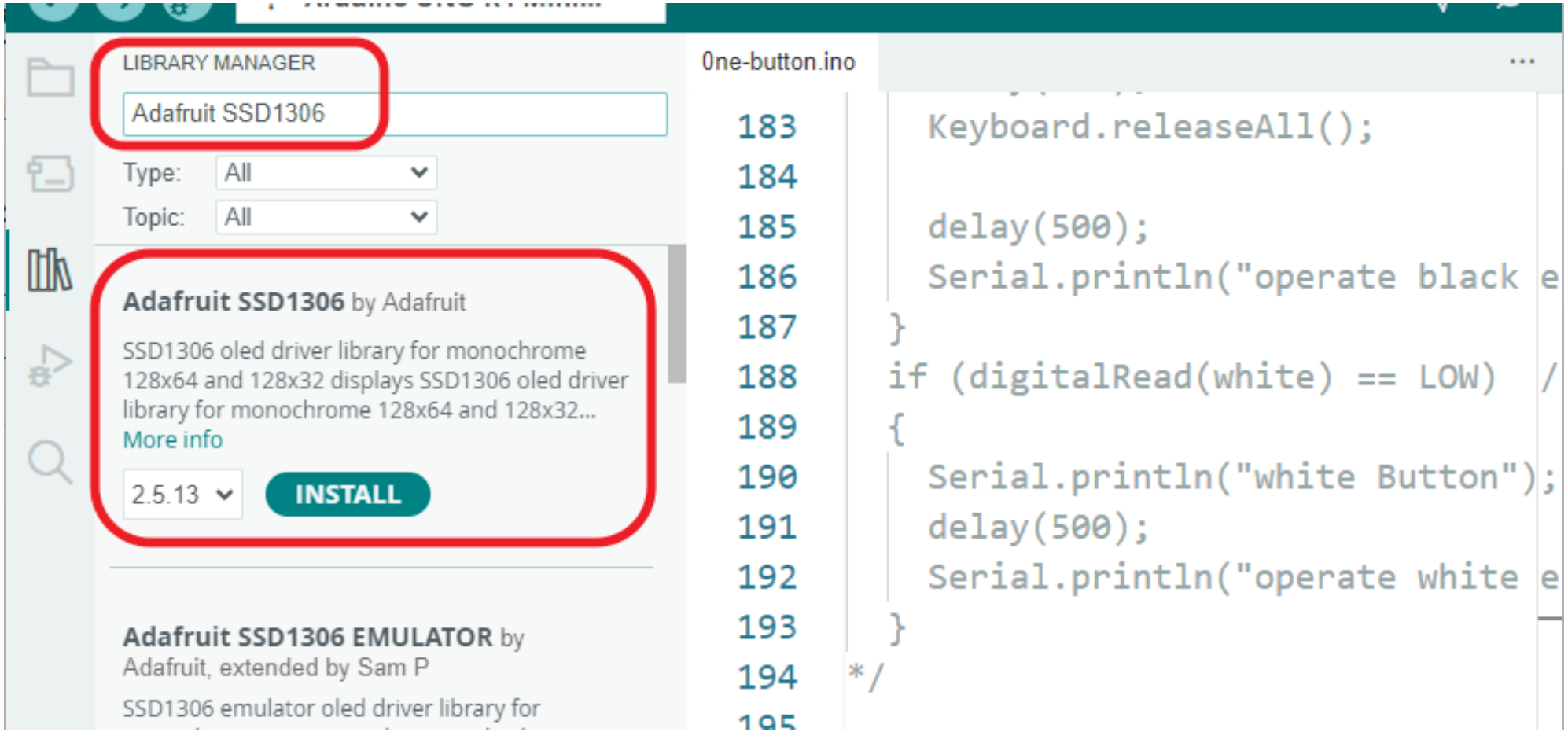
ESP32 Wi-Fi 두 대 서버 클라이언트

- Adafruit_BME280 라이브러리
- Adafruit unified sensor 라이브러리



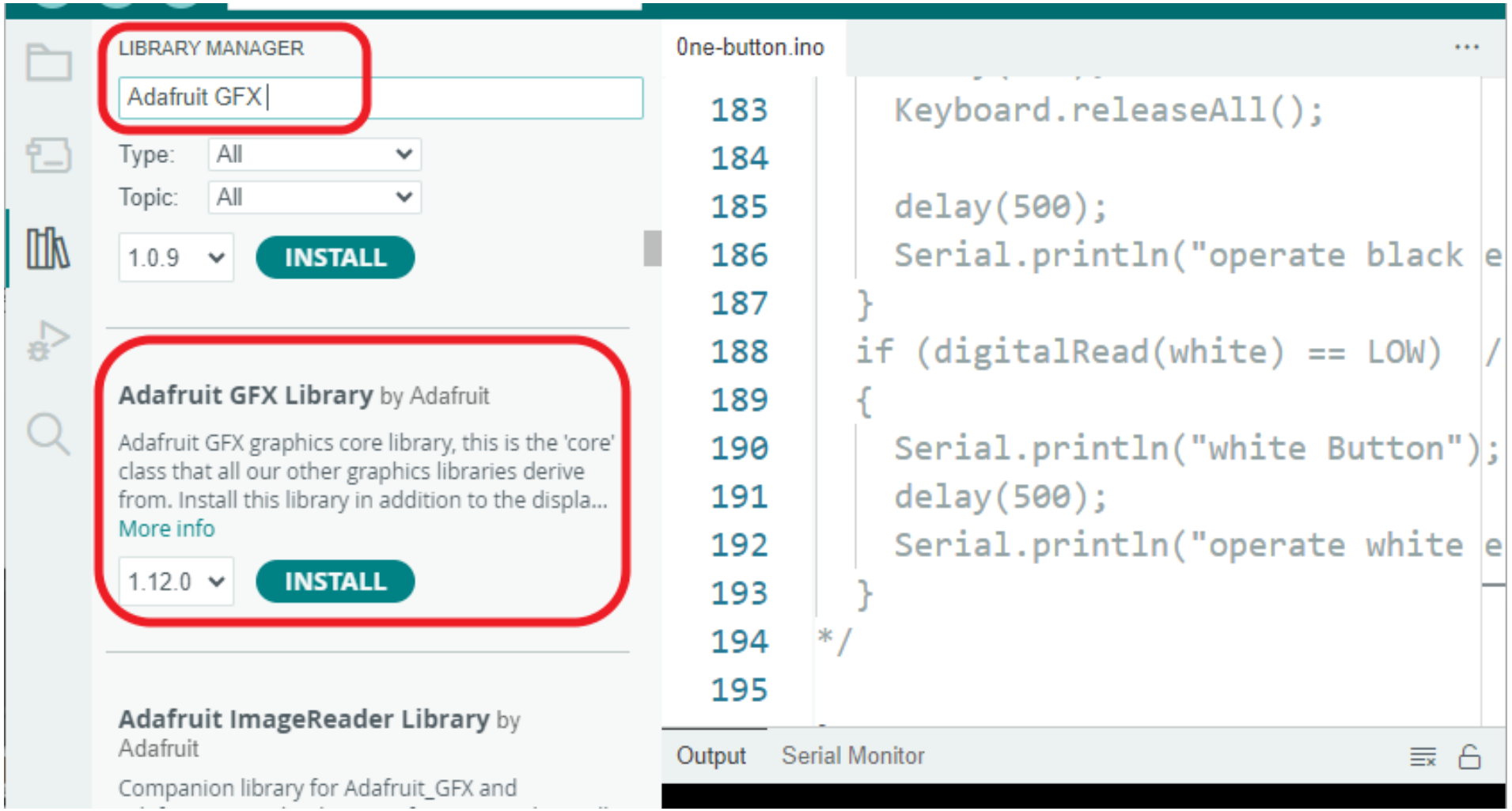
ESP32 Wi-Fi 두 대 서버 클라이언트

- Adafruit SSD1306
- Adafruit GFX 라이브러리



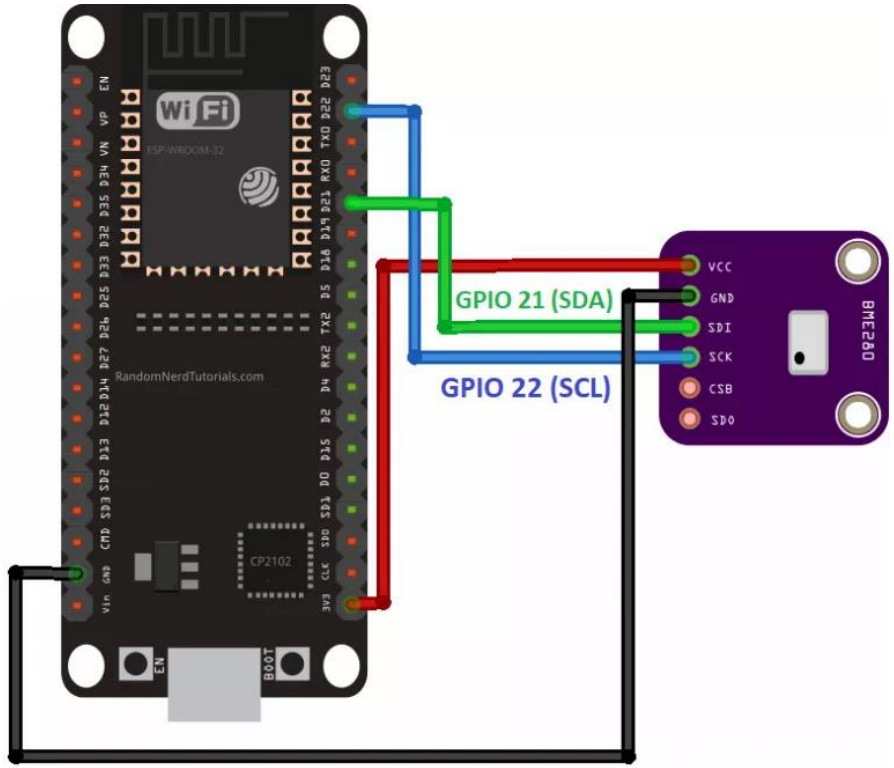
ESP32 Wi-Fi 두 대 서버 클라이언트

- Adafruit SSD1306
- Adafruit GFX 라이브러리

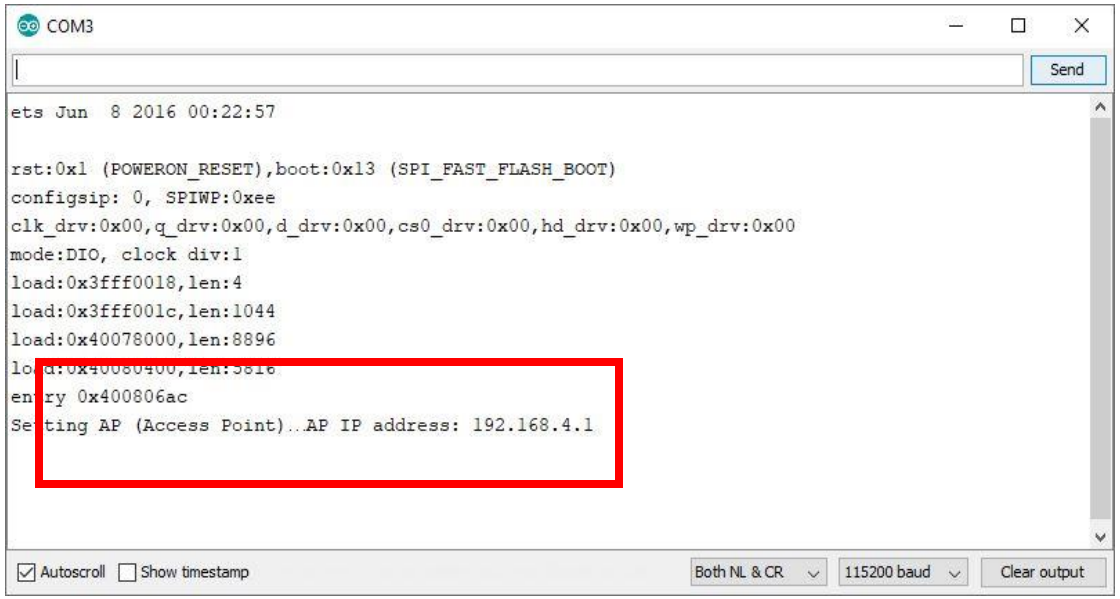


ESP32 Wi-Fi 두 대 서버 클라이언트

#1 ESP32 서버(액세스 포인트)

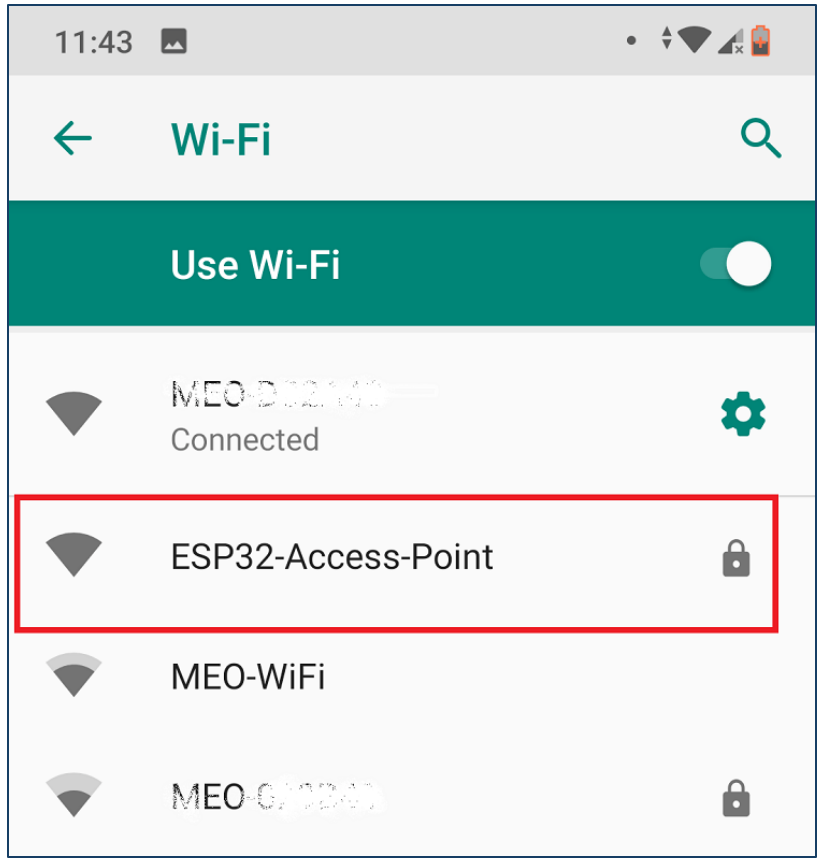


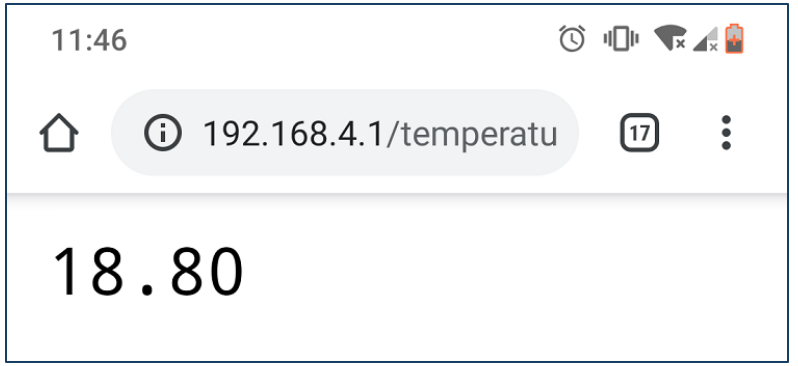
BME280	ESP32
VIN/VCC	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21

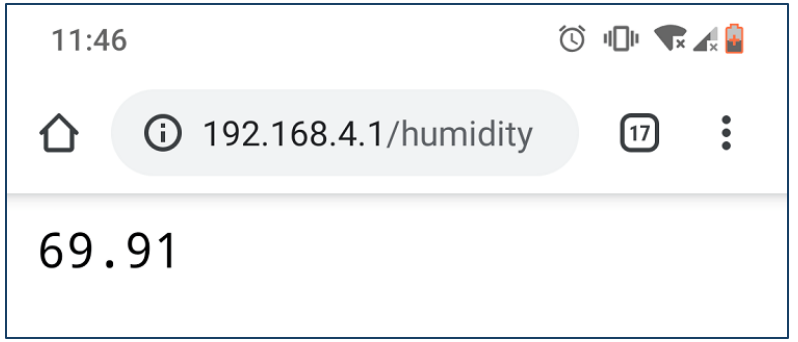


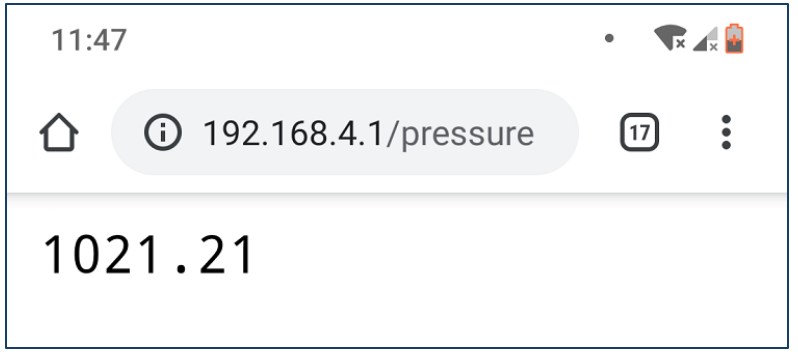
ESP32 Wi-Fi 두 대 서버 클라이언트

스마트폰에서 Wi-Fi 설정으로 이동하여 ESP32-Access-Point에 연결합니다. 비밀번호는 123456789입니다.



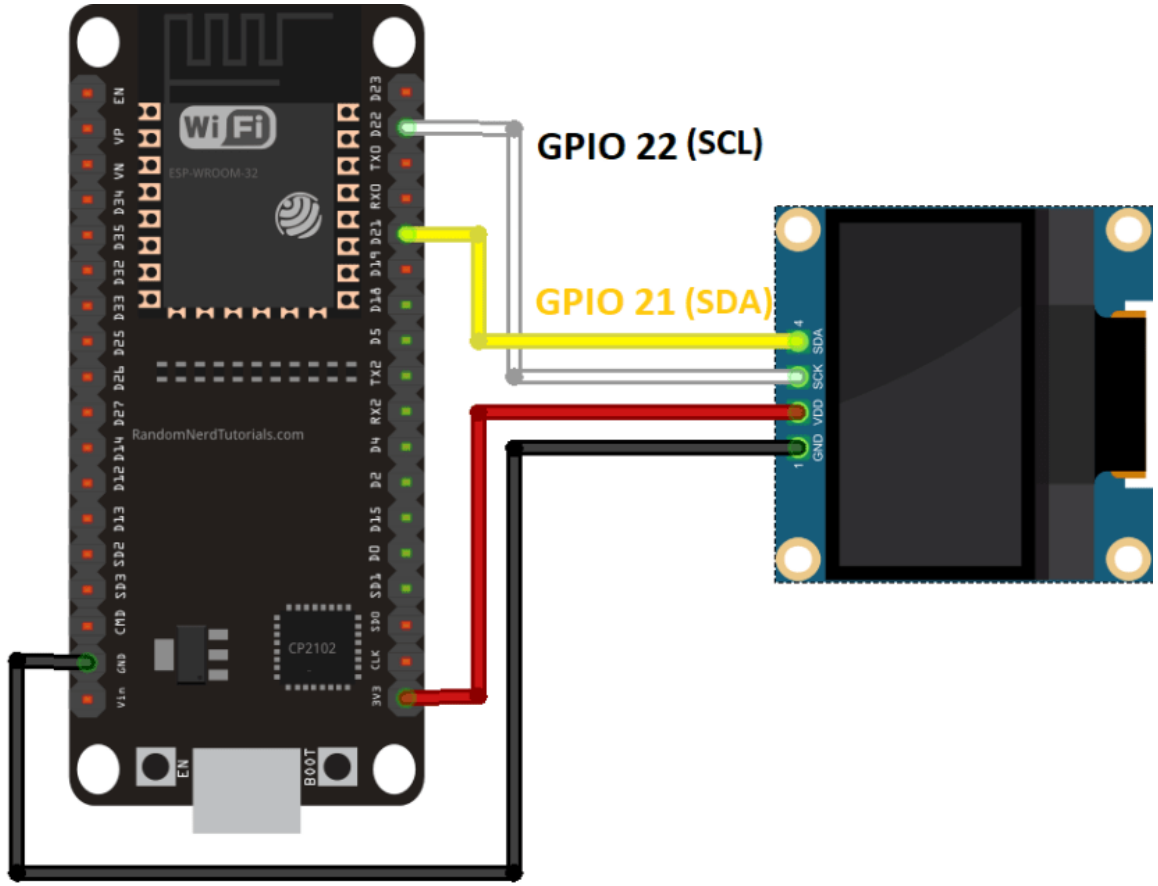




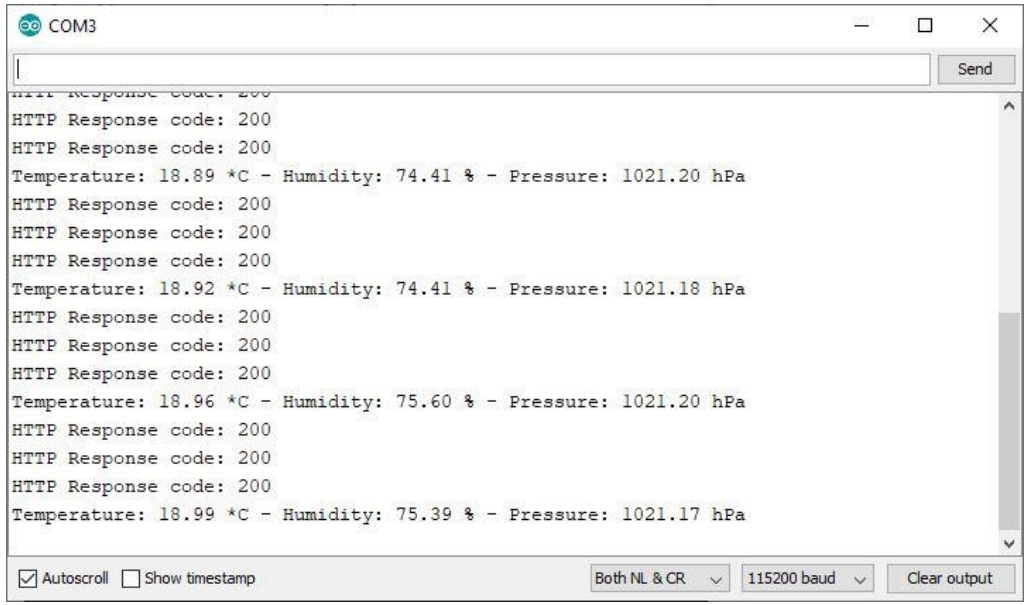


ESP32 Wi-Fi 두 대 서버 클라이언트

#2 ESP32 클라이언트(스테이션)



OLED	ESP32
VIN/VCC	VIN
GND	GND
SCL	GPIO 22
SDA	GPIO 21

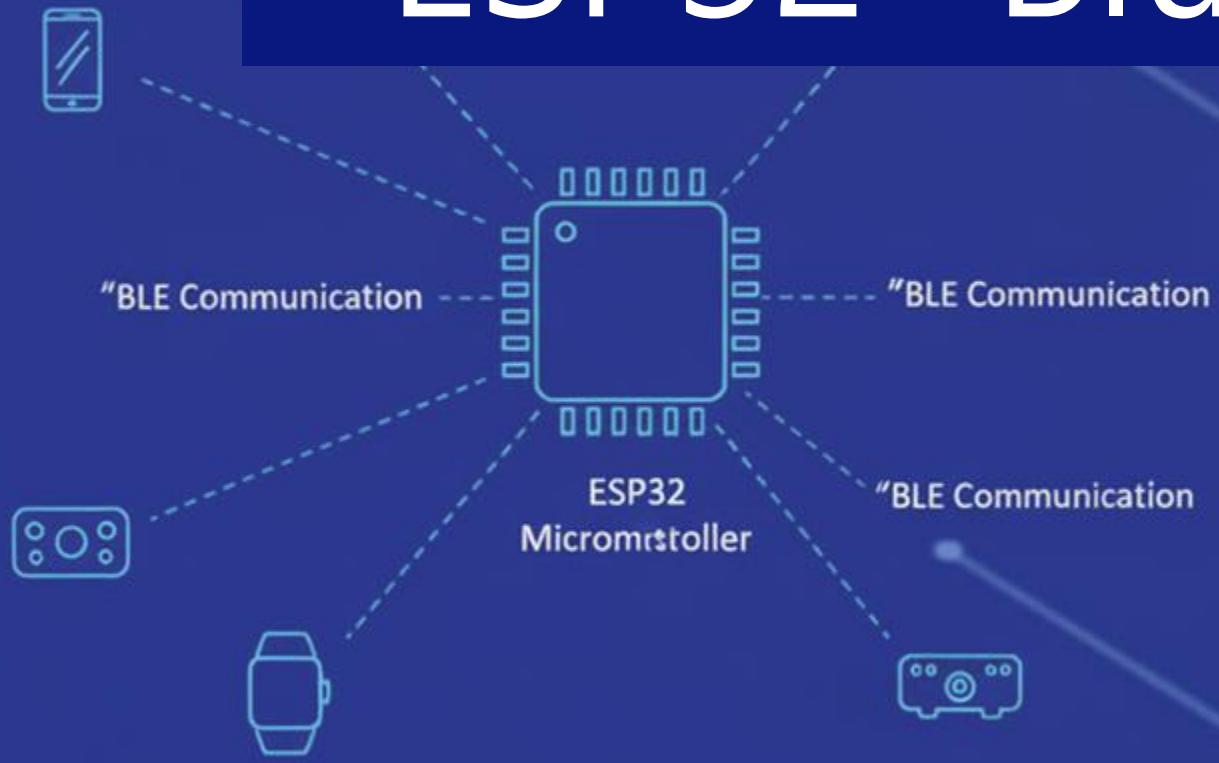


ESP32 DHT11, DHT22 웹 서버 만들기

참고 문서 https://fishpoint.tistory.com/9998

- DHT 센서에서 온도 및 습도를 읽어옵니다.
- ESPAsyncWebServer 라이브러리를 사용하여 비동기 웹 서버를 빌드합니다.
- 웹 페이지를 새로 고칠 필요 없이 자동으로 센서 판독값을 업데이트합니다.

ESP32 Bluetooth

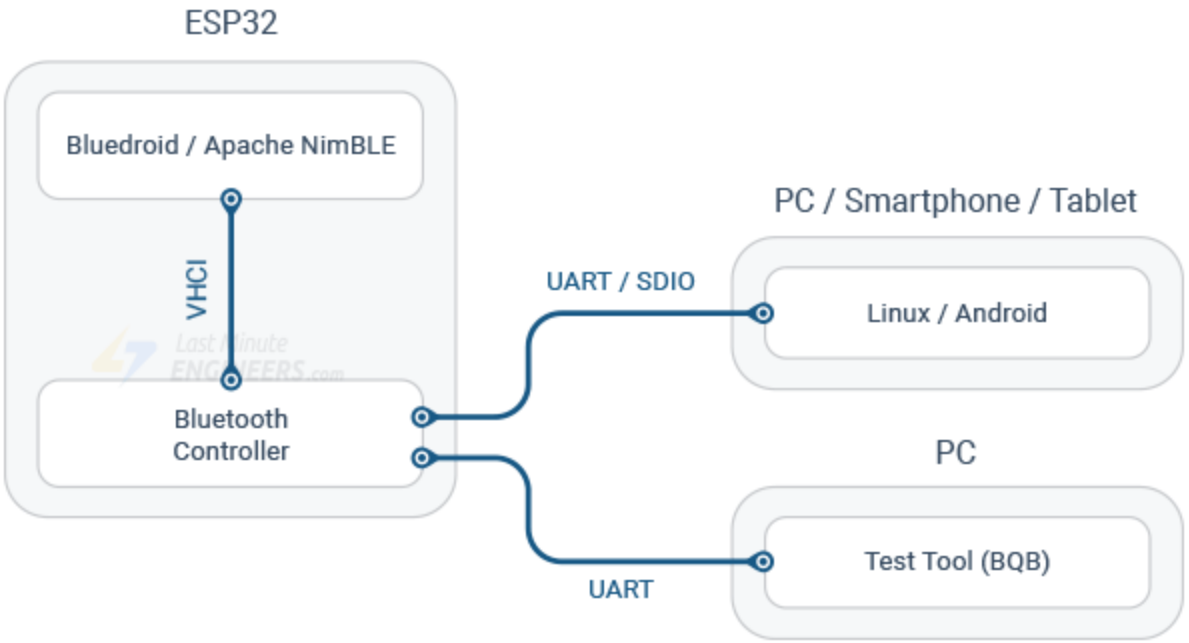


Care-lab.kr

Bluetooth Classic vs. BLE

	Bluetooth Classic	Bluetooth Low Energy (BLE)
Data Rate	1 Mbps for BR2-3 Mbps for EDR	500 kbps – 1 Mbps
Frequency Band	2.4 GHz ISM band	2.4 GHz ISM band
Number of Channels	79 Channels with 1 MHz spacing	40 Channels with 2 MHz spacing
Communication Range	8 m up to 100 m	8 m up to 100 m
Power Consumption	High (up to 1 W)	Low (0.01 W up to 0.5 W)
Device Pairing is Mandatory?	YES	NOT Mandatory
Supported Topologies	Point-to-Point (1:1)	Point-to-Point (1:1)Broadcast (1:many)Mesh (many:many)
Modulation Technique	GFSKπ/4 DQPSK8 DPSK	GFSK
Latency	35ms	2-16ms (Avg. 9ms)

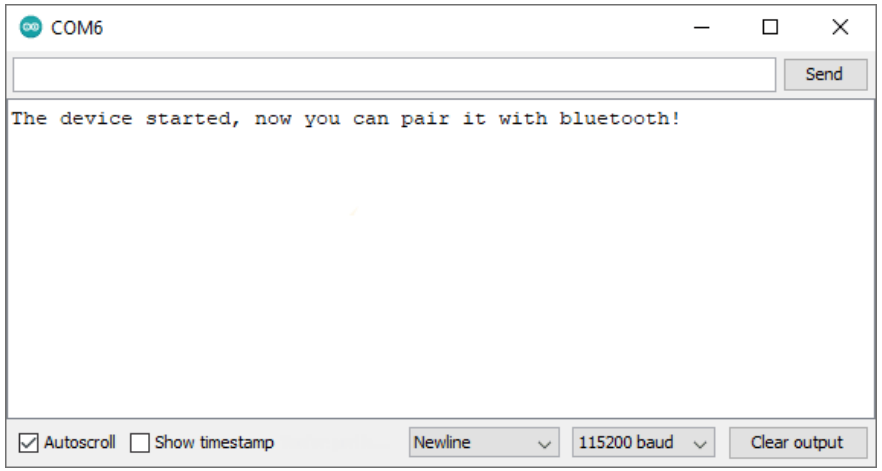
ESP32 Bluetooth Protocol Stack



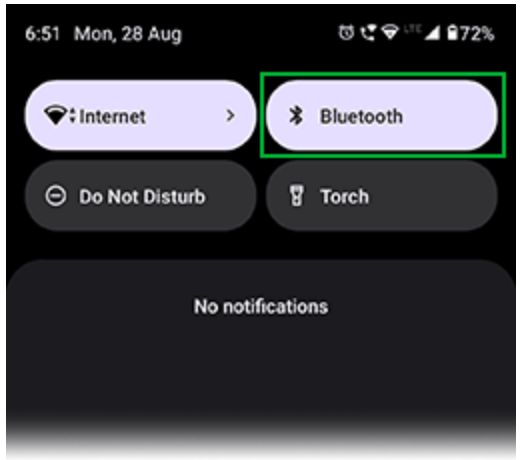
Espressif's Bluetooth architecture description.

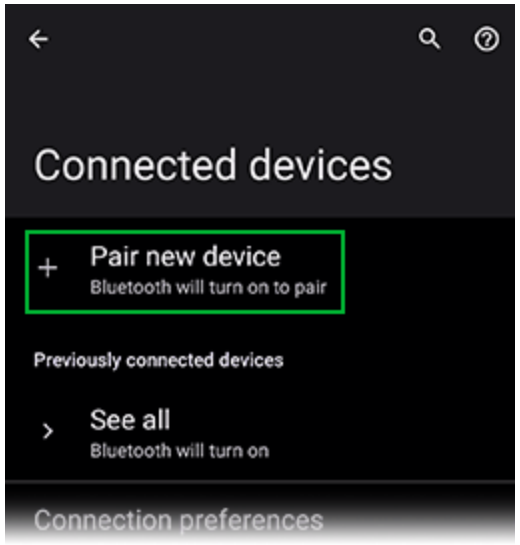
Code

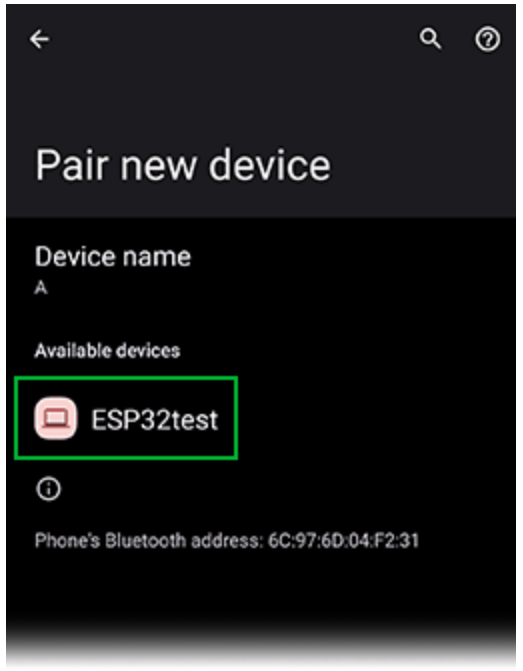
File > Examples > BluetoothSerial > SerialtoSerialBT

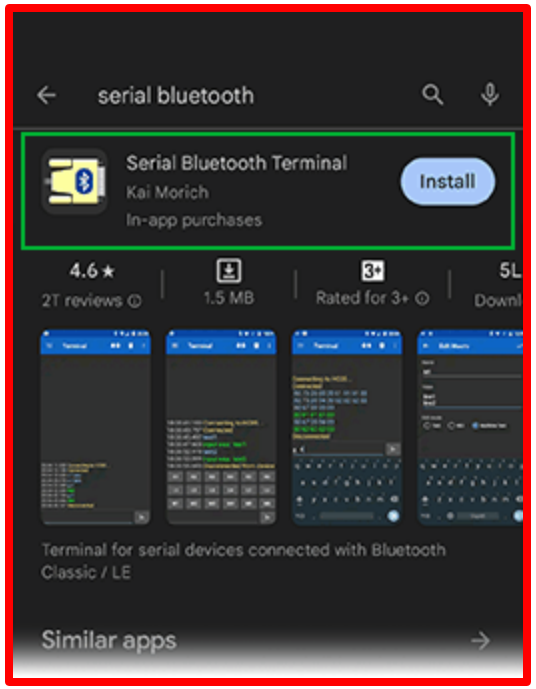


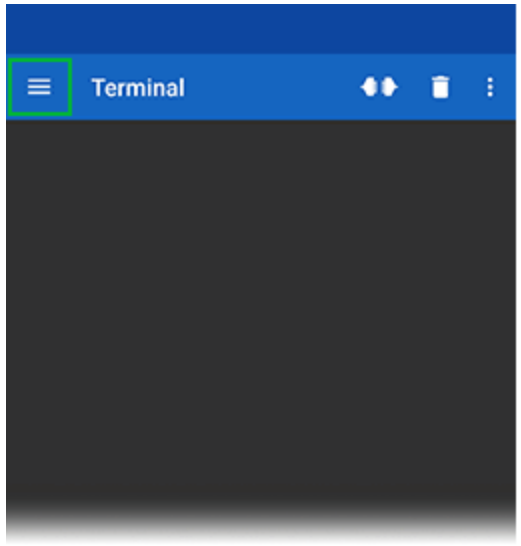
Connecting to the Android Phone

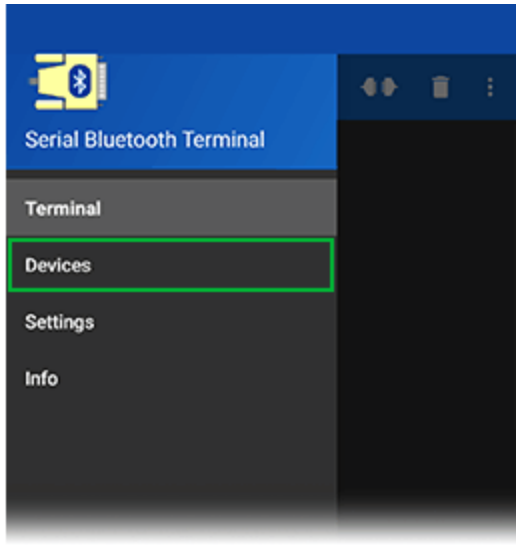


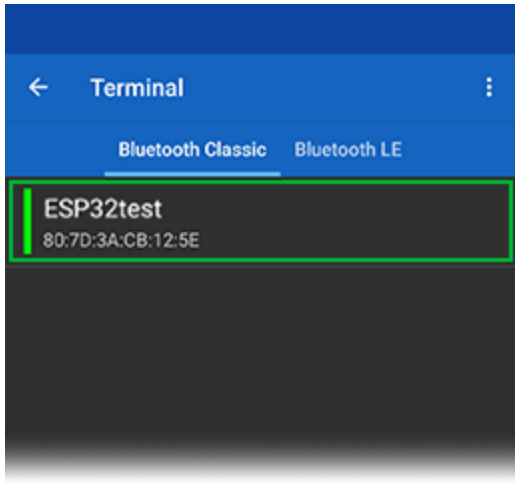




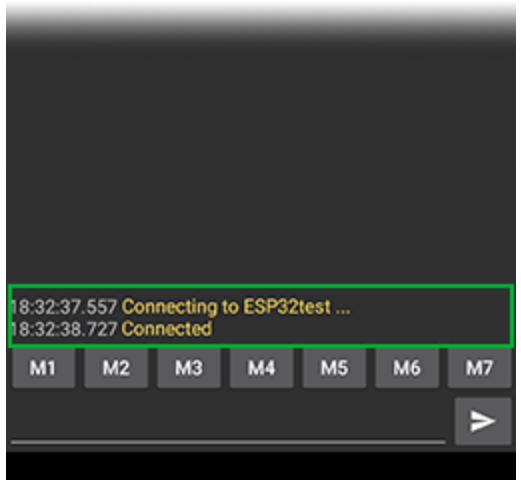


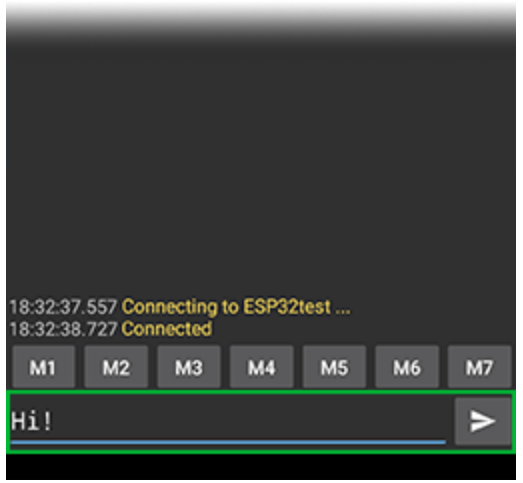


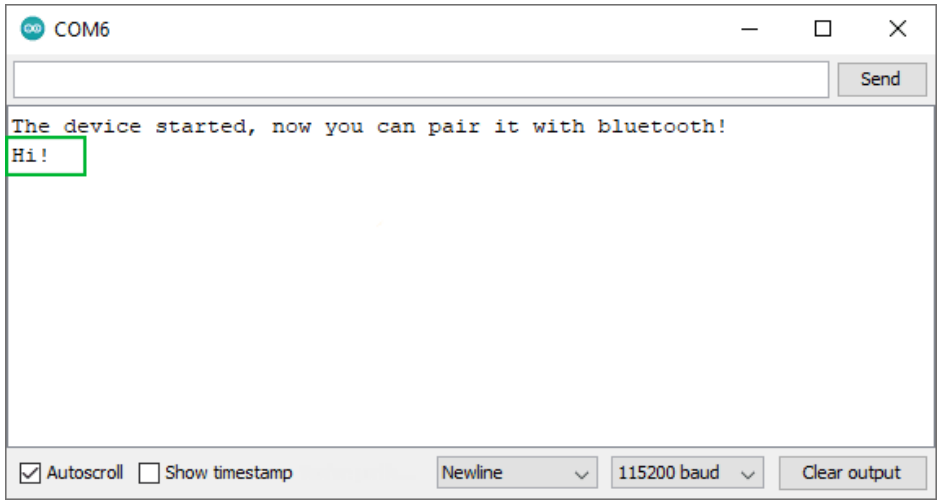


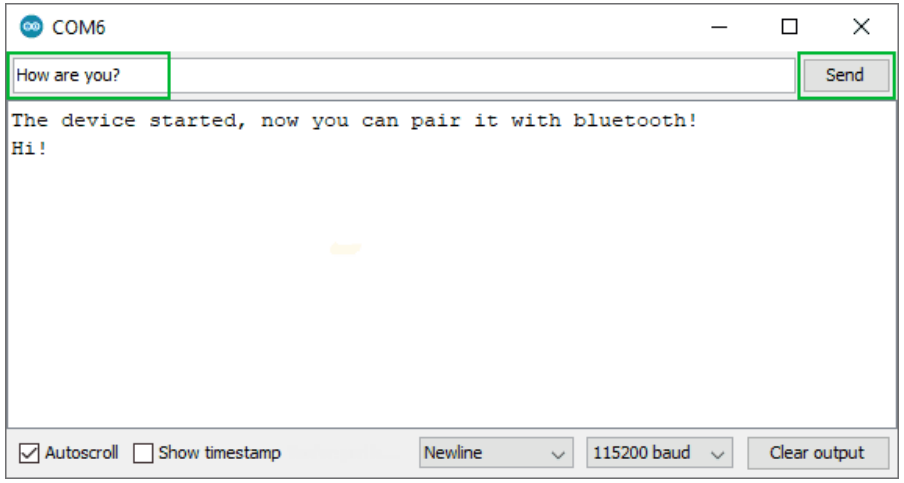


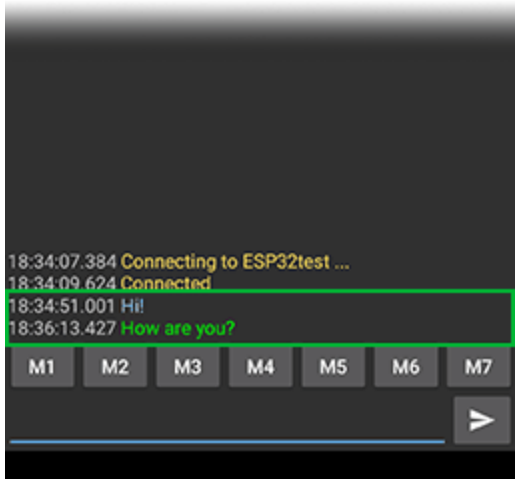
Connecting to the Android Phone



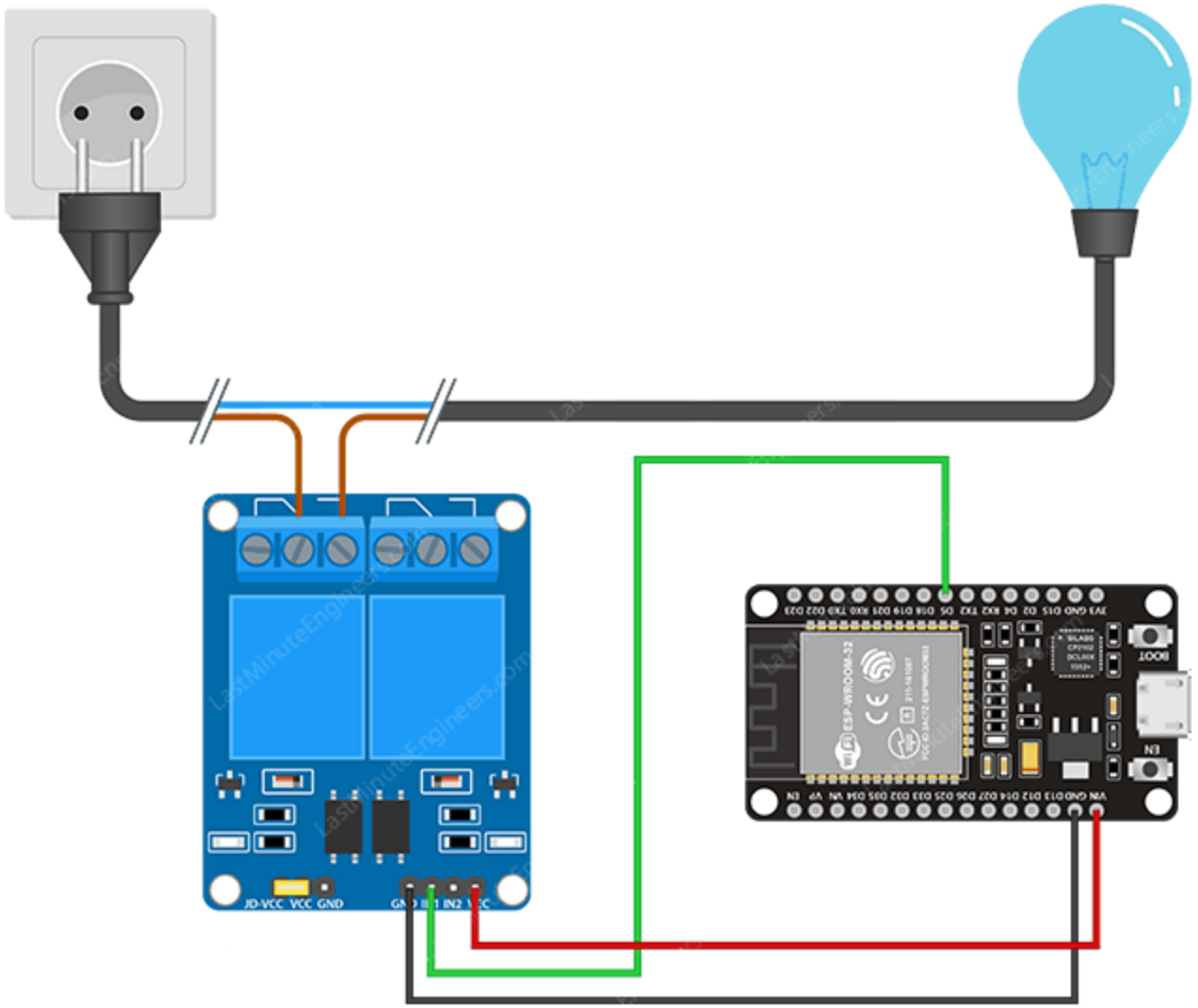




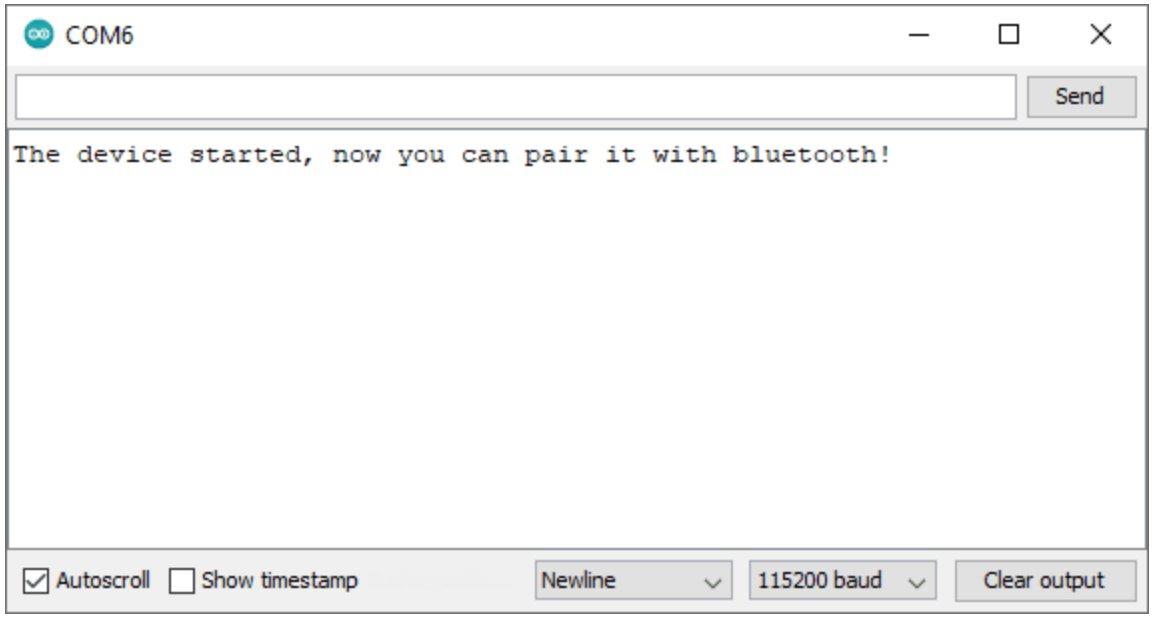


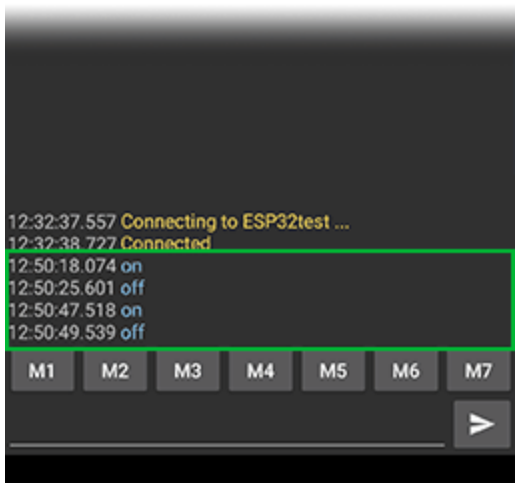


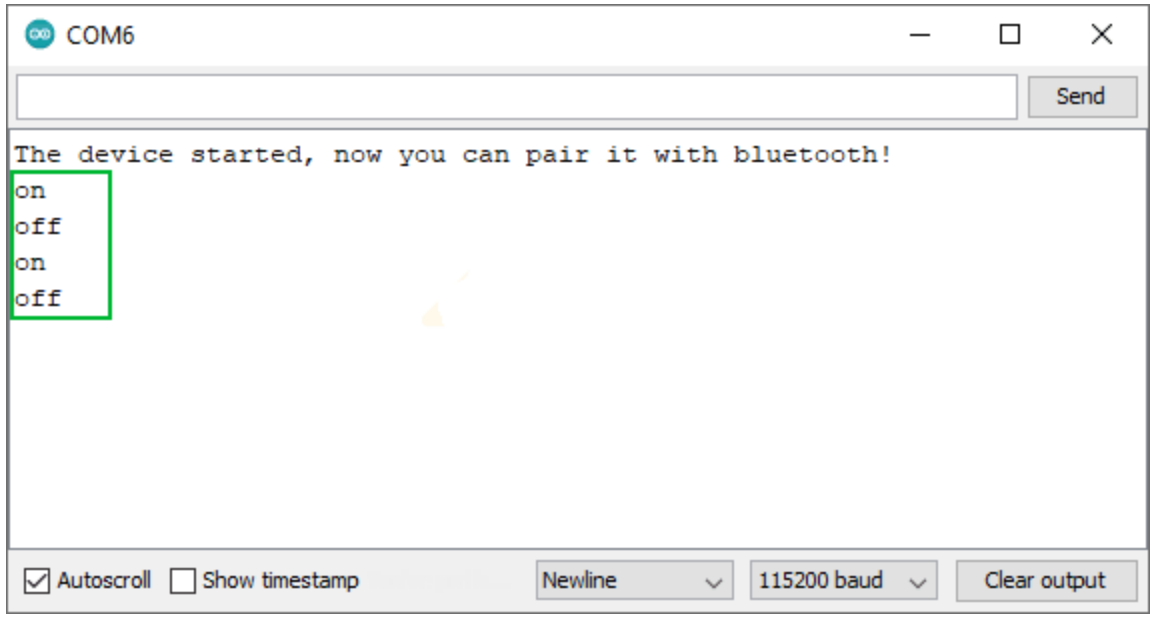
Bluetooth-Controlled Relay



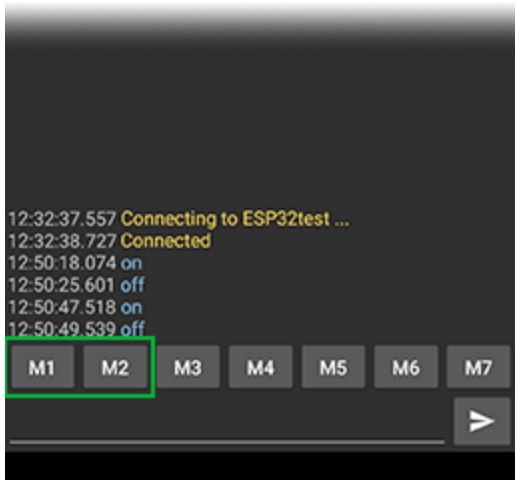
Bluetooth-Controlled Relay

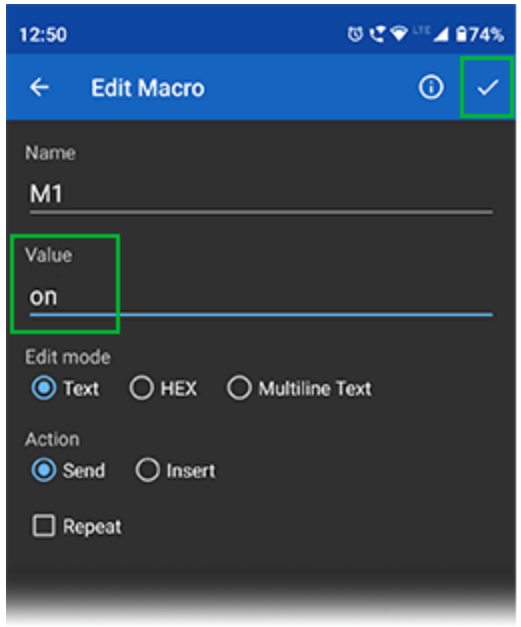


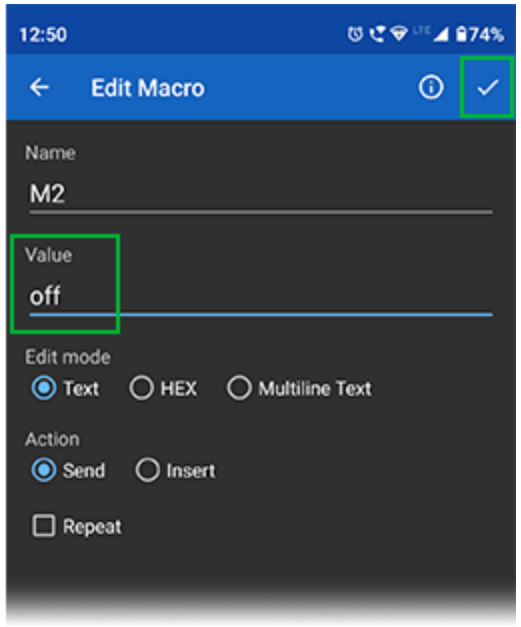




Bluetooth-Controlled Relay







Bluetooth Profiles

블루투스 프로파일은 기본 블루투스 표준을 기반으로 구축된 추가 프로토콜입니다. 이는 블루투스 모듈이 전송하는 데이터 유형을 지정하는 데 도움이 됩니다. 블루투스 사양이 기술의 작동 방식을 정의하는 반면, 프로파일은 사용 방식을 정의합니다.

블루투스 기기가 지원하는 프로파일은 해당 기기가 설계된 용도를 결정합니다. 예를 들어, 핸즈프리 블루투스 헤드셋은 헤드셋 프로파일(HSP)을 사용하는 반면, 무선 키보드는 휴먼 인터페이스 디바이스(HID) 프로파일을 사용합니다. 이러한 프로파일은 블루투스 SIG(Bluetooth Special Interest Group) 또는 주변기기 설계자에 의해 개발됩니다. 공식적으로 채택된 프로파일의 전체 목록은 여기에서 확인할 수 있습니다.

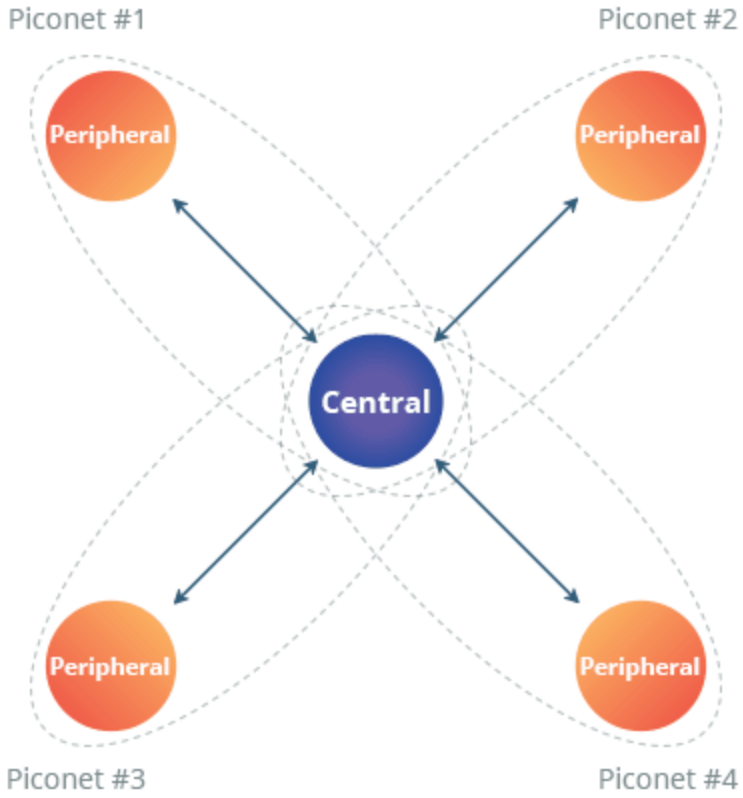
https://www.bluetooth.com/specifications/specs/

모든 표준 BLE 프로파일이 이들을 기반으로 하므로, 여러분이 반드시 숙지해야 할 프로파일은 GAP과 GATT입니다. 이제 이들을 더 자세히 알아보겠습니다.

GAP (Generic Access Profile)

Peripheral devices일반적으로 심박수 모니터나 근접 태그처럼 소형이고 저전력이며 자원이 제한된 장치로, 훨씬 더 강력한 중앙 기기에 연결될 수 있습니다.

Central devices일반적으로 휴대폰이나 태블릿처럼 훨씬 더 많은 처리 능력과 메모리를 가진 장치로, 사용자가 연결하는 대상입니다.

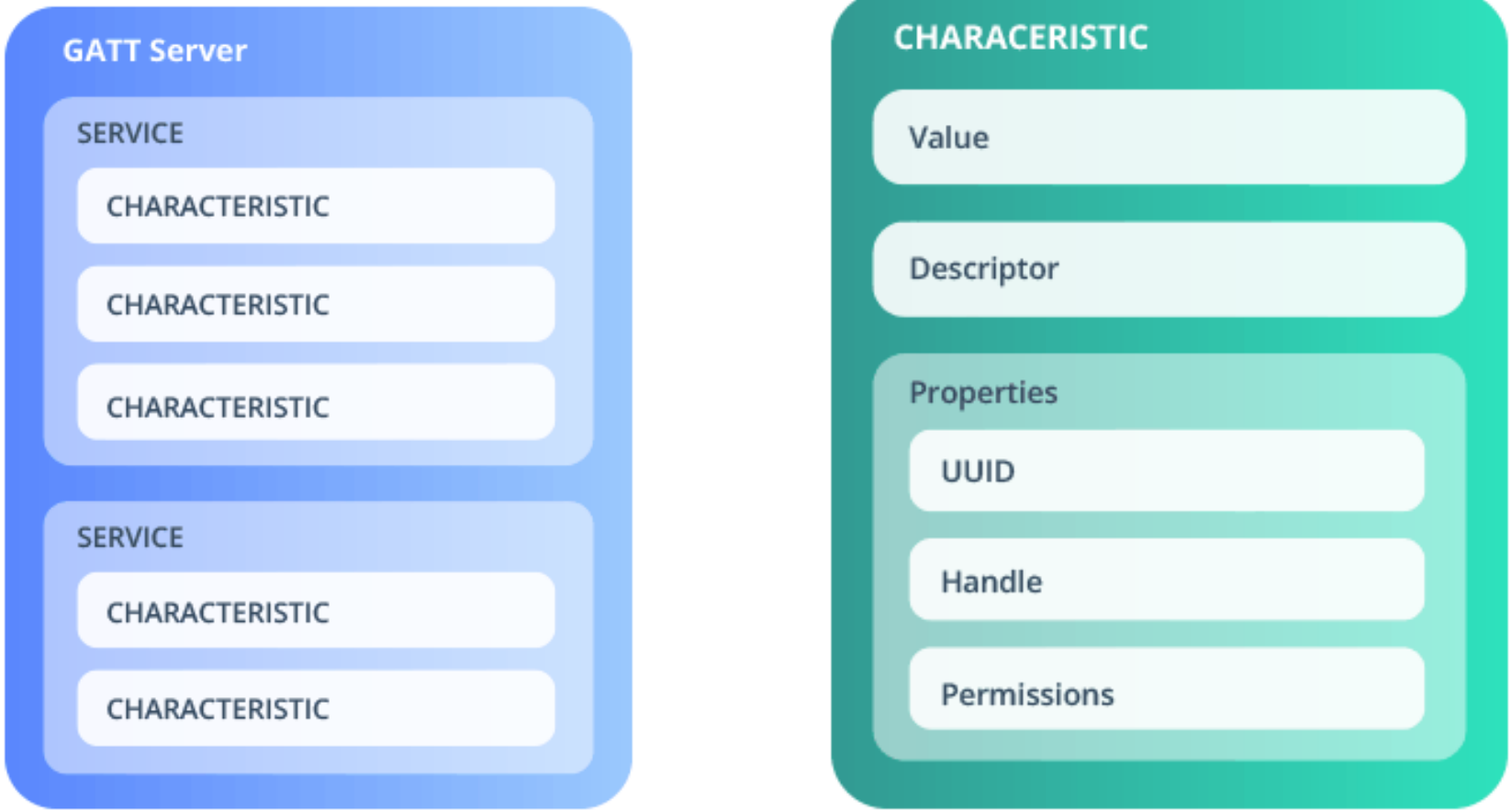


주변 장치는 설정된 간격으로 광고 패킷을 전송하여 주변의 중앙 장치에 자신의 존재를 알립니다. 주변 장치와 중앙 장치 간 연결이 수립되면 광고 프로세스가 중단되고 GATT가 작동하여 양방향 통신이 가능해집니다.

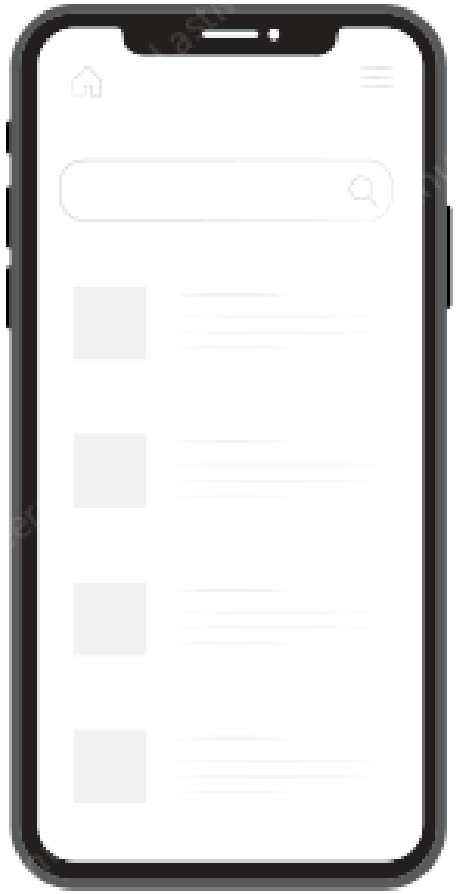
GATT (Generic ATTribute Profile)

GATT는 데이터의 구성 방식과 두 BLE 장치 간 데이터 공유 방법을 규정합니다. 장치와의 저수준 상호작용을 정의하는 GAP과 달리, GATT는 실제 데이터 전송 절차와 형식만을 다룹니다.

데이터는 아래 그림과 같이 계층적으로 '서비스'라는 섹션으로 구성되며, 이는 개념적으로 관련된 사용자 데이터 조각인 '특성'을 그룹화합니다:



GATT Server and GATT Client

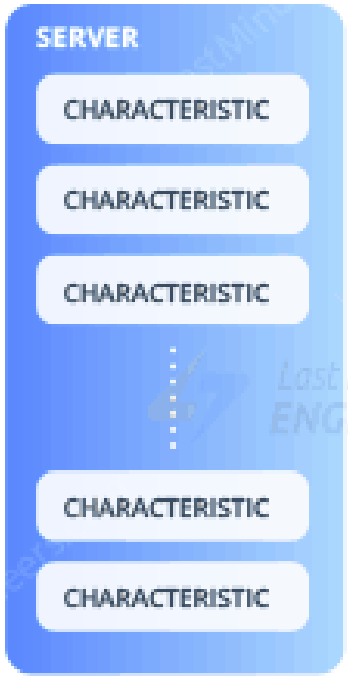


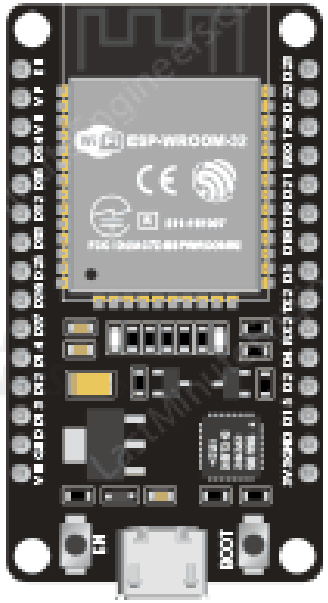
GATT Client

Smartphone / PC

Requests / Commands

Responses





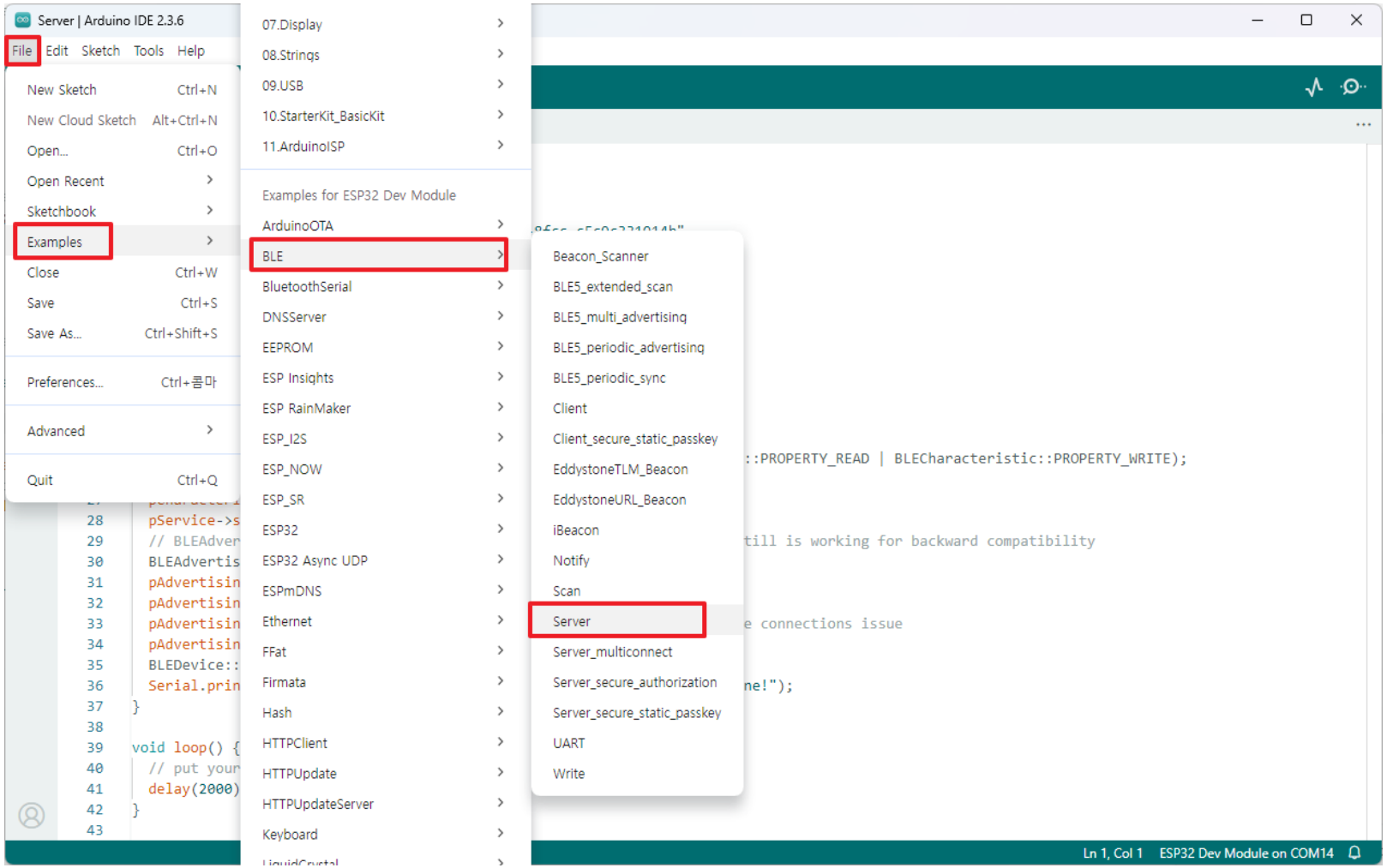
GATT Server

ESP32

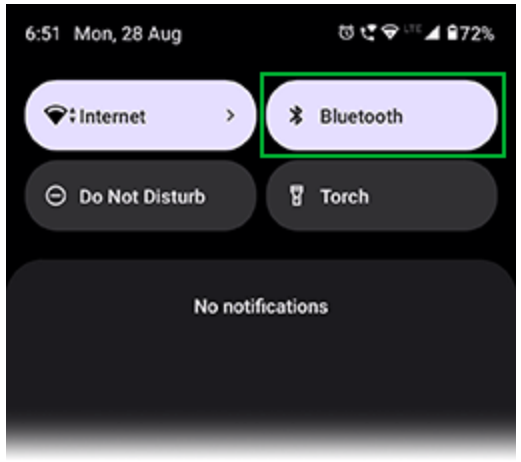
BLE on the ESP32

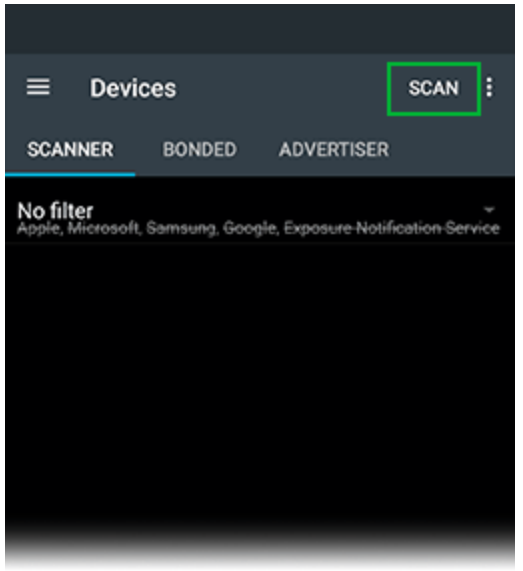
Code

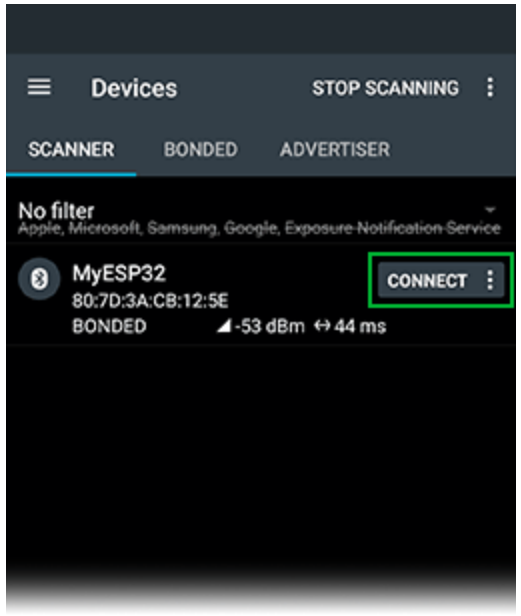
File > Examples > ESP32 BLE Arduino



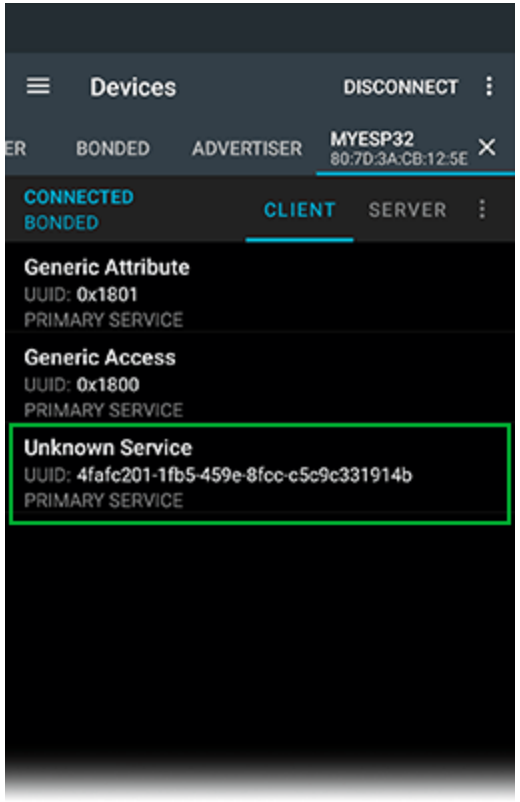
nRF Connect to Test

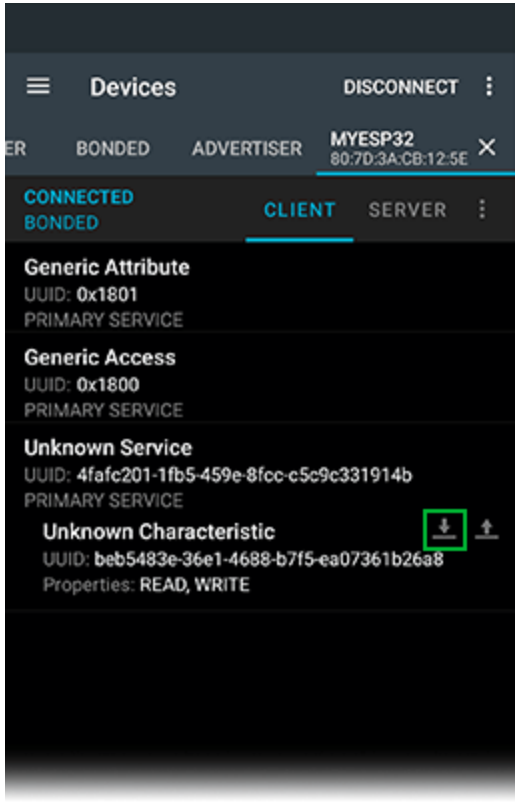


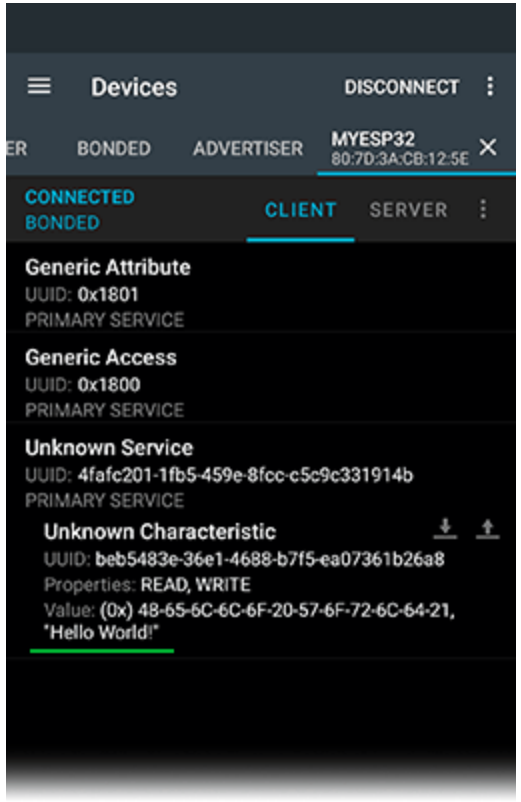




nRF Connect to Test





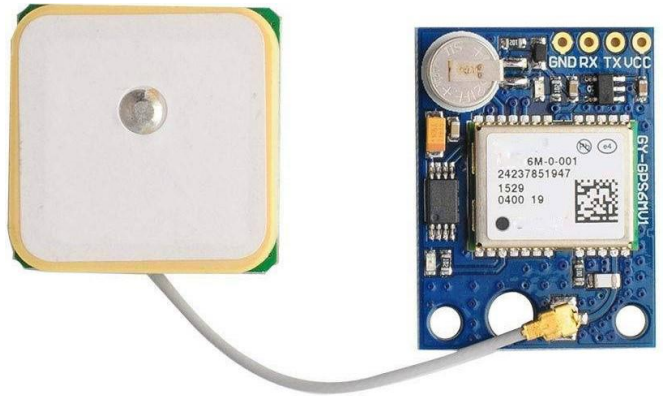


ESP32 GPS

Care-lab.kr

NEO-6M GPS 모듈 Spec

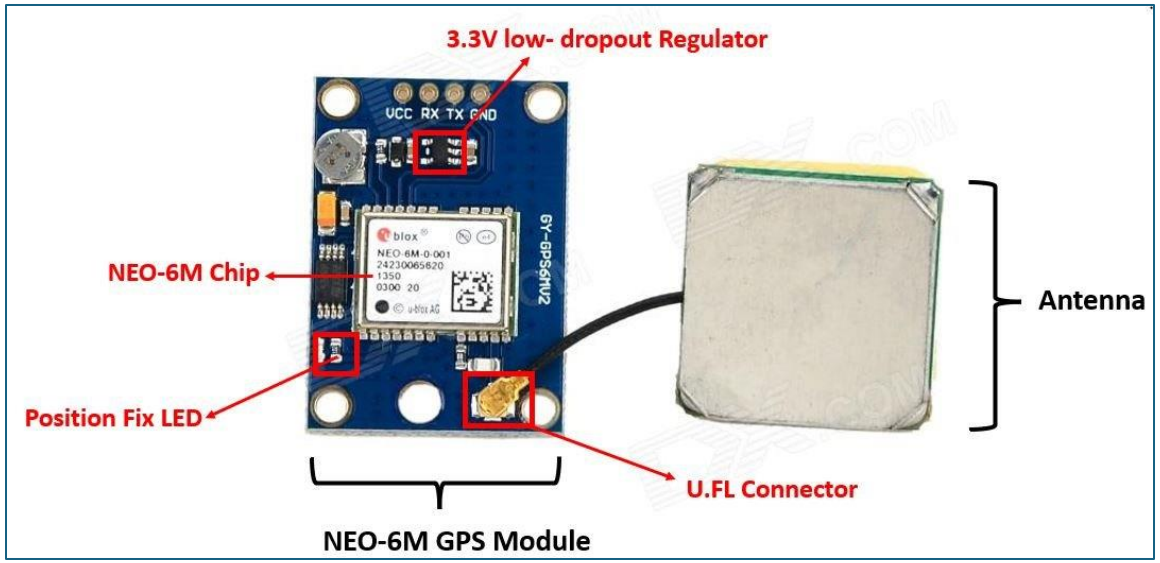
- GPS modules NEO-6M, 3V-5V power supply Universal
- Model: GY-GPS6MV2
- Destined module with ceramic antenna, signal super
- Save the configuration parameter data EEPROM Down
- With data backup battery
- There are LED signal indicator
- Antenna size 25 * 25mm
- Module size 25mm * 35mm
- PCB size 36mm*26mm
- Installation aperture 3mm
- Default Baud Rate: 9600
- Compatible with various flight control modules that provide GPS computer test software
- Tiny GPS 라이브러리
- 납땜은 되어 있지 않습니다.

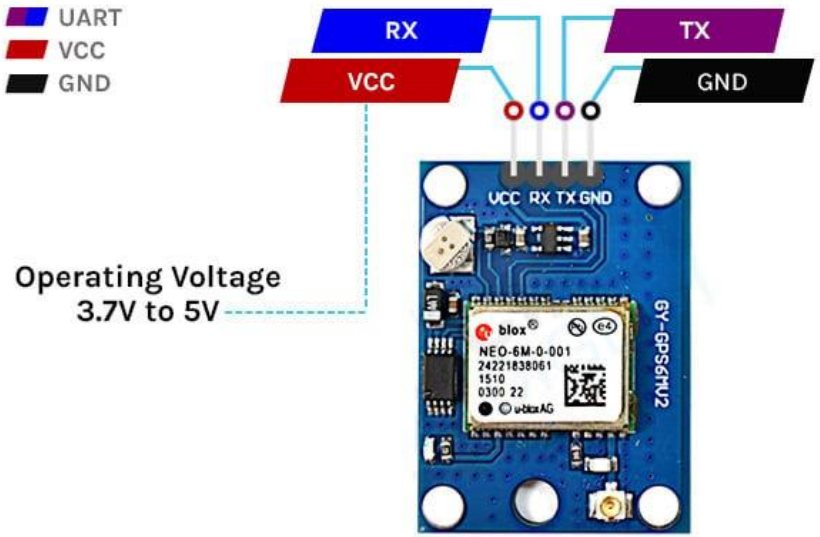


GY-GPS6MV2

참고: 디바이스마트 https://www.devicemart.co.kr/goods/view?no=1321968
Datasheet https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf

NEO-6M GPS 모듈 Interface





참고: 디바이스마트 https://www.devicemart.co.kr/goods/view?no=1321968

NMEA 문장 이해

NMEA는 National Marine Electronics Association의 약자로 , 거의 모든 GPS 수신기의 표준 메시지 형식

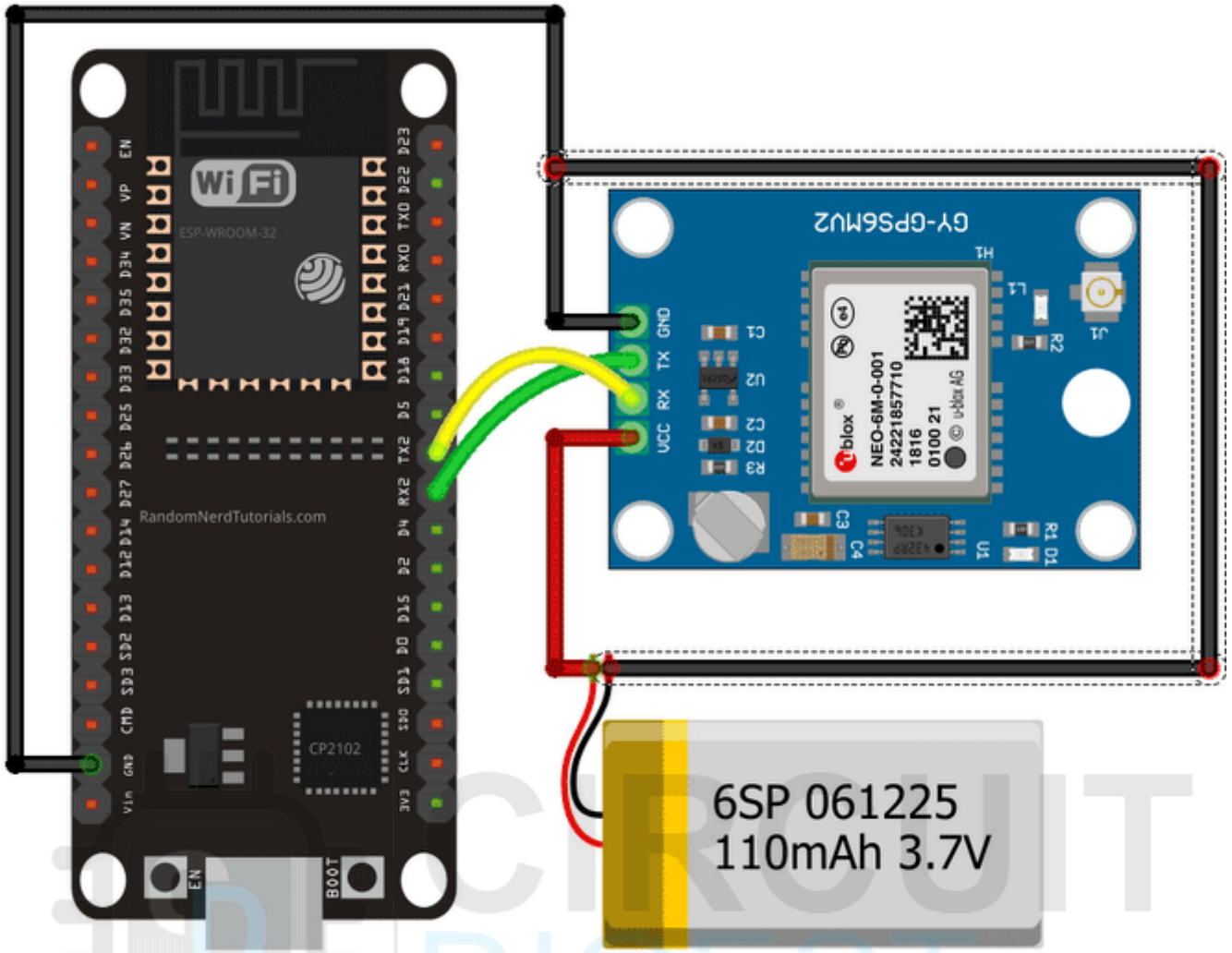
GPS - NMEA sentence information https://aprs.gids.nl/nmea /

$GPBOD - Bearing, origin to destination
$GPBWC - Bearing and distance to waypoint, great circle
$GPGGA - Global Positioning System Fix Data
$GPGLL - Geographic position, latitude / longitude
$GPGSA - GPS DOP and active satellites
$GPGSV - GPS Satellites in view
$GPHDT - Heading, True
$GPR00 - List of waypoints in currently active route
$GPRMA - Recommended minimum specific Loran-C data
$GPRMB - Recommended minimum navigation info
$GPRMC - Recommended minimum specific GPS/Transit data
$GPRTE - Routes
$GPTRF - Transit Fix Data
$GPSTN - Multiple Data ID
$GPVBW - Dual Ground / Water Speed
$GPVTG - Track made good and ground speed
$GPWPL - Waypoint location
$GPXTE - Cross-track error, Measured
$GPZDA - Date & Time

$GPGGA, 130113.00, 37XX.XXXX,N, 07XXX.XXXX, E,1,04,3.97,404.9,M,45.7,M,,*79

- **130113 - 데이터가 수집된 시간인 13:01:13 UTC를 나타냅니다.**
- **37XX.XXXX,N - 위도 37도 XX.XXXX' N**
- **07XXX.XXXX,E - 경도 007도 07XXX.XXXX,E**
- **1 - 품질 수정(0 = 유효하지 않음; 1 = GPS 수정; 2 = DGPS 수정; 3 = PPS 수정; 4 = 실시간 운동학; 5 = 플로트 RTK; 6 = 추정(추정 항법); 7 = 수동 입력 모드; 8 = 시뮬레이션 모드)**
- **04 – 추적 중인 위성 수**
- **3.97 – 위치의 수평 희석**
- **404.9, M – 해발 고도(미터)**
- **45.7, M – WGS84 타원체 위의 지오이드(평균 해수면) 높이**
- **비어 있음 - DGPS 업데이트 시간**
- **비어 있음 - DGPS 스테이션 ID**
- ***79 – 체크섬 데이터는 항상 *로 시작합니다.**

ESP32 NEO-6M GPS 모듈 회로도



* 코드와 문제 해결 https://fishpoint.tistory.com/12360

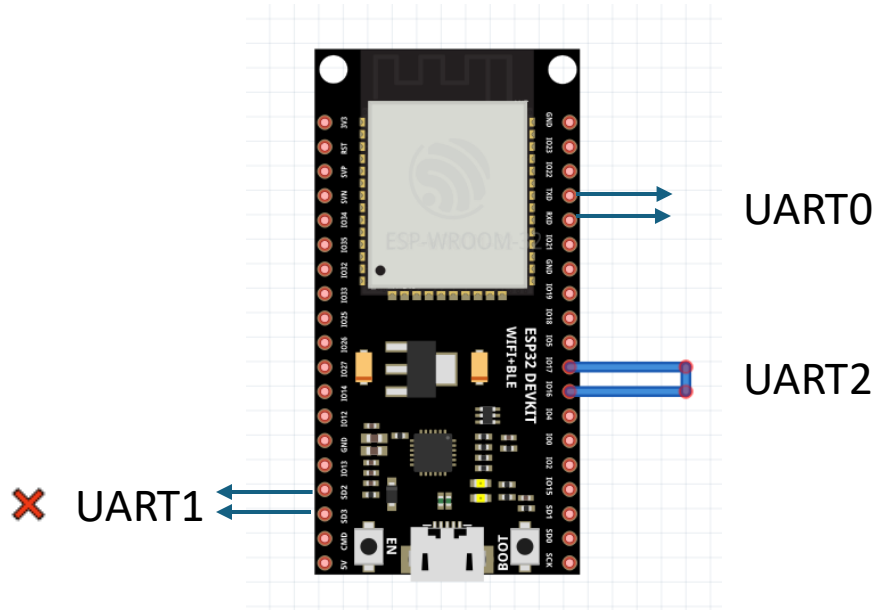
Code


```
#define RXD2 16
#define TXD2 17

#define GPS_BAUD 9600

// Create an instance of the HardwareSerial class for Serial 2
HardwareSerial gpsSerial(2);

void setup(){
  // Serial Monitor
  Serial.begin(115200);

  // Start Serial 2 with the defined RX and TX pins and a baud rate of 9600
  gpsSerial.begin(GPS_BAUD, SERIAL_8N1, RXD2, TXD2);
  Serial.println("Serial 2 started at 9600 baud rate");
}

void loop(){
  while (gpsSerial.available() > 0){
    // get the byte data from the GPS
    char gpsData = gpsSerial.read();
    Serial.print(gpsData);
  }
  delay(1000);
  Serial.println("----------------------------");
}
```

ESP32 - UART 시리얼(Serial)

UART	RX IO	TX IO	CTS	RTS
UART0	GPIO3	GPIO1	N/A	N/A
UART1	GPIO9	GPIO10	GPIO6	GPIO11
UART2	GPIO16	GPIO17	GPIO8	GPIO7

UART1(GPIO 9 및 GPIO10)에 관하여 – 이 GPIO는 ESP32 SPI 플래시 메모리에 연결되어 있으므로 그대로 사용할 수 없습니다. UART1을 사용하여 다른 장치와 통신하려면 HardwareSerial 라이브러리를 사용하여 다른 핀을 정의해야 합니다.



NEO-6M GPS 모듈 결과

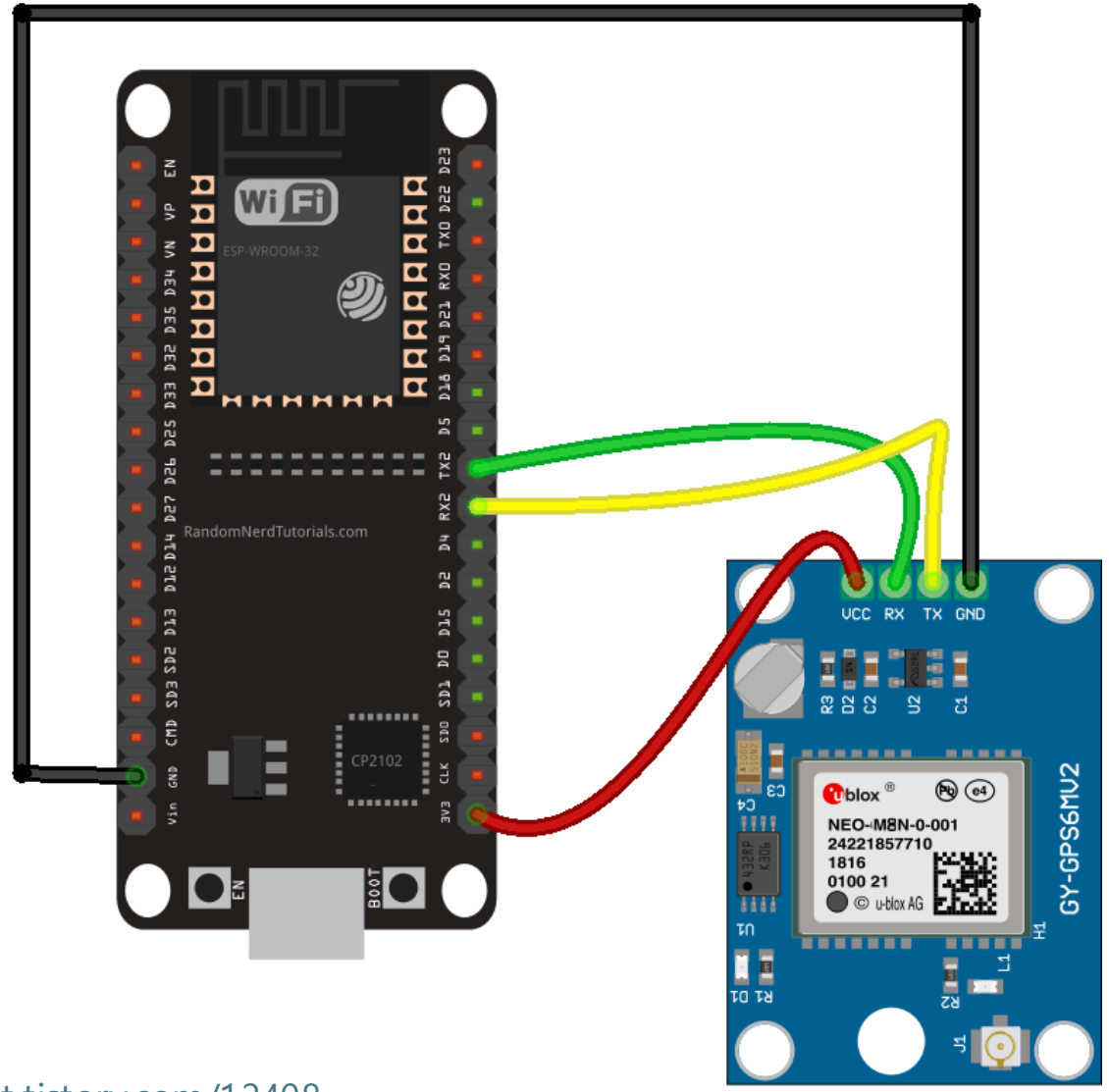
$GPRMC,041436.00,A,3722.34280,N,12657.10130,E,0.163,,291125,,,A*71
$GPVTG,,T,,M,0.163,N,0.303,K,A*27
$GPGGA,041436.00,3722.34280,N,12657.10130,E,1,06,1.20,47.7,M,18.5,M,,*68
$GPGSA,A,3,32,31,26,29,16,03,,,,,,,2.45,1.20,2.14*0D $GPGSV,2,1,06,03,22,296,2800,A,A*60
---------- $GPRMC,041437.00,A,3722.34257,N,12657.10132,E,0.207,,291125,,,A*79
$GPVTG,,T,,M,0.207,N,0.383,K,A*2E
$GPGGA,041437.00,3722.34257,N,12657.10132,E,1,06,1.20,47.5,M,18.5,M,,*63
$GPGSA,A,3,32,31,26,29,16,03,,,,,,,2.45,1.20,2.14*0D $GPGSV,2,1,06,03,22,296,2800,A,A*69

NEO-M8N GPS 모듈 Interface

- Specifications:
- GPS line order:
- black wire: GND
- Green line: TX
- Yellow Line: RX
- Red Line: + 5V
- White Line: SDA
- Orange Line: SCL
- Color: Black
- Size:53*53*10mm/2.09*2.09*0.39inch

- GPS/QZSS L1 C/A, GLONASS L10F, BeiDou B1
- SBAS L1 C/A: WAAS, EGNOS, MSAS
- Galileo-ready E1B/C (NEO-M8N)
- Nav. update rate1 Single GNSS: up to 18 HZ
- Concurrent GNSS: up to 10 Hz
- Position accuracy2 2.0 m CEP
- Acquisition2 Cold starts: 26 s
- Aided starts: 2 s
- Reacquisition: 1.5 s
- Sensitivity2 Tracking & Nav: –167 dBm
- Cold starts: –148 dBm
- Hot starts: -156 dBm
- Assistance AssistNow GNSS Online
- AssistNow GNSS Offline (up to 35 days)3
- AssistNow Autonomous (up to 6 days)
- OMA SUPL & 3GPP compliant
- Oscillator TCXO (NEO-M8N/Q),
- Crystal (NEO-M8M)

NEO-M8N GPS 모듈 Interface



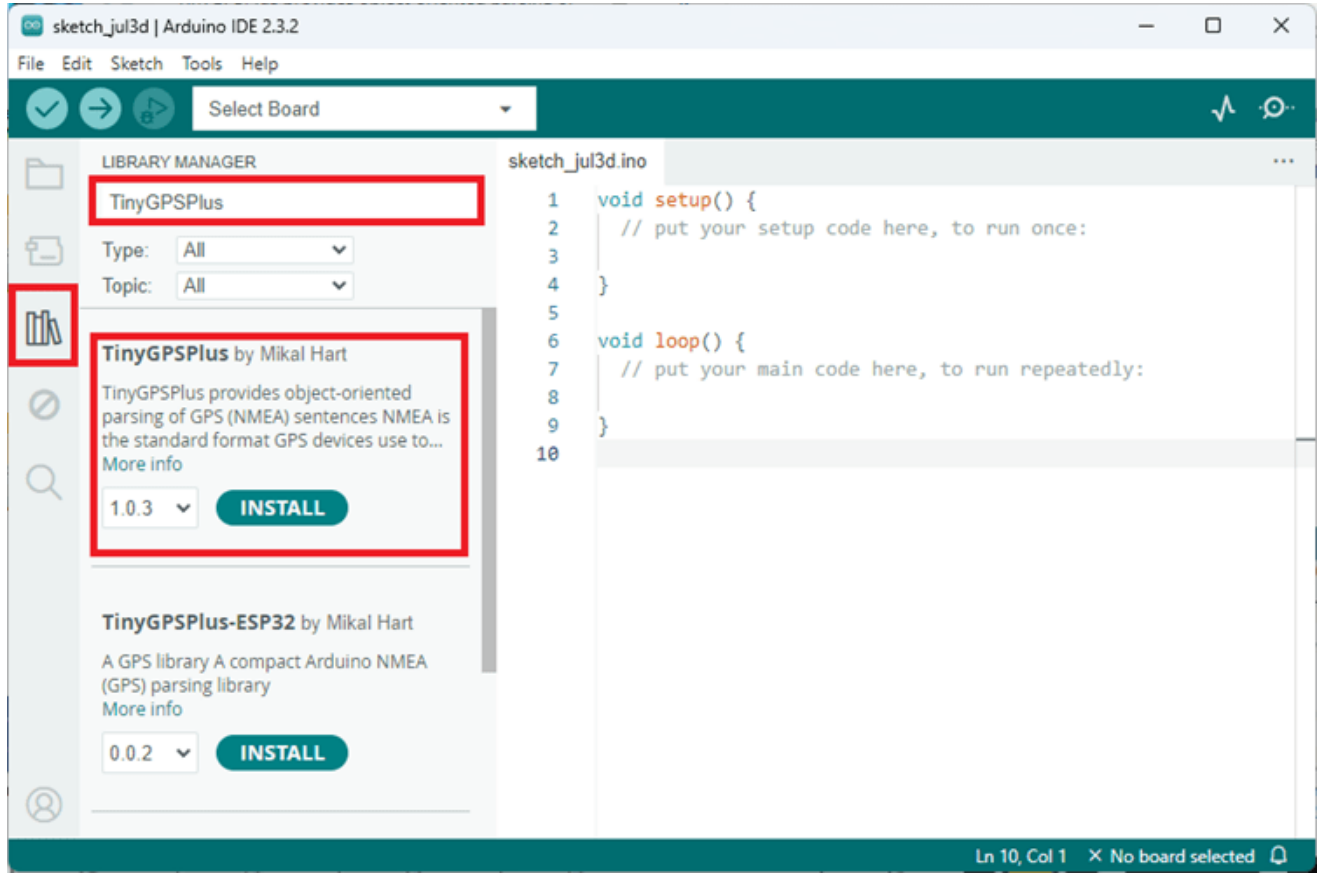
상세 사용법과 코드 https://fishpoint.tistory.com/12408

NEO-M8N GPS 모듈 실습 단계

1. 직렬로 NEO-M8N GPS 모듈을 ESP32에 연결합니다.
2. 원시 GPS 데이터를 얻으세요.
3. 원시 데이터를 분석하여 선택되고 읽을 수 있는 GPS 정보를 얻습니다.
4. 현재 위치를 확인하세요.
5. 위치를 microSD 카드의 파일에 기록합니다.
6. Google Earth에서 읽을 수 있는 .kml 파일 로 데이터를 변환합니다.
7. 경로를 표시하려면 .kml 파일을 Google Earth에 업로드하세요.

TinyGPSPlus 라이브러리를 사용한 NMEA 문장 구문 분석

TinyGPSPlus 라이브러리 설치



Code

neo_m8ntinyplus.ino

결과

```
LAT: 37.372395
LONG: 126.951942
SPEED (km/h) = 1.06
ALT (min)= 37.70
HDOP = 1.69
Satellites = 6
Time in UTC: 2025/12/8,3:27:40

LAT: 37.372395
LONG: 126.951942
SPEED (km/h) = 1.06
ALT (min)= 38.10
HDOP = 1.69
Satellites = 6
Time in UTC: 2025/12/8,3:27:41
```

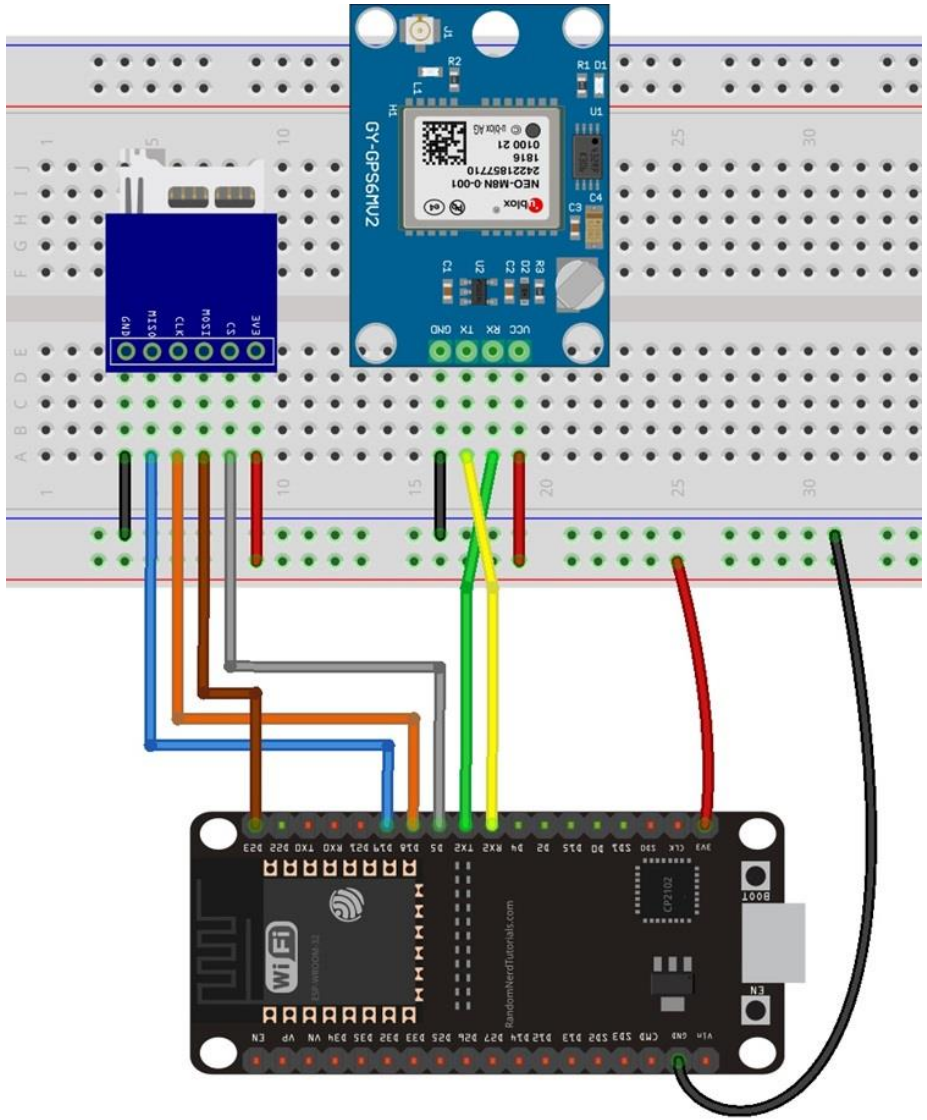
TinyGPSPlus 라이브러리를 데이터 출력

if (gps.location.isUpdated()) {

Latitude	gps.location.lat()
Longitude	gps.location.lng()
Speed (km/h)	gps.speed.kmph()
Altitude (meters)	gps.altitude.meters()
HDOP	gps.hdop.value()
The number of visible satellites	gps.satellites.value()
Year	gps.date.year()
Month	gps.date.month()
Day	gps.date.day()
Hour	gps.time.hour()
Minutes	gps.time.minute()
Seconds	gps.time.second()

위도 gps.위치.위도()
경도 gps.위치.lng()
속도(km/h) gps.속도.kmph()
고도(미터) gps.고도.미터()
HDOP gps.hdop.value()
눈에 보이는 위성의 수 gps.위성.값()
년도 gps.날짜.년()
월 gps.날짜.월()
낮 gps.날짜.요일()
시간 gps.시간.시간()
분 gps.시간.분()
초 gps.시간.초()

Google Earth에서 GPS 로거 및 경로 표시



NEO-M8N GPS Module	ESP32
VCC	3V3
RX	TX2 (GPIO 17)
TX	RX2 (GPIO 16)
GND	GND

MicroSD Card Module	ESP32
3V3*	3V3
CS	GPIO 5
MOSI	GPIO 23
CLK	GPIO 18
MISO	GPIO 19
GND	GND

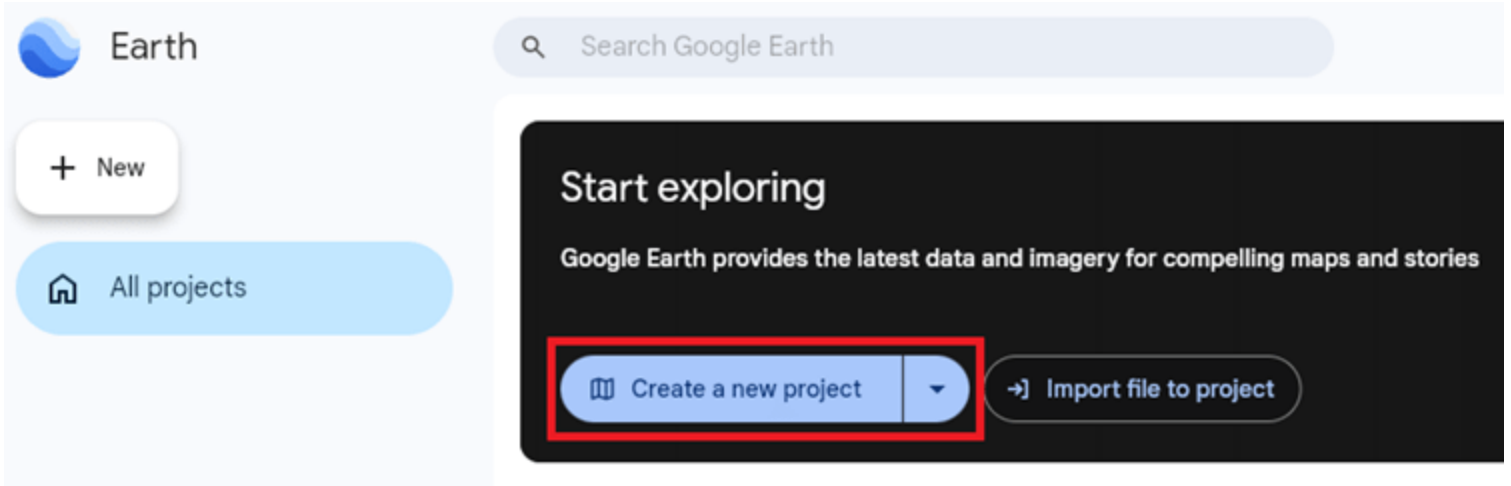
Code: neo-m8n-sdcard.ino

microSD 카드와 GPS 모듈을 초기화하고, 데이터 포인트가 사용 가능해지고 새로운 포인트가 이전 포인트에서 1미터 이상 떨어지면 GPS 데이터를 저장합니다.

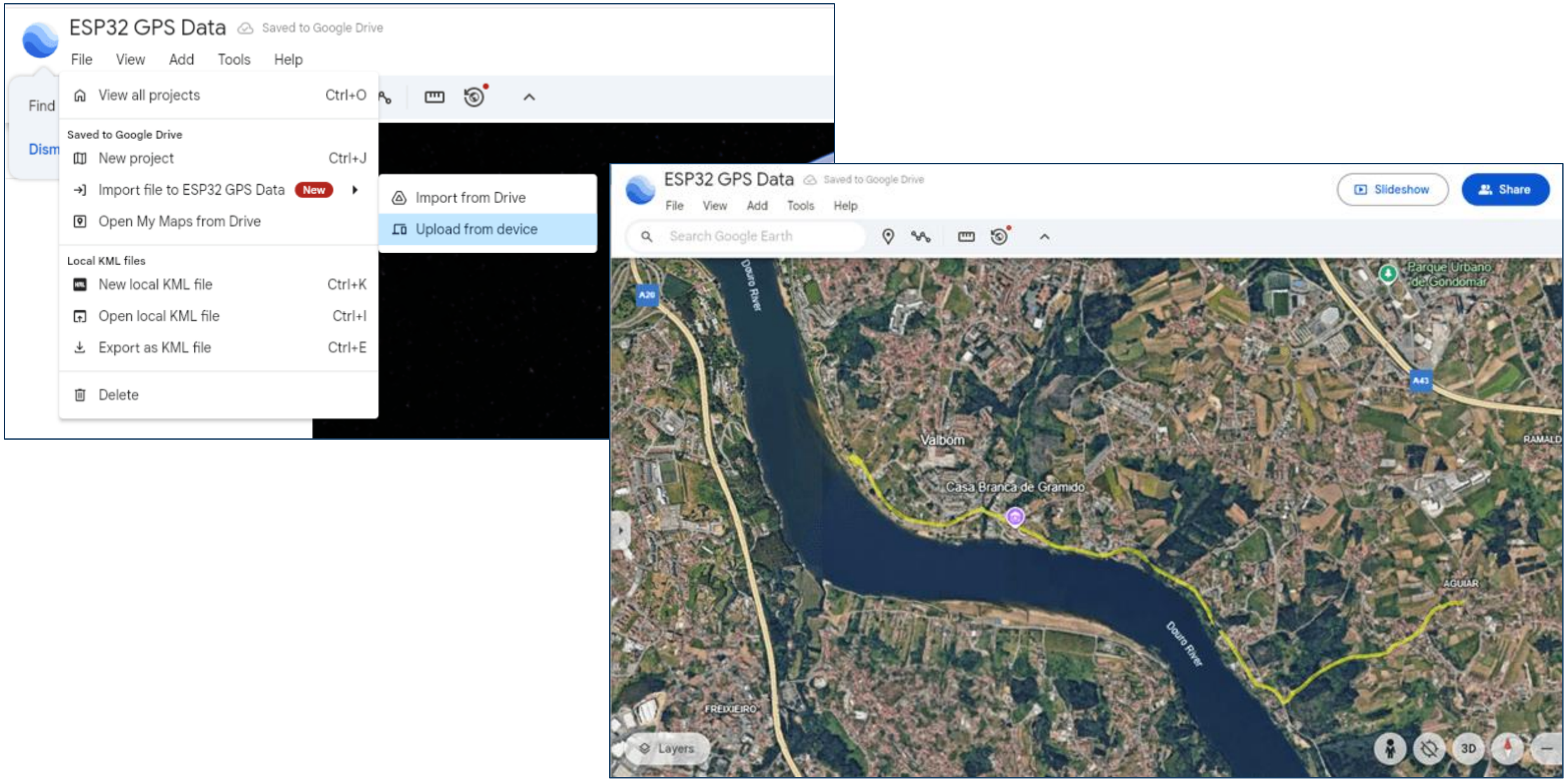
데이터를 기록한 후, microSD 카드를 모듈에서 분리하여 컴퓨터에 연결하세요. microSD 카드에는 경로 좌표가 저장된 data.txt 파일이 있습니다.

KML 파일
data.txt 파일을 .kml 형식으로 변환
https://developers.google.com/kml/documentation/kml_tut

Google Earth에 경로 표시



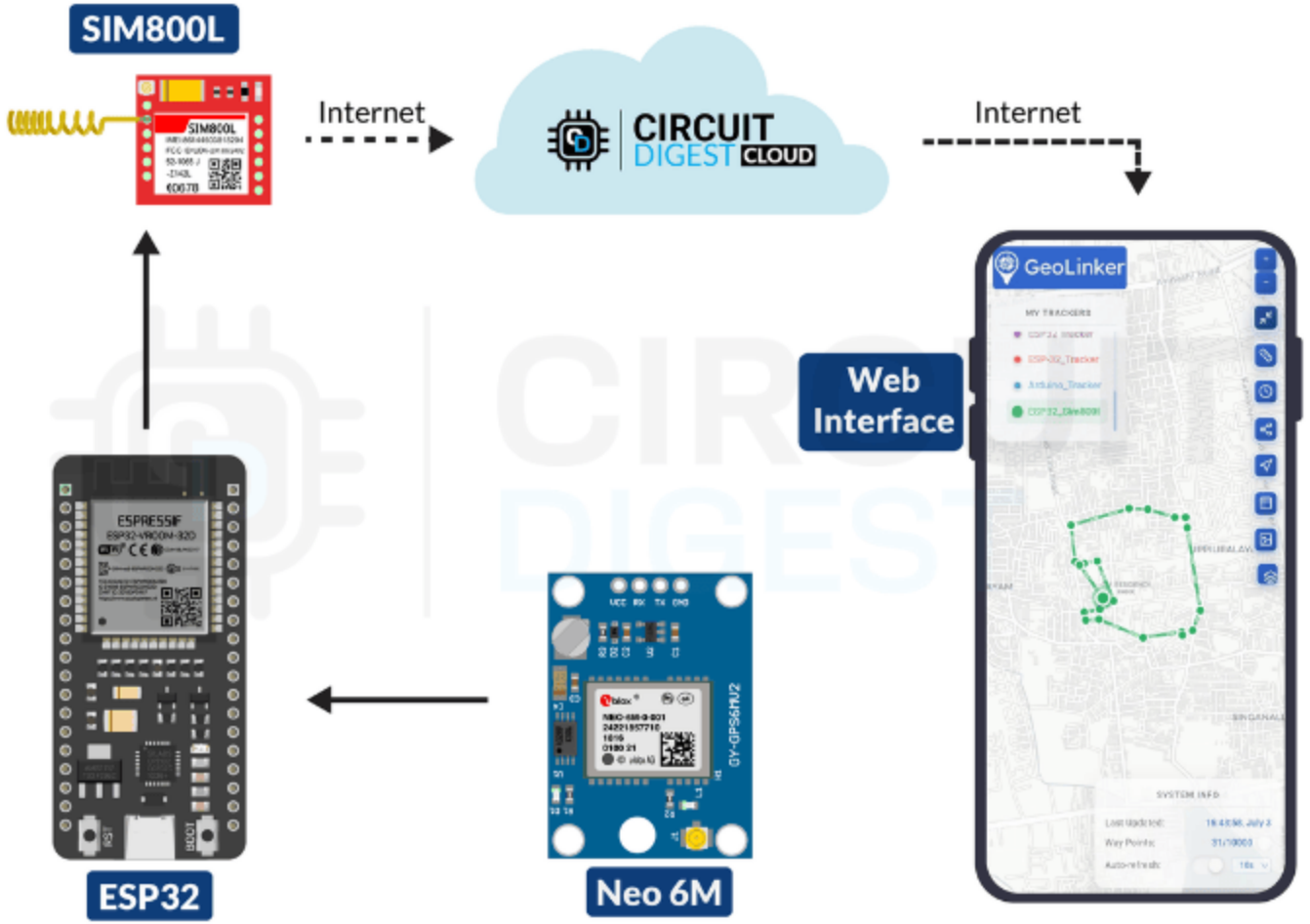
구글 어스에 경로 표시


ESP32 GPS Data
Saved to Google Drive
File
View
Add
Tools
Help
View all projects Ctrl+O
Saved to Google Drive
New project Ctrl+J
Import file to ESP32 GPS Data New
Open My Maps from Drive
Local KML files
New local KML file Ctrl+K
Open local KML file Ctrl+I
Export as KML file Ctrl+E
Delete
Import from Drive
Upload from device
Slideshow
Share
Search Google Earth
Douro River
A20
Parque Urbano de Gondomar
A43
Valbom
Casa Branca de Gramido
RAMALD
AGUIAR
FREIXIEIRO
Layers
3D

셀룰러 모듈 SIM800L 인터페이스

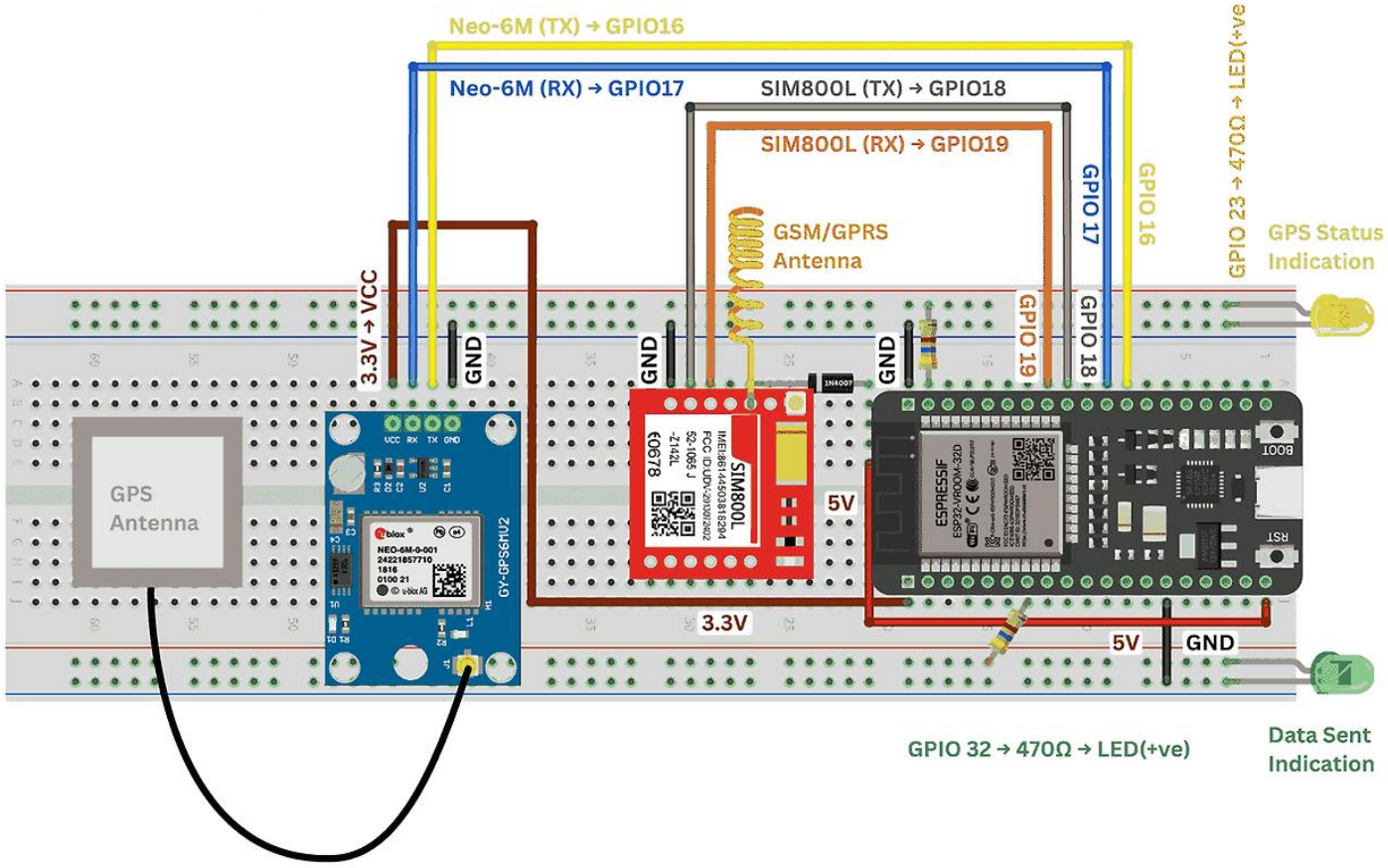
SIM800L GSM/GPRS Module 자료 https://www.espboards.dev/sensors/sim800l/#code-examples
구현 자료 https://fishpoint.tistory.com/12402

GSM 모듈은 Global System for Mobile Communications(이동통신을 위한 범용 시스템) 기술을 사용하여, 기기가 휴대폰처럼 GSM 네트워크를 통해 음성 및 데이터 통신(전화, 문자, 인터넷)을 할 수 있게 해주는 하드웨어 장치입니다. 이는 주로 마이크로컨트롤러(MCU)가 포함된 임베디드 시스템(사물 인터넷, 스마트 기기 등)에서 컴퓨터와 휴대폰의 기능을 결합시켜 무선 통신을 구현하는 데 사용됩니다.



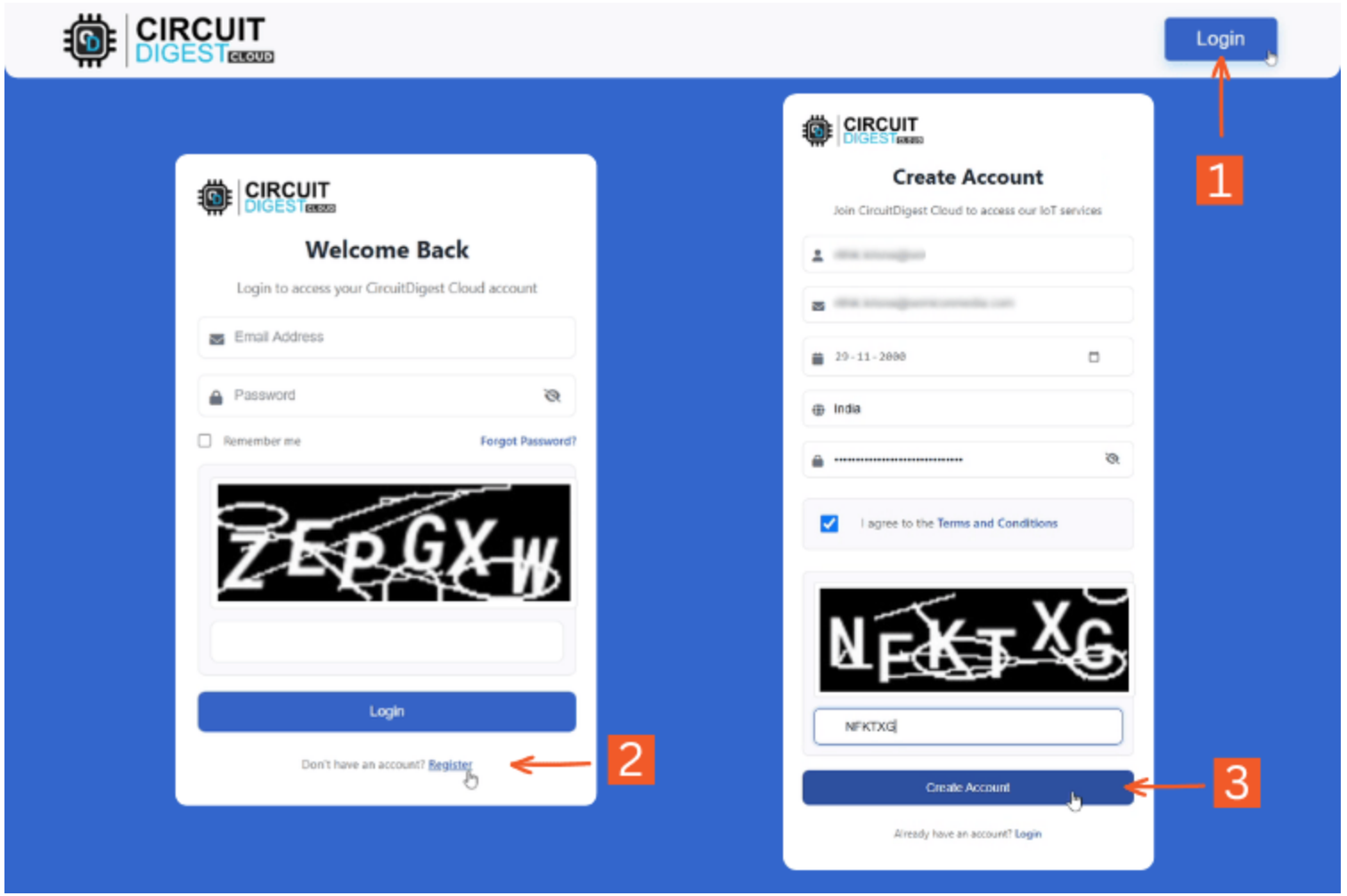
ESP32 GPS 추적기 구성 요소

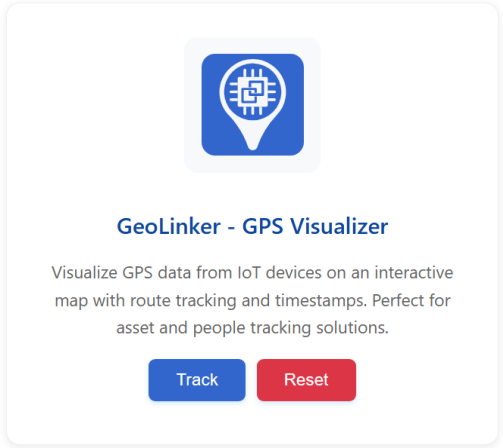
Component	Purpose	Voltage
ESP32 Development Board	Main processing unit & Wi-Fi	3.3V (5V input)
SIM800L GSM Module	GPRS cellular connectivity	3.7-4.2V
NEO-6M GPS Module	Satellite location tracking	3.3-5V



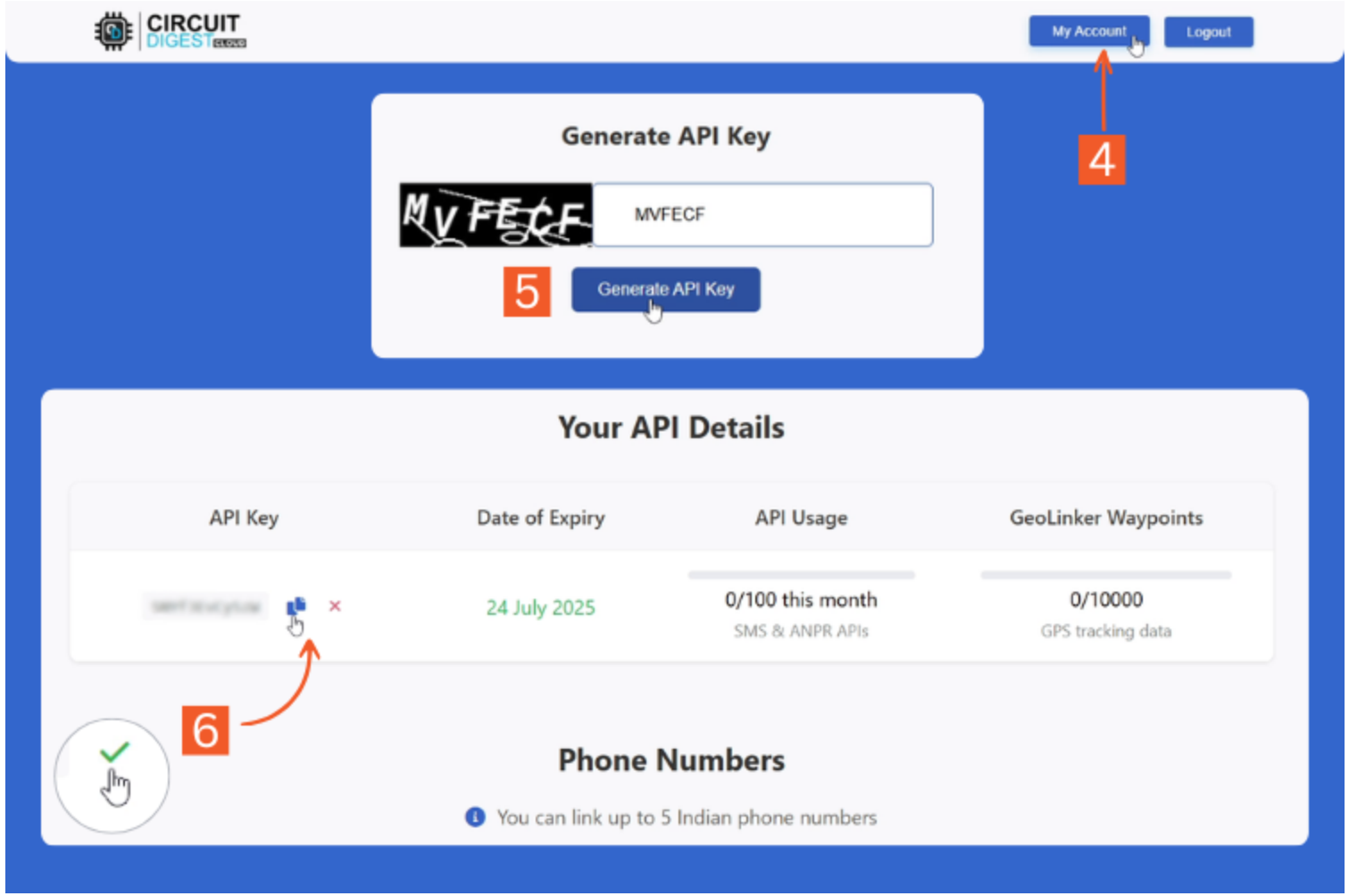
GPS 추적을 위한 GeoLinker 클라우드 플랫폼 설정

Circuit Digest Cloud에 새 계정을 만드세요. https://www.circuitdigest.cloud/





GeoLinker API 키 생성



참고: 현재 SMS 및 ANPR API의 경우 키당 사용 횟수 는 100회로 제한되며 , GPS 추적의 경우 데이터 포인트는 10,000개로 제한 됩니다. 이 제한에 도달하면 100회 사용을 추가로 제공하는 새 키를 생성할 수 있습니다. 이 사용 횟수 제한은 서버 과부하를 방지하기 위해 적용됩니다.

ESP32 GPS 추적기 프로그래밍

1. **라이브러리 및 종속성 #include <GeoLinker.h>**

2. **Neo-6M GPS 모듈 ESP32 인터페이스**

```
HardwareSerial gpsSerial(1);
#define GPS_RX 16
#define GPS_TX 17
#define GPS_BAUD 9600
```

3. **GSM 모듈 구성**

```
HardwareSerial gsmSerial(2);
#define GSM_RX 18
#define GSM_TX 19
#define GSM_BAUD 9600
#define GSM_PWR_PIN -1
#define GSM_RST_PIN -1
```

4. **셀룰러 네트워크 구성**

```
const char* apn = "yourAPN";
const char* gsmUser = nullptr;
const char* gsmPass = nullptr;
```

- APN은 Access Point Name: 스마트폰 같은 기기가 통신사 망을 통해 인터넷이나 MMS 같은 데이터 서비스를 이용할 수 있도록 연결해 주는 '게이트웨이' 역할을 하는 설정 정보

ESP32 GPS 추적기 프로그래밍

5. GeoLinker 클라우드 구성

```
const char* apiKey = "xxxxxxxxxxxx";
const char* deviceID = "ESP32_Sim800l";
```

데이터 전송 및 버퍼 설정:
연결 및 시간대 설정:

6. LED 상태 표시기

- 위치가 성공적으로 업로드되면 DataSent_LED가 깜박입니다.
- GPS가 유효한 위치 데이터를 받으면 GPSFix_LED가 켜집니다.

7. GeoLinker 초기화

```
GeoLinker geo;
```

geo라는 강력한 객체가 모든 추적, GSM 및 클라우드 작업을 관리합니다.

ESP32 GPS 추적기 프로그래밍

8. 설정GeoLinker 클라우드 구성

```
const char* apiKey = "xxxxxxxxxxxx";
const char* deviceID = "ESP32_Sim800l";
```

데이터 전송 및 버퍼 설정:
연결 및 시간대 설정:

6. LED 상태 표시기

- 위치가 성공적으로 업로드되면 DataSent_LED가 깜박입니다.
- GPS가 유효한 위치 데이터를 받으면 GPSFix_LED가 켜집니다.

7. GeoLinker 초기화

```
GeoLinker geo;
```

geo라는 강력한 객체가 모든 추적, GSM 및 클라우드 작업을 관리합니다.

코드, 회로도 저장 Git https://github.com/Circuit-Digest/GeoLinker.git