

# ESP32 동작 실습

2024.06

(주)다인시스

# 목 차

---

## ◆ 가상 머신 활용

가상머신 실습환경 구축

ESP32 제어

WLAN 네트워크

# 가상 실습 환경

---

## ◆ 가상 머신

- HW 장치를 Emulation(Simulation) 해 주는 환경
- 다양한 ESP32 가상 머신 사용 가능

## ◆ 주요 ESP32 지원 가상 머신

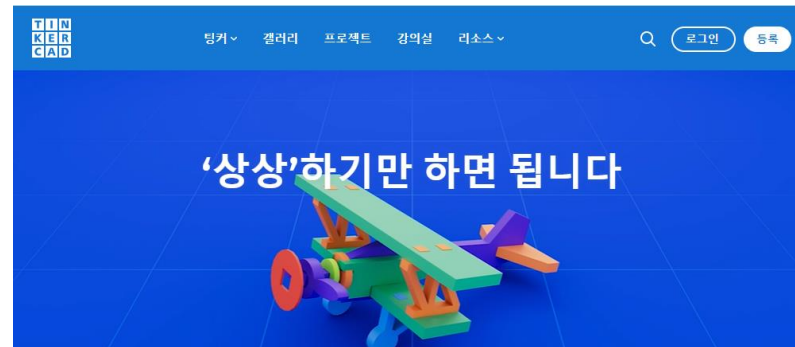
- Tinkercad Circuit
  - ❖ <https://www.tinkercad.com/circuits>
- Virtronics
  - ❖ <https://virtronics.com.au/Simulator-for-Arduino.html>
- Proteus
  - ❖ <https://www.labcenter.com/>
- QEMU ESP32
  - ❖ [https://github.com/Ebiroll/qemu\\_esp32](https://github.com/Ebiroll/qemu_esp32)
- ESP32 웹 시뮬레이터 (WOKWI)
  - ❖ <https://wokwi.com/>

# Thinkcad Circuit

---

## ◆ Thinkcad Circuit 가상 머신

- <https://www.tinkercad.com/circuits>
- 온라인 회로 시뮬레이터와 디자인 툴
- 브라우저에서 회로 디자인, 편집, 시뮬레이션, 프로토타입 제작, PCB 디자인 가능
- 접근성이 높고 사용하기 쉬워, 교육적인 용도로도 많이 활용



## ◆ Thinkcad Circuit 활용

- 간단한 LED 회로부터 복잡한 마이크로컨트롤러와 센서를 사용한 IoT 디바이스까지 다양한 회로를 설계 가능
- 시뮬레이션을 통해 회로의 동작을 미리 확인 가능
- 프로토타입 제작 및 PCB 디자인을 통해 실제 제품 제작 가능

# Virtronics

## ◆ Virtronics 가상 머신

- 임베디드 시스템 개발용 가상 시뮬레이션 툴 개발 회사
- 다양한 툴 제공
- Arduino IDE와 연동 가능한 Virtual Breadboard 툴 제공

The screenshot shows the Virtronics website interface. On the left is a vertical navigation menu with links: Home, News, Simulator for Arduino (highlighted), Shields, Links, About Us, Forum, and Products. The main content area features the Virtronics logo at the top right. Below the logo, there are navigation links 'Prev<<' and '>>Next'. The main heading is 'Simulator for Arduino v1.14'. Below this, there are three user testimonials in quotes. To the right of the testimonials is a red 'Buy Now' button. Below the button, the pricing is listed: 'Pro version licence \$19.99 (support until Dec-31)'. Further down, a paragraph states 'Simulator for Arduino is the most full featured Arduino Simulator available at the present time (watch the latest vic'. Below this, a section titled 'The benefits and features of the Arduino Simulator are:' lists three bullet points: 'The ability to teach and demonstrate the inner workings of an Arduino sketch', 'Test out a sketch without the hardware, or prior to purchasing hardware', and 'Debug a sketch'.

Virtronics

[Prev<<](#) --- [>>Next](#)

## Simulator for Arduino v1.14

*"This is a life saver and honestly an excellent software."*

*"I've just bought your wonderfull product. Keep up the good work! "*

*"Thanks! Works a treat!"*

[Buy Now](#)

Pro version licence  
\$19.99 (support until  
Dec-31)

Simulator for Arduino is the most full featured Arduino Simulator available at the present time (watch the latest vic

The benefits and features of the Arduino Simulator are:

- The ability to teach and demonstrate the inner workings of an Arduino sketch
- Test out a sketch without the hardware, or prior to purchasing hardware
- Debug a sketch

# Proteus

## ◆ Proteus Design Suit 가상 머신

- <https://www.labcenter.com/>
- 영국 Labcenter Electronics Ltd.에서 개발한 전자제품 설계 및 시뮬레이션 SW
- Proteus Design Suite는 PCB 설계 및 제작, 마이크로컨트롤러 프로그래밍 및 시뮬레이션, 가상 프로토타이핑, 시스템 수준 시뮬레이션, 오실로스코프 및 함수 발생기 시뮬레이션 등의 기능 제공



# QEMU

## ◆ QEMU(Quick EMUlator) 가상 머신

- [https://github.com/Ebiroll/qemu\\_esp32](https://github.com/Ebiroll/qemu_esp32)
- 가상화를 위한 오픈 소스 머신 에뮬레이터 및 가상화 솔루션
- CPU, 메모리, 네트워크 인터페이스, 디스크 등 다양한 하드웨어를 가상화
- 호스트 시스템에서 가상화된 게스트 시스템을 실행 가능
- QEMU는 다양한 아키텍처와 플랫폼 지원
- 간단한 컴퓨터 시스템에서부터 복잡한 임베디드 시스템까지 다양한 환경에서 사용
- KVM(Kernel-based Virtual Machine)과 함께 사용되어 가상화의 성능 향상
- 라이브러리 형태로 사용 가능하여 다양한 가상화 솔루션을 개발 가능
- GDB와 같은 디버깅 도구와 함께 사용할 수 있어 디버깅에 유용

```
> xtensa-esp32-elf-gdb build/app-template.elf -ex 'target remote:1234'

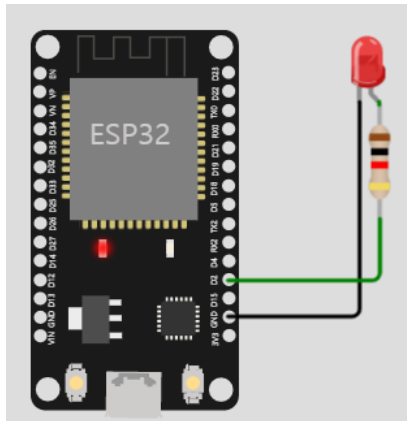
(gdb) x/10i $pc
(gdb) x/10i 0x40000400
(gdb) p/x $a0

Call0 will set a0 with return adress, this will list the instructions at the call
(gdb) x/10i $a0-3
(gdb) info symbol 0x400d0eb1
(gdb) layout next
(gdb) b app_main
(gdb) continue
(gdb) list *$pc
Ctrl-X and the 0 opens the source
```

# ESP32 웹 시뮬레이터

## ◆ Wokwi 가상 머신

- <https://wokwi.com/>
- Wokwi is an online Electronics simulator.
- Simulate Arduino, ESP32



```
WOKWI SAVE SHARE esp32-arduino.ino by urish  
esp32-blink.ino diagram.json Library Manager  
  
1  #define LED 2  
2  
3  void setup() {  
4      pinMode(LED, OUTPUT);  
5  }  
6  
7  void loop() {  
8      digitalWrite(LED, HIGH);  
9      delay(500);  
10     digitalWrite(LED, LOW);  
11     delay(500);  
12 }  
13
```

## ◆ Wokwi ESP32 플랫폼

- Espressif Systems에서 개발한 ESP32 DevKitC V4 보드와 호환
- ESP32 모듈의 WiFi 및 Bluetooth 모듈, GPIO, ADC, DAC 등의 다양한 핀 시뮬레이션 가능
- Wokwi에서 제공하는 ESP32 라이브러리와 함께 Arduino IDE 또는 PlatformIO와 같은 개발 도구를 사용하여 코드 작성 및 시뮬레이션 가능
- ESP32용 OLED 디스플레이, LED 매트릭스, 터치스크린 등의 추가 모듈도 지원



# ESP32-S3 웹 시뮬레이터

---

## ◆ ESP32 보드 지원 가상 머신

- <https://docs.wokwi.com/guides/esp32>
- Simulate Arduino, ESP32-S3 (Beta Version)

## ESP32 boards

Name	Chip	Description
ESP32 DevKit v1	ESP32	Popular ESP32 development board
ESP32-S2-DevKitM-1	ESP32-S2	Entry-level ESP32-S2 development board
Franzininho WiFi	ESP32-S2	Board by the Franzininho Community
Wemos S2 mini	ESP32-S2	Small ESP32-S2 board by Wemos
ESP32-C3-DevKitM-1	ESP32-C3	Entry-level ESP32-C3 development board
Rust Board ESP32-C3	ESP32-C3	ESP32-C3 board designed for Rust trainings
ESP32-S3-DevKitC-1	ESP32-S3	Entry-level ESP32-C3 development board (beta)
ESP32-C6-DevKitC-1	ESP32-C6	Entry-level ESP32-C6 (alpha)

# 목 차

---

가상머신 활용

◆ 가상머신 실습 환경 구축

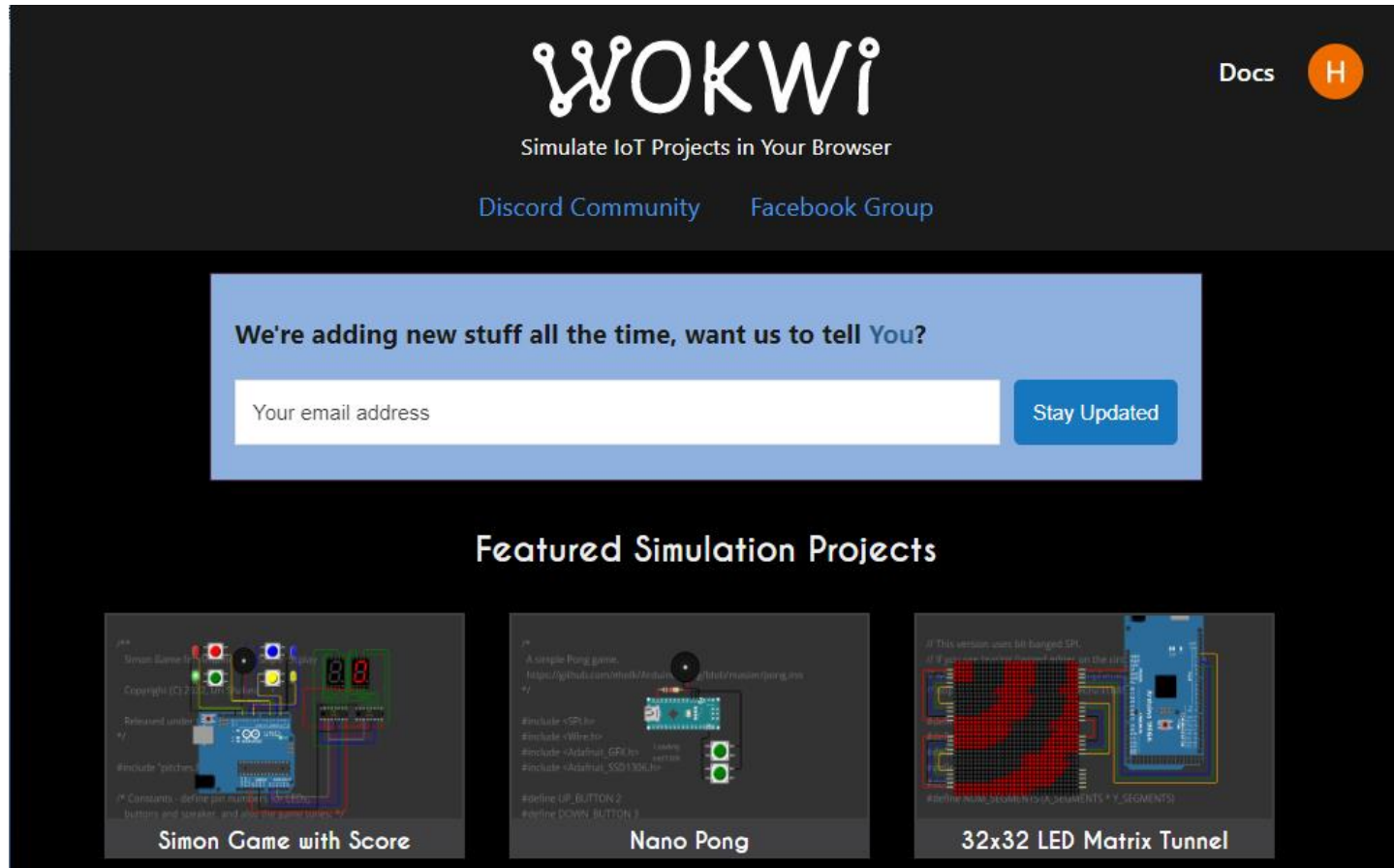
ESP32 제어

WLAN 네트워크

# ESP32 웹 시뮬레이터

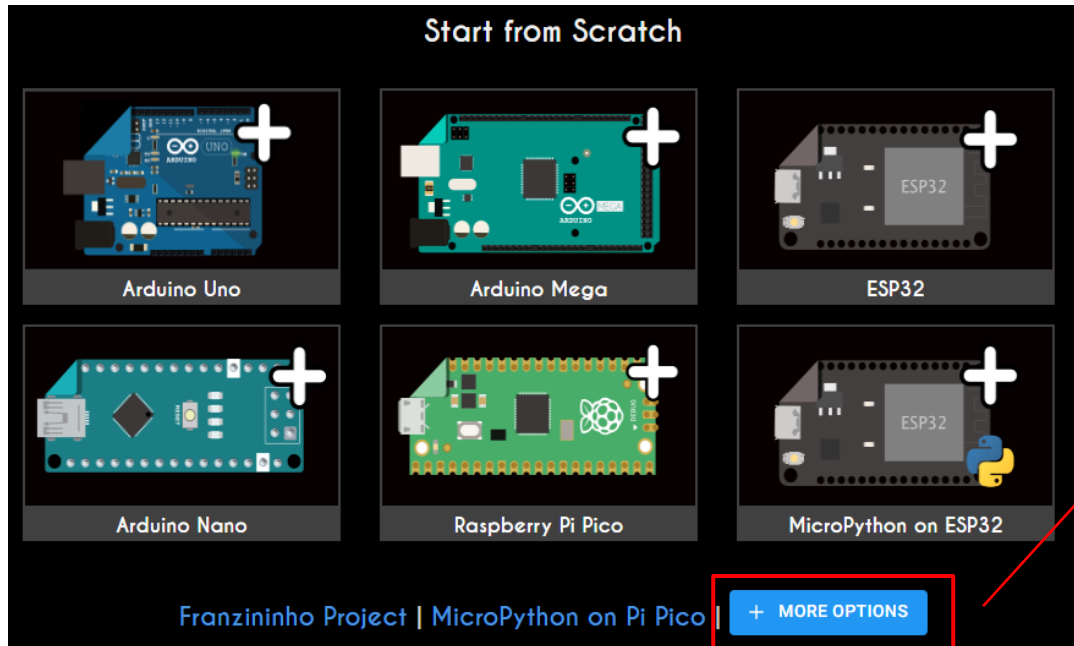
## ◆ Wokwi 시뮬레이터 웹 접속

- <https://wokwi.com/>



# 신규 프로젝트 실행

## ◆ Start from Scratch



- ATtiny85
- ESP32-S2
- ESP32-S3 (beta)
- ESP32-C3
- Raspberry Pi Pico (SDK)
- CircuitPython on Raspberry Pi Pico
- Raspberry Pi Pico W
- Raspberry Pi Pico W (SDK)
- MicroPython on Raspberry Pi Pico W
- Franzininho WiFi (ESP32-S2)
- Rust on ESP32
- Rust on ESP32-S2
- Rust on ESP32-C3
- Rust on ESP32 Rust Board

# ESP32 Simulation Feature

Peripheral	ESP32	S2	S3	C3	Notes
Processor core(s)	✓	✓	✓	✓	
GPIO	✓	✓	●	✓	Interrupts supported
IOMUX	✓	✓	✓	✓	
PSRAM	✓	✓	✓	—	4MB of external SRAM *
UART	✓	✓	✓	✓	
USB	—	✓	✓	✗	Support for UART over USB (CDC)
I2C	✓	✓	✓	✓	Master only. 10-bit addressing not supp
I2S	✗	✗	✗	✗	Open for voting
SPI	✓	✓	✓	✓	
TWAI	✗	✗	✗	✗	
RMT	●	●	✗	●	Transmit-only, use to control NeoPixels
LEDC PWM	✓	✓	✗	✓	Used by analogWrite(), Servo, Buzzer, et
MCPWM	✗	—	✗	—	
DMA	●	●	✗	✗	

WiFi	✓	✓	✓	✓	See the <a href="#">ESP32 WiFi Guide</a>
Bluetooth	✗	—	✗	✗	Open for voting
Timers	●	✓	●	✓	
Watchdog	✗	✗	✗	✗	
RTC	●	●	●	●	Only RTC Pull-up / Pull-down resistors
ADC	✓	✓	✗	✓	Note: analogRead() returns values up to 4095
RNG	✓	✓	✓	✓	Random Number Generator
AES Accelerator	✓	✓	✗	✓	
SHA Accelerator	✓	✓	✗	✓	
RSA Accelerator	✓	✓	✗	✓	
Hall Effect Sensor	✗	—	✗	—	
ULP Processor	✗	✗	✗	✗	
GDB Debugging	✓	✓	✓	✓	Works with <a href="#">Wokwi for VS Code</a>

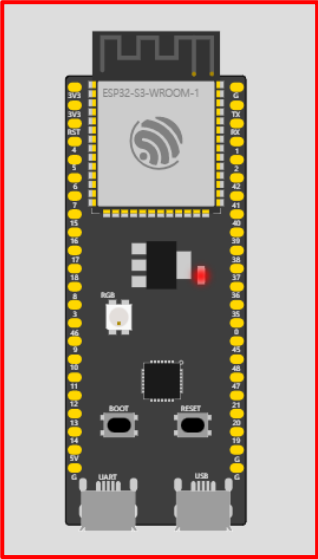
# ESP32-S3 프로젝트 시작

WOKWI SAVE SHARE Docs H

sketch.ino diagram.json Library Manager

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   Serial.begin(115200);  
4   Serial.println("Hello, ESP32-S3!");  
5 }  
6  
7 void loop() {  
8   // put your main code here, to run repeatedly:  
9   delay(10); // this speeds up the simulation  
10 }  
11
```

Simulation 00:04.783 97%



Load: 0x403cc700, len: 0x2364  
SHA-256 comparison failed:  
Calculated: bea01f04c6f0e287a682f128805e3fce115955179100e98dbf412fe7697f8bdc  
Expected: 08ea492e448f88de75319ed18ac319444e578d9c6bc7003c5c4807382bf389bd  
Attempting to boot anyway...  
entry 0x403c98ac  
Hello, ESP32-S3!

Arduino Programing 환경

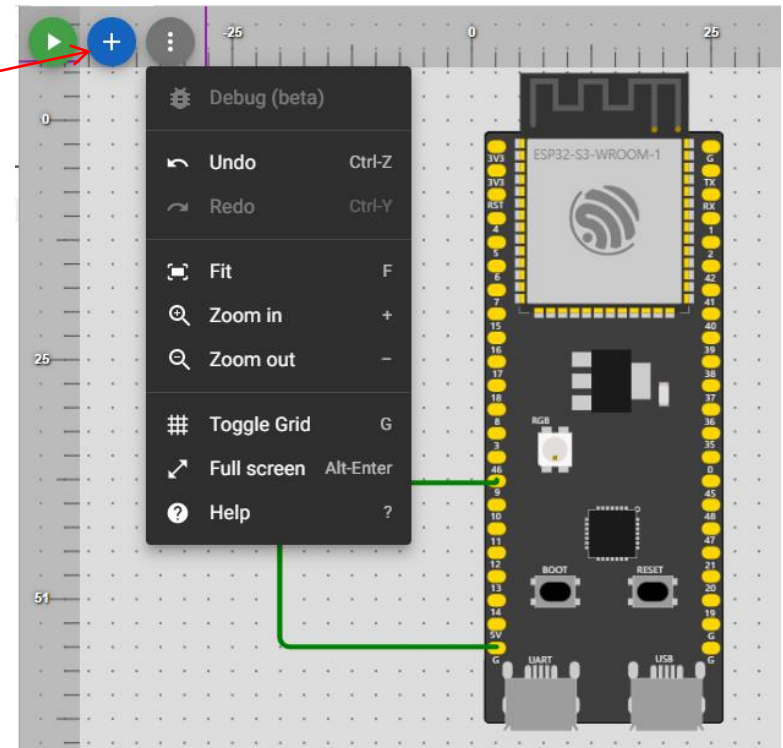
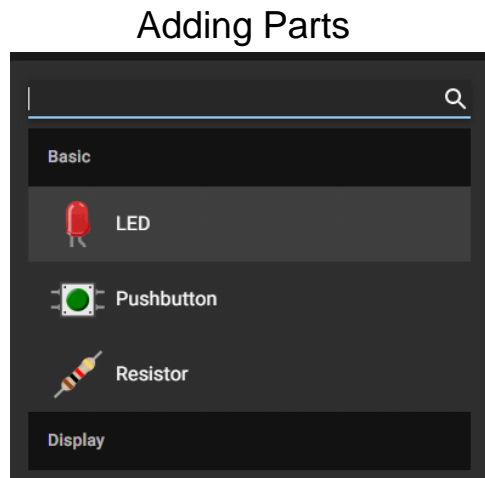
ESP32-S3 DevKitC-1 호환

시리얼 콘솔 출력

# Wokwi Diagram Editor

## ◆ Interactive Diagram Editor

- <https://docs.wokwi.com/guides/diagram-editor>



# How to Wokwi Diagram Editor

---

## ◆ Reference

- <https://docs.wokwi.com/guides/diagram-editor>

## ◆ Editing parts

- Adding a part
- Moving a part
- Rotating a part
- Duplicating a part
- Deliging a part
- Selecting multiple parts
- Copying and pasting parts

## ◆ Editing wires

- Creating wire between two parts
- Changing the color of a wire
- Deleting a wire

## ◆ Keyboard shortcuts

## ◆ Grid snapping



# Arduino Serial.print (1)

---

- ◆ 시리얼(Serial) 포트로 다양한 형식의 데이터를 ASCII 문자로 출력
  - Serial 포트 출력은 Arduino IDE의 시리얼 모니터 출력으로 확인 가능
  - 단, 시뮬레이터 환경에서는 우측 시리얼 콘솔 모니터 확인 가능

- ◆ 함수 Serial.print()

```
size_t Serial.print(val);  
size_t Serial.print(val, format);
```

```
size_t Serial.println(val);  
size_t Serial.println(val, format);
```

- val : 출력할 값
- format : 출력 형식(BIN,OCT,DEC,HEX) 및 소수점 자릿수
- 리턴 값 : 출력한 바이트 수
- Serial.println은 줄바꿈 포함
- 참고 : Serial.print() 사용 전에 Serial.begin() 함수를 이용하여 설정 필수

# Arduino Serial.print (2)

## ◆ 함수 Serial.begin()

```
void Serial.begin(speed);  
void Serial.begin (speed, config);
```

- speed : long 형식의 bps (전송 속도)
- config : data, parity, stop 비트 설정
  - ❖ 예) SERIAL\_8N1

## ◆ 사용 예

```
void setup() {  
  Serial.begin(115200);
```

```
  Serial.println(56);           // "56"  
  Serial.println(56, BIN);      // "111000"  
  Serial.println(56, OCT);      // "70"  
  Serial.println(56, DEC);      // "56"  
  Serial.println(56, HEX);      // "38"  
  Serial.println(56.7860, 0);    // "57" *반올림되어 출력됨  
  Serial.println(56.7860, 4);    // "56.7860"  
}
```

```
void setup() {  
  Serial.begin(115200);  
  
  Serial.println(56);           // "56"  
  Serial.println(56.7865);      // "56.79" *반올림되어 출력됨  
  Serial.println("H");          // "H"  
  Serial.println("Hello!!");    // "Hello!!"  
}
```

# 목 차

---

가상머신 활용

가상머신 실습 환경 구축

◆ **ESP32 제어**

WLAN 네트워크

# ESP32 하드웨어 패키지 지원 내용

Peripheral	ESP32	ESP32-S2	ESP32-C3	ESP32-S3
ADC	Yes	Yes	Yes	Yes
Bluetooth	Yes	Not Supported	Not Supported	Not Supported
BLE	Yes	Not Supported	Yes	Yes
DAC	Yes	Yes	Not Supported	Not Supported
Ethernet	Yes	Not Supported	Not Supported	Not Supported
GPIO	Yes	Yes	Yes	Yes
Hall Sensor	Yes	Not Supported	Not Supported	Not Supported
I2C	Yes	Yes	Yes	Yes
I2S	Yes	Yes	Yes	Yes
LEDC	Yes	Yes	Yes	Yes
Motor PWM	No	Not Supported	Not Supported	Not Supported
Pulse Counter	No	No	No	No
RMT	Yes	Yes	Yes	Yes
SDIO	No	No	No	No
SDMMC	Yes	Not Supported	Not Supported	Yes
Timer	Yes	Yes	Yes	Yes
Temp. Sensor	Not Supported	Yes	Yes	Yes
Touch	Yes	Yes	Not Supported	Yes
TWAI	No	No	No	No
UART	Yes	Yes	Yes	Yes
USB	Not Supported	Yes	Yes	Yes
Wi-Fi	Yes	Yes	Yes	Yes

# ESP32-S3 Pins for Arduino

```
#ifndef Pins_Arduino_h
#define Pins_Arduino_h

#include <stdint.h>
#include "soc/soc_caps.h"

#define USB_VID 0x303a
#define USB_PID 0x1001

#define EXTERNAL_NUM_INTERRUPTS          46
#define NUM_DIGITAL_PINS                 48
#define NUM_ANALOG_INPUTS                20

// Some boards have too low voltage on this pin (board design bug)
// Use different pin with 3V and connect with 48
// and change this setup for the chosen pin (for example 38)
static const uint8_t LED_BUILTIN = SOC_GPIO_PIN_COUNT+48;
#define BUILTIN_LED              LED_BUILTIN // backward compatibility
#define LED_BUILTIN              LED_BUILTIN
#define RGB_BUILTIN              LED_BUILTIN
#define RGB_BRIGHTNESS          64

#define analogInputToDigitalPin(p)      (((p)<20)?(analogChannelToDigitalPin(p)):-1)
#define digitalPinToInterrupt(p)        (((p)<48)?(p):-1)
#define digitalPinHasPWM(p)             (p < 46)
```

```
static const uint8_t TX = 43;
static const uint8_t RX = 44;

static const uint8_t SDA = 8;
static const uint8_t SCL = 9;

static const uint8_t SS  = 10;
static const uint8_t MOSI = 11;
static const uint8_t MISO = 13;
static const uint8_t SCK = 12;

static const uint8_t A0 = 1;
static const uint8_t A1 = 2;
static const uint8_t A2 = 3;
static const uint8_t A3 = 4;
static const uint8_t A4 = 5;
static const uint8_t A5 = 6;
static const uint8_t A6 = 7;
static const uint8_t A7 = 8;
static const uint8_t A8 = 9;
static const uint8_t A9 = 10;
static const uint8_t A10 = 11;
static const uint8_t A11 = 12;
static const uint8_t A12 = 13;
```

```
static const uint8_t A13 = 14;
static const uint8_t A15 = 16;
static const uint8_t A16 = 17;
static const uint8_t A17 = 18;
static const uint8_t A18 = 19;
static const uint8_t A19 = 20;

static const uint8_t T1 = 1;
static const uint8_t T2 = 2;
static const uint8_t T3 = 3;
static const uint8_t T4 = 4;
static const uint8_t T5 = 5;
static const uint8_t T6 = 6;
static const uint8_t T7 = 7;
static const uint8_t T8 = 8;
static const uint8_t T9 = 9;
static const uint8_t T10 = 10;
static const uint8_t T11 = 11;
static const uint8_t T12 = 12;
static const uint8_t T13 = 13;
static const uint8_t T14 = 14;

#endif /* Pins_Arduino_h */
```

# [실습1] ESP32-S3 Pins for Arduino (1)

```
void setup() {
  // put your setup code here, to run once:
  delay(500);
  Serial.begin(115200);
  delay(500);
  Serial.println("\n\n=====");
  Serial.printf("Chip Model: %s %s %d\n",
    ESP.getChipModel(),
    "rev.",
    (int)ESP.getChipRevision());
  Serial.printf("with number of cores = %d\n", (int)ESP.getChipCores());
  Serial.println("=====");

#ifdef EXTERNAL_NUM_INTERRUPTS
  Serial.printf("EXTERNAL_NUM_INTERRUPTS = %d\n",
    EXTERNAL_NUM_INTERRUPTS);
#endif

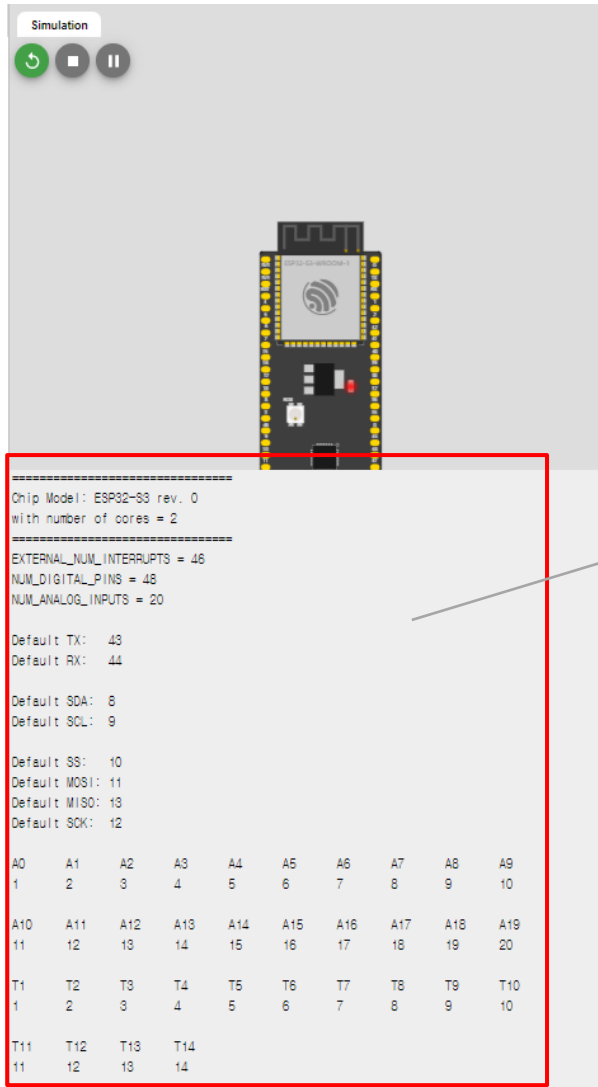
#ifdef NUM_DIGITAL_PINS
  Serial.printf("NUM_DIGITAL_PINS = %d\n", NUM_DIGITAL_PINS);
#endif

#ifdef NUM_ANALOG_INPUTS
  Serial.printf("NUM_ANALOG_INPUTS = %d\n", NUM_ANALOG_INPUTS);
#endif

  Serial.println();
  Serial.printf("Default TX: %d\n", TX);
  Serial.printf("Default RX: %d\n", RX);
```

```
Serial.println();
Serial.printf("Default SDA: %d\n", SDA);
Serial.printf("Default SCL: %d\n", SCL);
Serial.println();
Serial.printf("Default SS: %d\n", SS);
Serial.printf("Default MOSI: %d\n", MOSI);
Serial.printf("Default MISO: %d\n", MISO);
Serial.printf("Default SCK: %d\n", SCK);
Serial.println();
Serial.printf("A0\tA1\tA2\tA3\tA4\tA5\tA6\tA7\tA8\tA9\n");
Serial.printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
  A0, A1, A2, A3, A4, A5, A6, A7, A8, A9);
Serial.println();
Serial.printf("A10\tA11\tA12\tA13\tA14\tA15\tA16\tA17\tA18\tA19\n");
Serial.printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
  A10, A11, A12, A13, A14, A15, A16, A17, A18, A19);
Serial.println();
Serial.printf("T1\tT2\tT3\tT4\tT5\tT6\tT7\tT8\tT9\tT10\n");
Serial.printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
  T1, T2, T3, T4, T5, T6, T7, T8, T9, T10);
Serial.println();
Serial.printf("T11\tT12\tT13\tT14\n");
Serial.printf("%d\t%d\t%d\t%d\n",
  T11, T12, T13, T14);
Serial.println("=====");
}
```

# [실습1] ESP32-S3 Pins for Arduino (2)



## ESP32-S3 핀 정보 표출

```
=====
Chip Model: ESP32-S3 rev. 0
with number of cores = 2
=====
EXTERNAL_NUM_INTERRUPTS = 46
NUM_DIGITAL_PINS = 48
NUM_ANALOG_INPUTS = 20

Default TX: 43
Default RX: 44

Default SDA: 8
Default SCL: 9

Default SS: 10
Default MOSI: 11
Default MISO: 13
Default SCK: 12

A0  A1  A2  A3  A4  A5  A6  A7  A8  A9
1   2   3   4   5   6   7   8   9   10

A10 A11 A12 A13 A14 A15 A16 A17 A18 A19
11  12  13  14  15  16  17  18  19  20

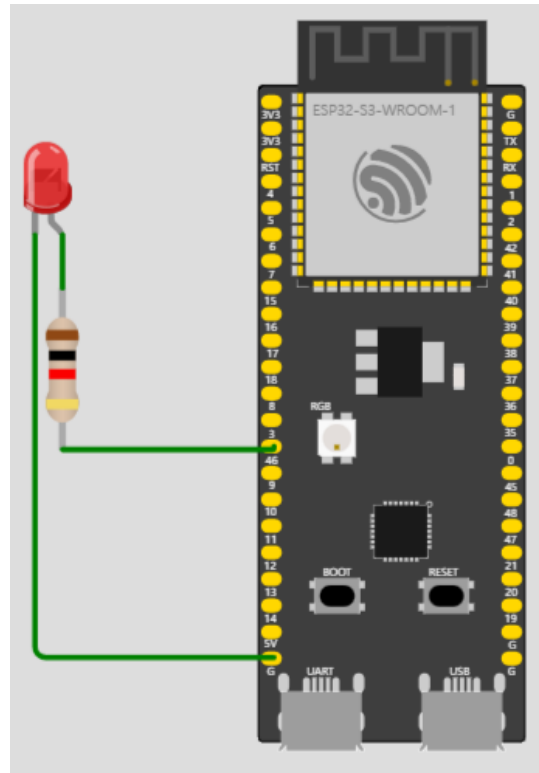
T1  T2  T3  T4  T5  T6  T7  T8  T9  T10
1   2   3   4   5   6   7   8   9   10

T11 T12 T13 T14
11  12  13  14
=====
```

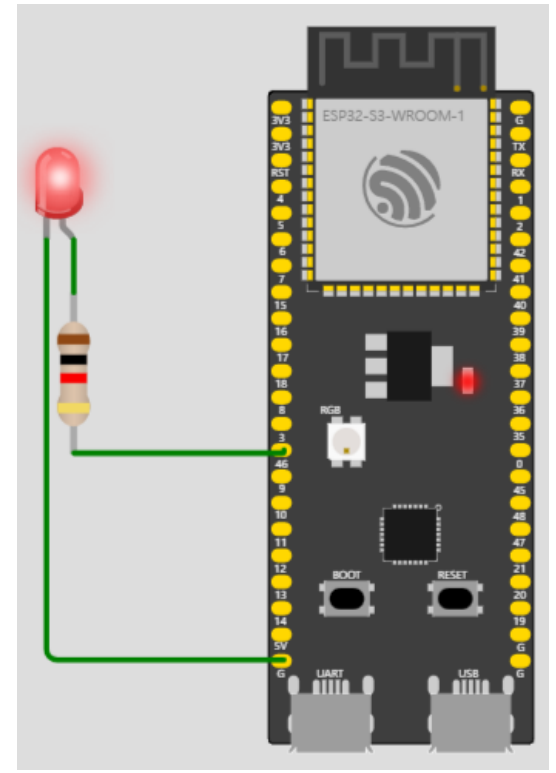
# [실습2] LED 구동

## ◆ LED 회로 및 인터페이스

- ESP32-S3-DevKitC-1 회로도 및 ESP32-S3 WROOM 모듈 사양서로 확인
- RGB LED on GPIO46



▲ LED OFF



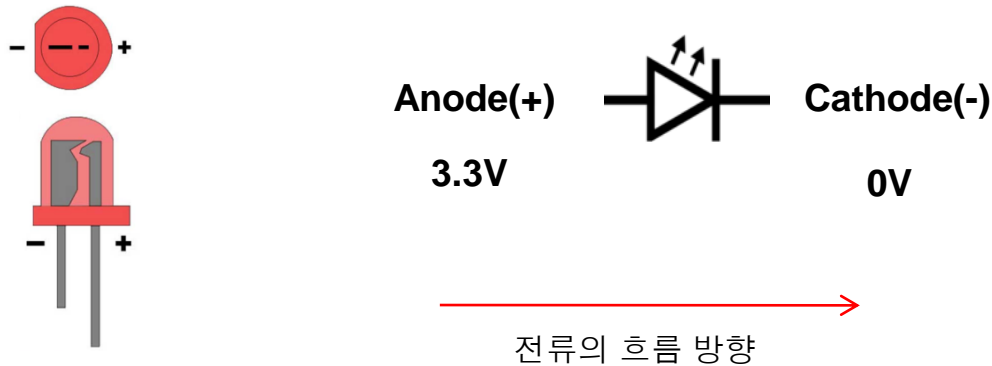
▲ LED ON



# [실습2] LED 구동 – LED

## ◆ LED (Light Emitting Diode)

- 전기 신호가 인가되면 빛을 내는 다이오드
- 다이오드 : 전류를 한쪽 방향으로만 흐르도록 제어, 역방향 전류 흐름 방지



- ESP32에서 GPIO를 이용하여 0V 또는 3.3V 구동 가능

# [실습2] LED 구동 – GPIO 동작 이해

## ◆ GPIO (General Purpose Input Output)

- 범용으로 사용하는 디지털 입출력 신호선 : ESP32-S3에 48개의 GPIO 지원
- 0 또는 1을 입력하여 외부 출력 신호를 제어하거나 외부에서 입력되는 신호에 따라 0 또는 1값을 확인

## ◆ GPIO 사용

- GPIO 사용을 하기 위해서 칩셋 내부에 각 pin을 제어하는 레지스터 사용
  - ❖ 입출력 모드 변경 register, 입출력 데이터 값 Read/Write하는 register
- 출력(Output) : Output 모드로 설정 후 0 또는 1을 레지스터에 Write
- 입력(Input) : Input 모드로 설정 후 해당 pin의 값을 읽는다

## ◆ Arduino에서의 GPIO 사용

- GPIO 사용을 하기 위한 API 사용

구분	함수 이름	설명
Digital	pinMode(pin, mode)	입력된 핀번호에 해당하는 핀의 동작(Input, Output)을 설정 한다.
	digitalWrite(pin, value)	입력된 핀번호에 해당하는 핀으로 디지털값(High, Low)값을 출력 한다.
	digitalRead(pin)	입력된 핀번호로부터 디지털 값(High, Low)을 읽는다.

0V	3.3V
Logic 0	Logic 1
Off	On
Low	High
Clear	Set
False	True
Open	Close

<https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/gpio.html>

# [실습2] LED 구동 – Arduino GPIO API

---

## ◆ pinMode

```
void pinMode(uint8_t pin, uint8_t mode);
```

- pin : GPIO 핀번호 지정
- mode : GPIO 동작 모드 설정
  - ❖ INPUT : 입력 모드로 선언, High Impedance
  - ❖ OUTPUT : 출력 모드로 선언
  - ❖ INPUT\_PULLDOWN : 입력모드로 선언, 내부적으로 pulldown (45K  $\Omega$ )
  - ❖ INPUT\_PULLUP : 입력 모드로 선언, 내부적으로 pullup (45K  $\Omega$ )

## ◆ digitalWrite

```
int digitalWrite(uint8_t pin, uint8_t val);
```

- pin : GPIO 핀번호 지정
- val : 디지털 상태 값 설정, HIGH or LOW

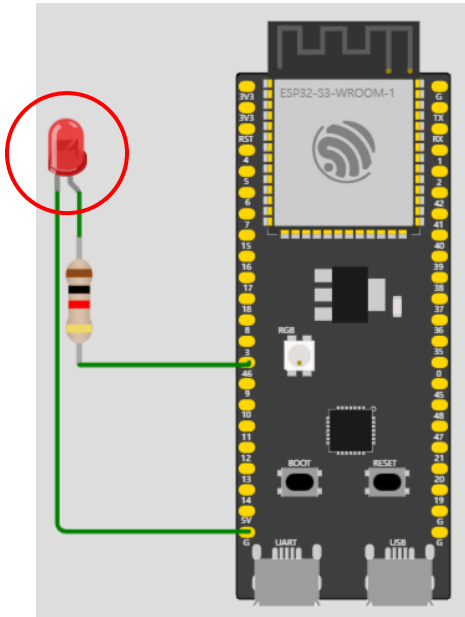
## ◆ digitalRead

```
int digitalRead(uint8_t pin);
```

- pin : GPIO 핀번호 지정
- Return 값 : 디지털 상태 값 리턴, HIGH or LOW

# [실습2] LED 구동 - 동작 구현

- ◆ 소스를 구현하고 컴파일 및 업로드 한 후 사진의 LED가 깜빡이면 성공



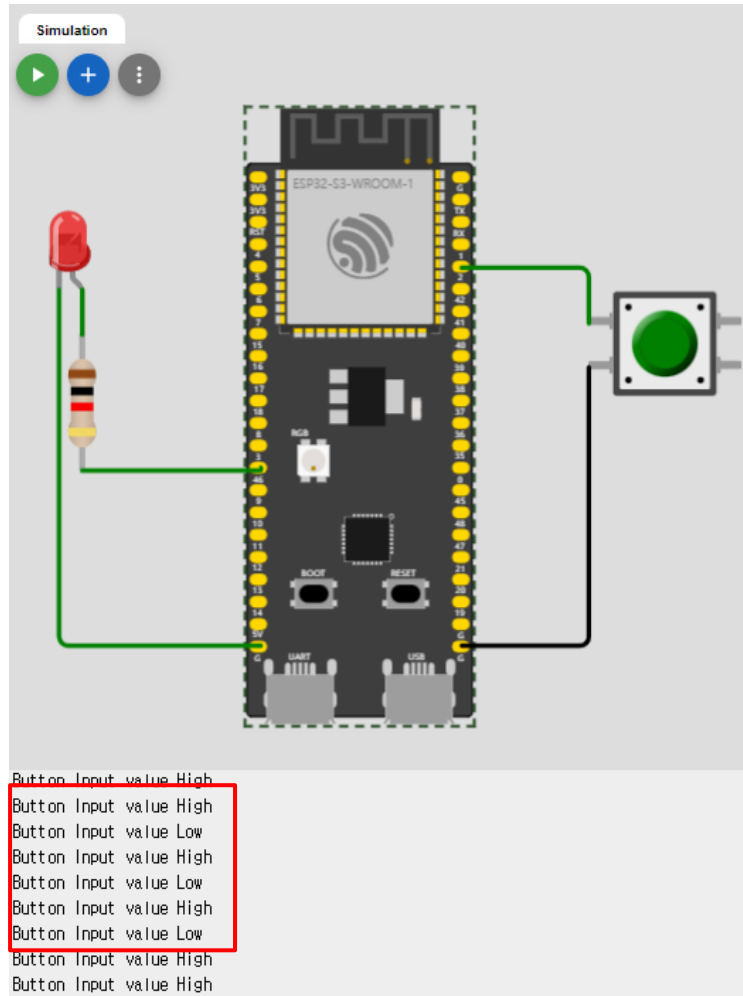
```
// 할당된 GPIO에 해당하는 Blink LED장치 Define 매크로 함수 정의
#define LED [Your code here]

void setup() {
    // 프로그램이 시작할 때 한번만 실행되는 함수, 초기설정
    // 입출력장치 핀설정 및 입출력 모드 설정
    // LED 제어를 하기위해 PIN의 모드를 출력(OUTPUT) 모드로 설정
    [Your code here]
}

void loop() {
    // 프로그램이 반복적으로 실행되는 함수 : 메인코드
    // 1. LED 값을 HIGH로 구동 (LED OFF)
    // 2. ms 단위 딜레이 : 예) delay(500)
    // 3. LED 값을 LOW로 구동 (LED ON)
    // 4. ms 단위 딜레이 : 예) delay(500)
    [Your code here]
}
```

# [실습3] BUTTON 입력

## ◆ BUTTON 입력 상태를 모니터에 프린트



```
#define buttonPin 5
```

```
int buttonState = 0;
```

```
void setup() {
```

```
    // 시리얼 포트 설정
```

```
    // BUTTON으로 사용할 PIN 모드 설정
```

```
    // BUTTON 핀 번호 출력
```

```
    [Your code here]
```

```
}
```

```
void loop() {
```

```
    // BUTTON 핀의 값을 출력
```

```
    // BUTTON 입력 값 출력
```

```
    // 만약(if) BUTTON 상태가 HIGH 이면
```

```
    // "Button input value High" 출력
```

```
    // 아니면 (else)
```

```
    // "Button input value Low" 출력
```

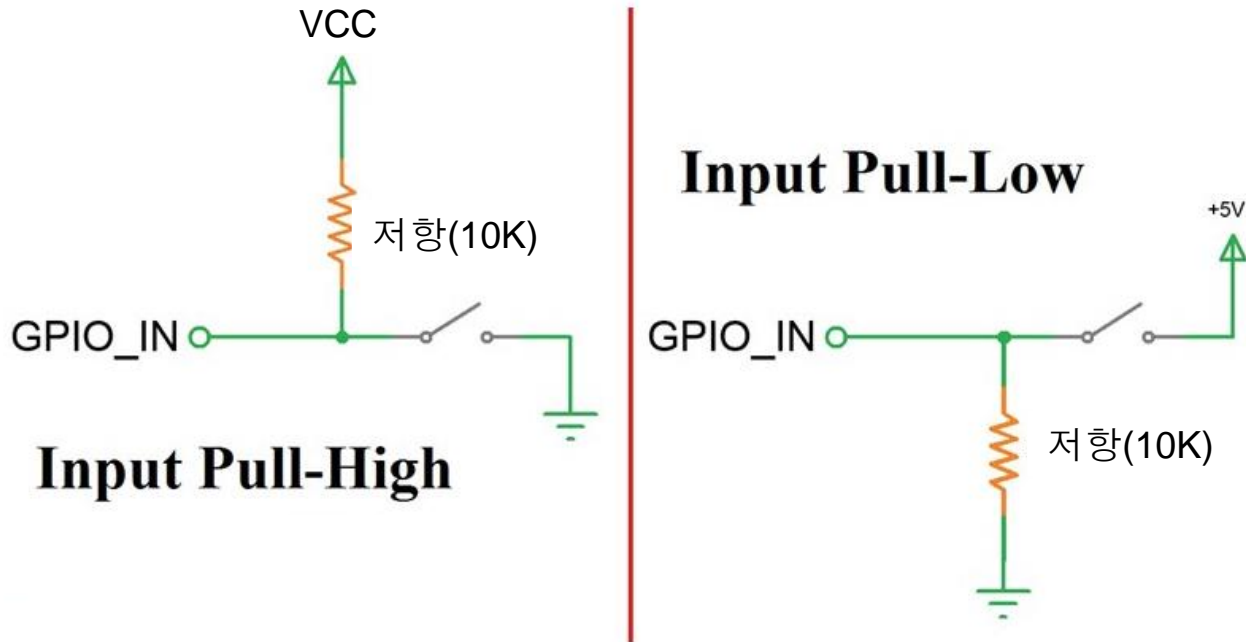
```
    // 1초 딜레이
```

```
    [Your code here]
```

```
}
```

## [실습3] BUTTON 입력 – 동작

---



# [실습4] BUTTON 입력+LED 제어

- ◆ BUTTON 입력 되면 LED ON, BUTTON 입력 없으면 OFF

```
#define LED LED_BUILTIN
#define buttonPin 5
int buttonState = 0;

void setup() {
    // 시리얼 포트 설정
    // LED 제어를 위한 PIN 모드 설정
    // BUTTON으로 사용할 PIN 모드 설정
    // BUTTON 핀 번호 출력
    [Your code here]
}
```

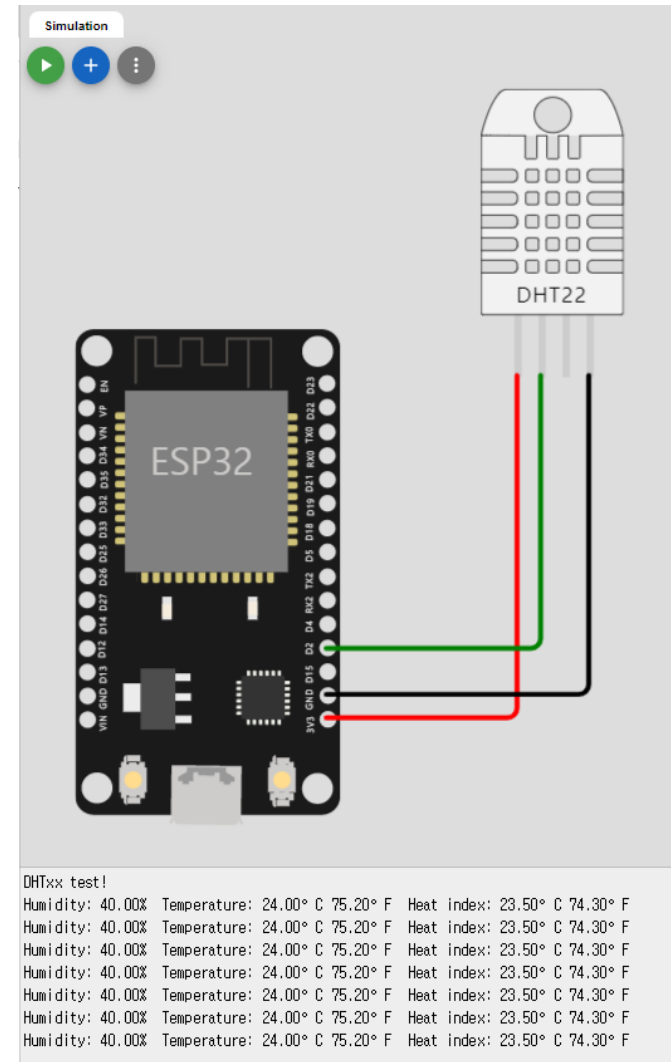
```
void loop() {
    // BUTTON 핀의 값을 출력
    // BUTTON 입력 값 출력
    // 만약(if) BUTTON 상태가 HIGH 이면
    // LED 값을 LOW로 구동 (LED ON)
    // 아니면 (else)
    // LED 값을 HIGH로 구동 (LED OFF)
    // 1초 딜레이
    [Your code here]
}
```

# [Project 1] 온습도 센서 읽기

- ◆ DHT22를 이용한 온도 및 습도 읽어서 콘솔 출력
  - 에디터를 이용하여 하드웨어 구성
  - DHT sensor library, DHT22 library 추가
  - ESP32 아두이노 소스 작성

## ※ 주의

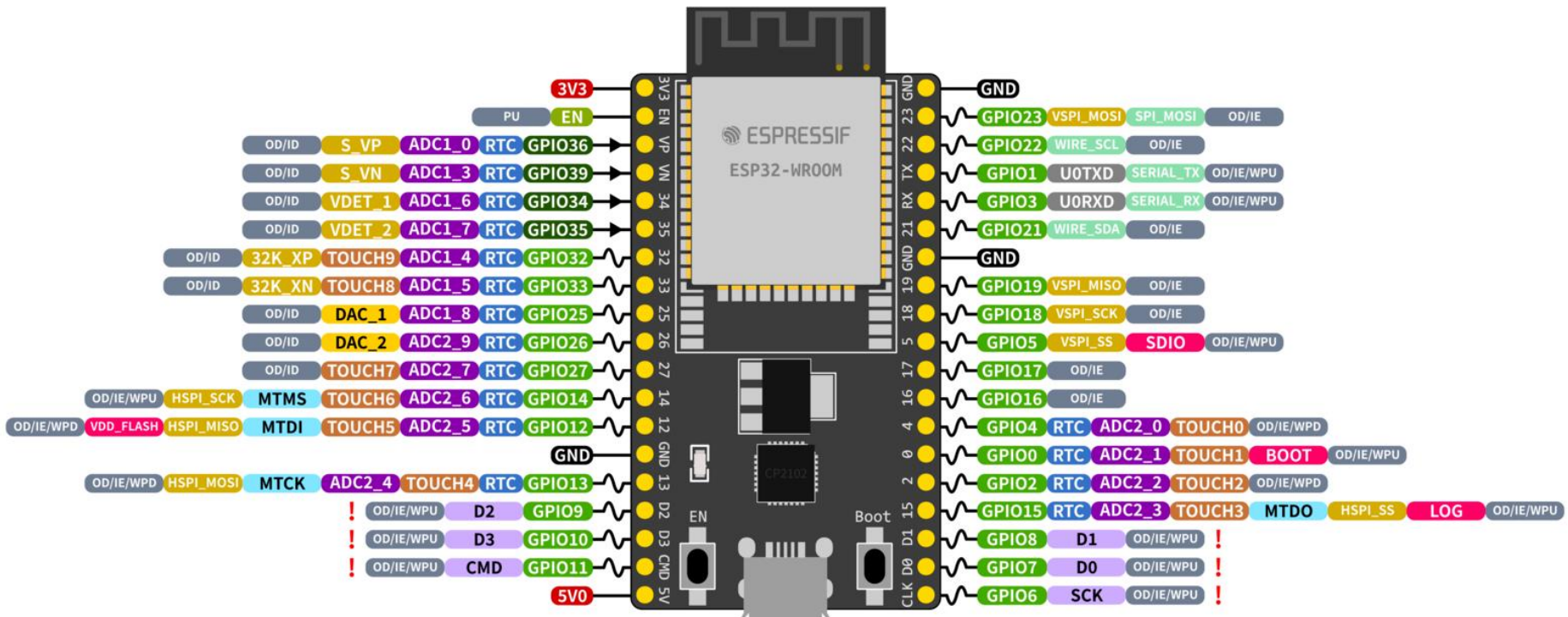
ESP32-S3는 Beta 버전으로 지원되지 않는 기능 많음  
Simulator 환경에서는 ESP32 사용하여 실습





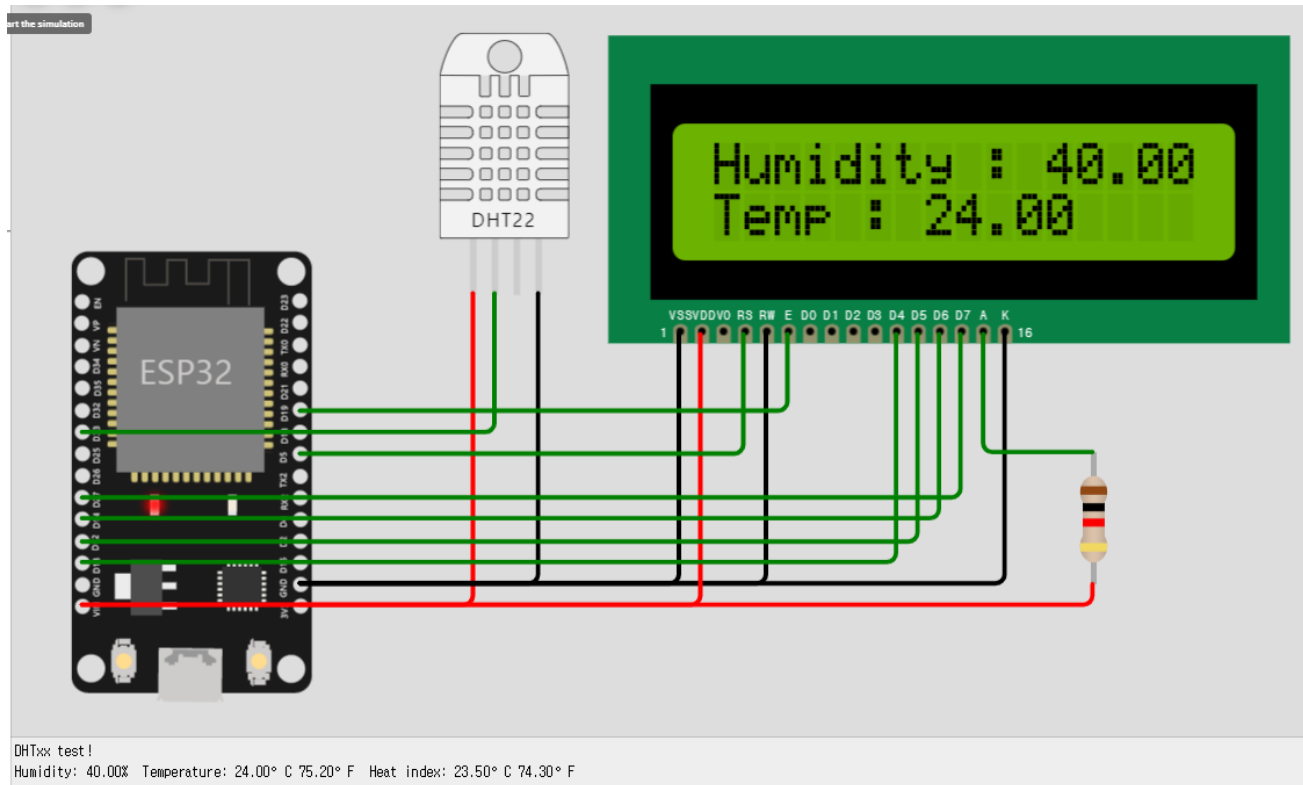
# ESP32 DevKit

- ◆ <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html>
- ◆ <https://docs.wokwi.com/guides/esp32>



# [Project 2] 온습도 Char LCD 출력

- ◆ DHT22를 이용한 온도 및 습도 읽어서 Char LCD 출력
  - 에디터를 이용하여 하드웨어 구성
  - DHT sensor library, DHT22 library 추가
  - ESP32 아두이노 소스 작성



# 아두이노 Character LCD

---

## ◆ API Reference

- <https://www.arduino.cc/reference/en/libraries/liquidcrystal/>
- <https://github.com/arduino-libraries/LiquidCrystal/blob/master/docs/api.md>

```
LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
```

```
#include <LiquidCrystal.h>
```

### Functions

- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()
- createChar()

# [Project 3] Ultrasonic Distance Sensor

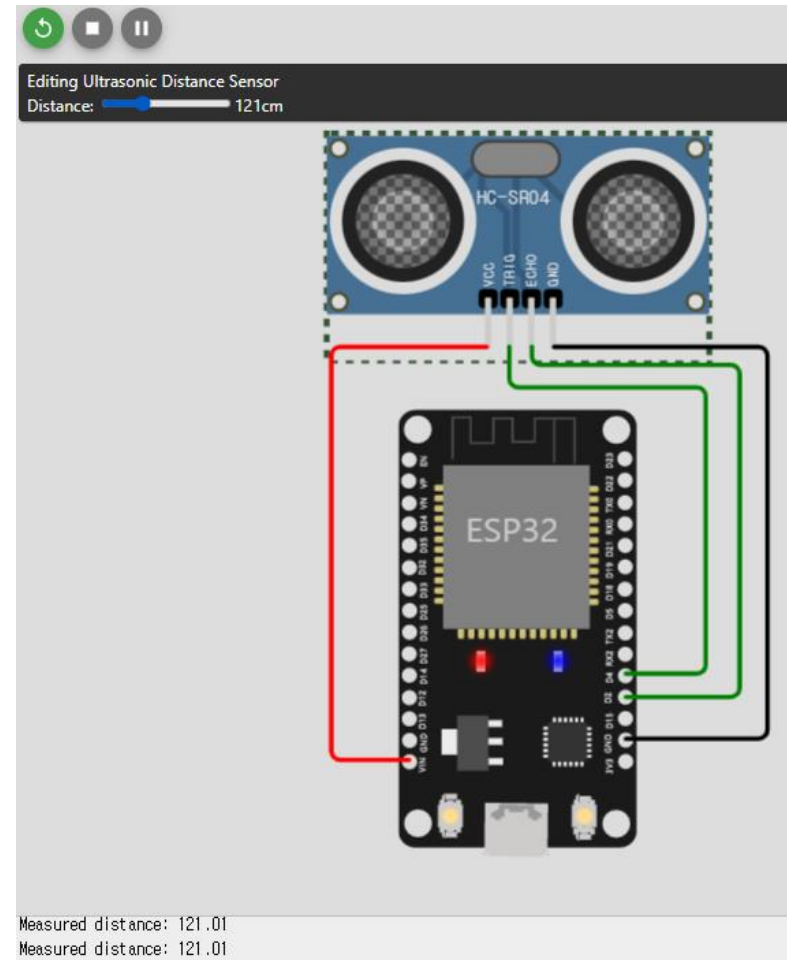
## ◆ HC-SR04 Ultrasonic Distance Sensor 활용 거리 측정

- 거리를 측정하여 시리얼로 측정된 거리 출력

### Operation

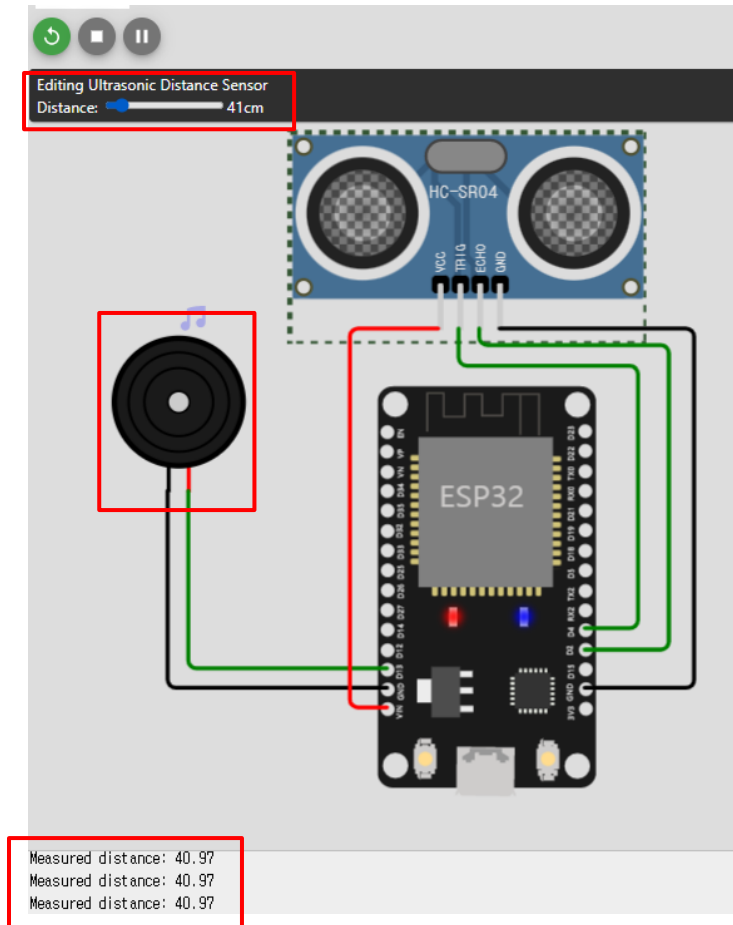
To start a new distance measurement set the TRIG pin to high for 10uS or more. Then wait until the ECHO pin goes high, and count the time it stays high (pulse length). The length of the ECHO high pulse is proportional to the distance. Use the following table to convert the ECHO pulse length in microseconds into centimeters / inches:

Unit	Distance
Centimeters	$\text{PulseMicros} / 58$
Inches	$\text{PulseMicros} / 148$

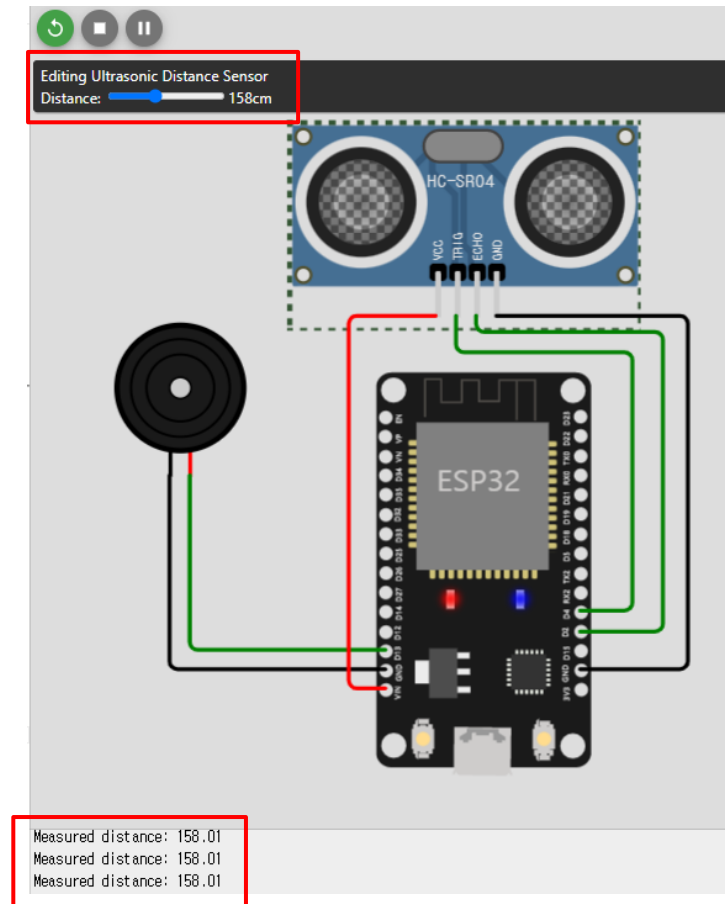


# [Project 4] 거리에 따른 Buzzer 구동

- ◆ 100cm 이하에서 Buzzer 구동 :



Buzzer On at Distance < 100



Buzzer Off at Distance > 100

# Buzzers

## ◆ Piezoelectric(압전식) Buzzer

- 피에조 효과로 소리를 내는 능동 부저(Active Buzzer) 및 수동 부저(Passive Buzzer)로 구분
- 압전기(壓電氣)(Piezoelectricity, /pi, eɪzəʊ, ɪlɛkˈtrɪsɪti/)란 기계적 일그러짐을 가함으로써 유전 분극을 일으키는 현상을 말한다. (Wikipedia)

## ◆ 능동 부저

- 내장된 회로를 통해 외부에서 전류가 흐르면 소리가 나는 부저

## ◆ 수동 부저

- 별도 내장된 회로 없이 미세한 진동으로 소리를 내는 장치



▲ 능동부저(Active Buzzer)와 수동부저(Passive Buzzer)

# Buzzer 동작 원리

## ◆ 능동 부저

- 동작 전압(ex 3.3~5V)을 인가하면 약 2KHz대역의 단일음 소리가 발생
- 경보음 또는 알림 용도로 사용

## ◆ 수동 부저

- 주파수를 활용하여 소리를 내기 때문에 음계 별 표준 주파수로 다양한 음역대의 소리 가능

( 단위 : Hz )

음계 \ 옥타브	1	2	3	4	5	6	7	8
C(도)	32.7032	65.4064	130.8128	261.6256	523.2511	1046.502	2093.005	4186.009
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D(레)	36.7081	73.4162	146.8324	293.6648	587.3295	1174.659	2349.318	4698.636
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E(미)	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F(파)	43.6535	87.3071	174.6141	349.2282	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.978	2959.955	5919.911
G(솔)	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A(라)	55.0000	110.0000	220.0000	440.0000	880.0000	1760.000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7458.620
B(시)	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

# ESP PWM을 활용한 Buzzer 동작 code

---

## ◆ PWM 변수 설정

- `int freq; // PWM 주파수`
- `int channel; // PWM 채널 설정: 0--15`
- `int resolution; // 해상도 설정: 8-bit (0--255)`

## ◆ PWM 초기화 in `setup()`

- `ledcSetup(channel, freq, resolution)`

## ◆ 핀-PWM 채널 연결 in `setup()`

- `ledcAttachPin(gpio_pin, channel)`

## ◆ Buzzer tone 설정

- `ledcWriteTone(channel, target_freq) // frequency 설정`
- `ledcWrite(channel, duty_cycle) // duty cycle 설정`



# 목 차

---

가상머신 활용

가상머신 실습 환경 구축

ESP32 제어

◆ **WLAN 네트워크**

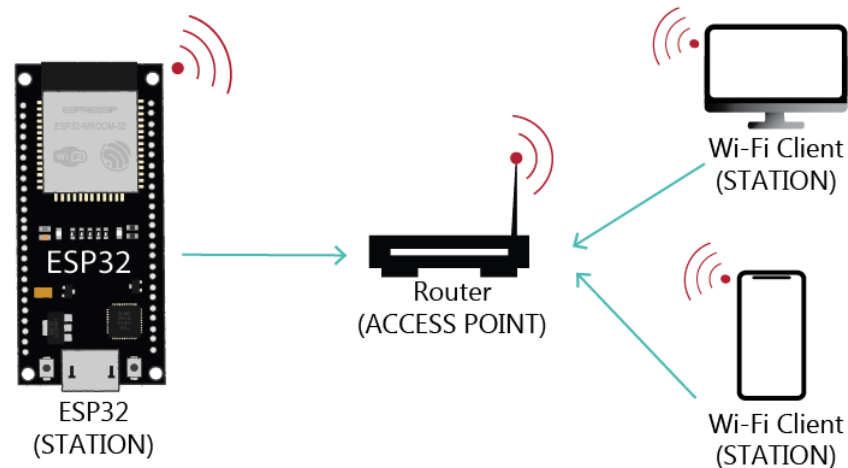
# Arduino WiFi 사용

## ◆ ESP32 WiFi 사용 API Reference

- <https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/>

## ◆ Station Mode Operation

```
WiFi.mode(WIFI_STA);
```



# Arduino WiFi 사용 API

## ▼ 아두이노 Wi-Fi 클래스 정보

Class	함수	Parameters	설명
WiFi	WiFi.begin()	ssid, keyIndex, key	Wi-Fi 라이브러리의 네트워크 설정 초기화
	WiFi.mode()	mode	Wi-Fi 모드 설정 (WIFI_STA, WIFI_AP, WIFI_STA_AP)
	WiFi.disconnect()	none	네트워크 연결 끊기
	WiFi.config()	ip, dns, gateway	DNS, 게이트웨이 및 서브넷 주소를 변경하고 정적 IP구성
	WiFi.SSID()	wifiAccessPoint	네트워크 SSID정보
	WiFi.RSSI()	wifiAccessPoint	네트워크 연결 신호 강도
	WiFi.encryptionType()	wifiAccessPoint	네트워크의 암호화 유형 정의
	WiFi.scanNetworks()	none	주변의 사용 가능한 네트워크를 검색하고 정보를 반환
	WiFi.getSocket()	none	사용가능한 소켓을 받는다.
	WiFi.macAddress()	mac	Wi-Fi장치의 MAC주소값
	WiFi.status()	none	현재 연결 상태 반환
IP	WiFi.localIP()	none	IP주소값
	WiFi.subnetMask()	none	서브넷 마스크의 정보
	WiFi.gatewayIP()	none	게이트웨이 IP주소 정보
Server	WiFiServer server(port)	port(int)	서버 포트 생성
	server.begin()	none	서버연결
	server.available()	none	서버 연결 정보 반환
	server.write()	data(byte)	서버에 연결된 클라이언트 제이터 쓰기
Client	WiFiClient client;	none	클라이언트 연결
	client.connected()	none	클라이언트 연결 정보 반환
	client.connect()	ip, URL, Port	클라이언트 연결
	client.write()	data(byte)	연결된 클라이언트 데이터 쓰기
	client.available()	none	읽을 수 있는 바이트수 반환
	client.read()	none	클라이언트 데이터 읽기
	client.stop()	none	서버 연결 끊기

# [실습5] WiFi 연결 하기 (1)

- ◆ WiFi 연결 후 시리얼 콘솔에 IP 주소 출력



## [실습5] WiFi 연결하기 (2)

---

```
#include <WiFi.h>

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

  Serial.print("Connecting to WiFi");

  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }

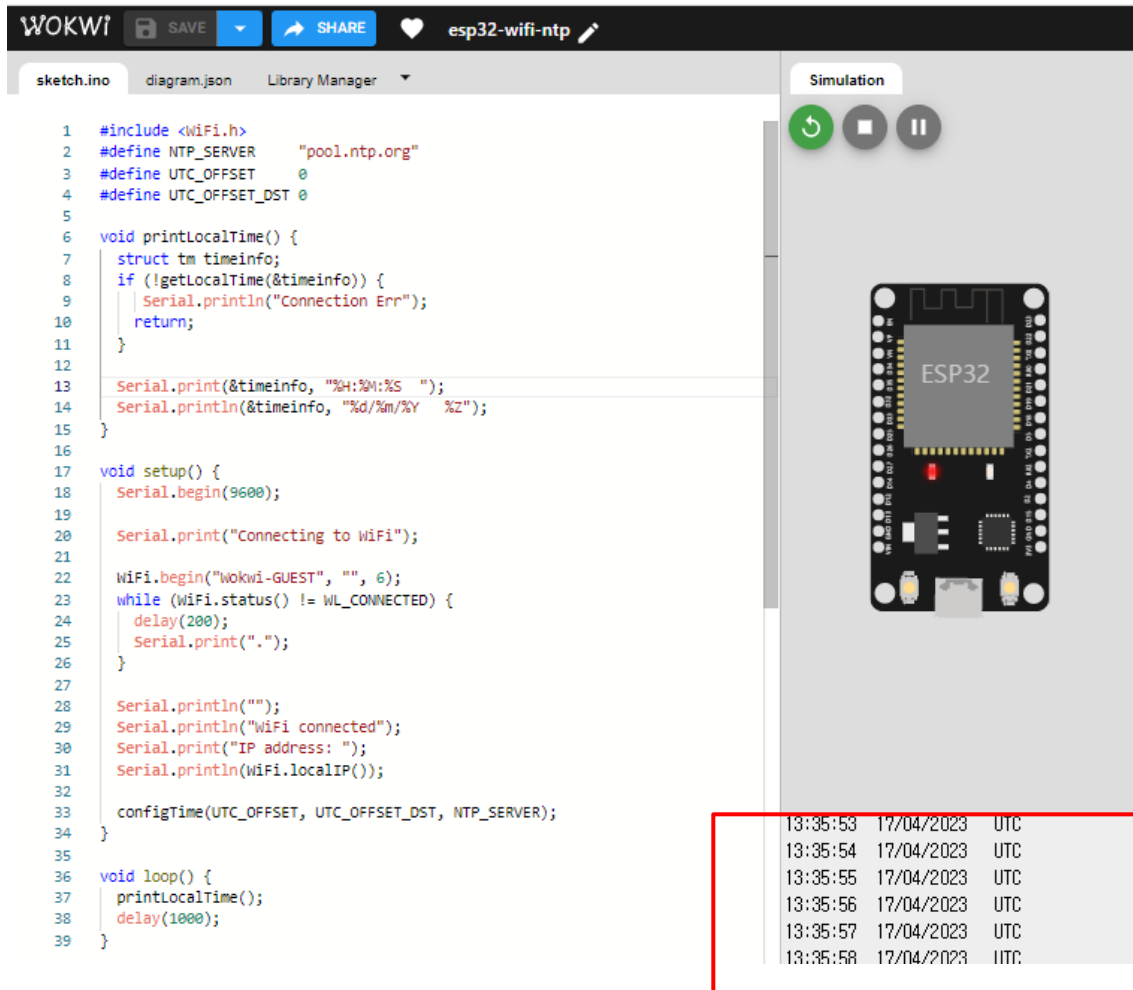
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {

  // Wait a few seconds between measurements.
  delay(2000);
}
```

# [실습6] NTP 서버에서 현재시간 가져오기 (1)

- ◆ NTP 서버에서 현재날짜와 시간을 읽어서 시리얼 콘솔에 출력



The screenshot shows the Wokwi IDE interface. The top bar includes 'WOKWI', 'SAVE', 'SHARE', and the project name 'esp32-wifi-ntp'. Below the bar are tabs for 'sketch.ino', 'diagram.json', and 'Library Manager'. The main editor area contains the following C++ code:

```
1 #include <WiFi.h>
2 #define NTP_SERVER "pool.ntp.org"
3 #define UTC_OFFSET 0
4 #define UTC_OFFSET_DST 0
5
6 void printLocalTime() {
7     struct tm timeinfo;
8     if (!getLocalTime(&timeinfo)) {
9         Serial.println("Connection Err");
10        return;
11    }
12
13    Serial.print(&timeinfo, "%H:%M:%S ");
14    Serial.println(&timeinfo, "%d/%m/%Y %Z");
15 }
16
17 void setup() {
18     Serial.begin(9600);
19
20     Serial.print("Connecting to WiFi");
21
22     WiFi.begin("wokwi-GUEST", "", 6);
23     while (WiFi.status() != WL_CONNECTED) {
24         delay(200);
25         Serial.print(".");
26     }
27
28     Serial.println("");
29     Serial.println("WiFi connected");
30     Serial.print("IP address: ");
31     Serial.println(WiFi.localIP());
32
33     configTime(UTC_OFFSET, UTC_OFFSET_DST, NTP_SERVER);
34 }
35
36 void loop() {
37     printLocalTime();
38     delay(1000);
39 }
```

On the right, the 'Simulation' tab shows a 3D model of an ESP32 board. Below the board is a serial monitor window with a red border, displaying the following output:

13:35:53	17/04/2023	UTC
13:35:54	17/04/2023	UTC
13:35:55	17/04/2023	UTC
13:35:56	17/04/2023	UTC
13:35:57	17/04/2023	UTC
13:35:58	17/04/2023	UTC

# [실습6] NTP 서버에서 현재시간 가져오기 (2)

```
#include <WiFi.h>
#define NTP_SERVER    "pool.ntp.org"
#define UTC_OFFSET    0
#define UTC_OFFSET_DST 0

void printLocalTime() {
    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) {
        Serial.println("Connection Err");
        return;
    }

    Serial.print(&timeinfo, "%H:%M:%S ");
    Serial.println(&timeinfo, "%d/%m/%Y %Z");
}
```

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

    Serial.print("Connecting to WiFi");

    WiFi.begin("Wokwi-GUEST", "", 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(200);
        Serial.print(".");
    }

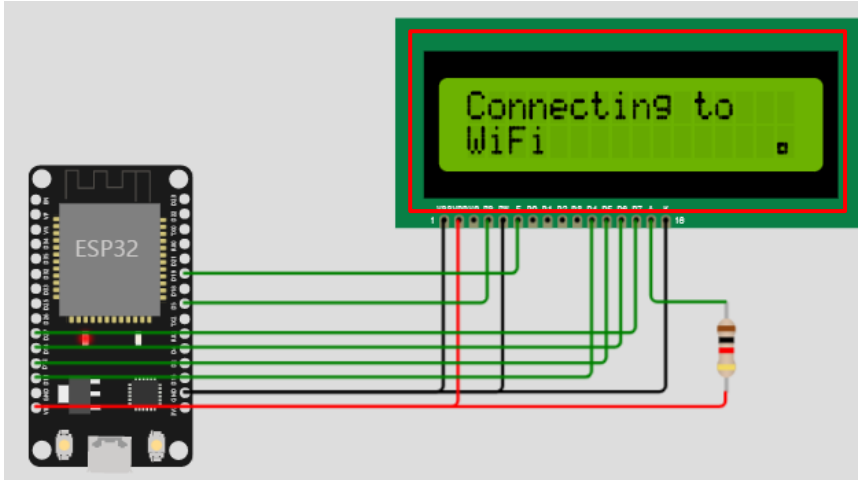
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());

    configTime(UTC_OFFSET, UTC_OFFSET_DST, NTP_SERVER);
}

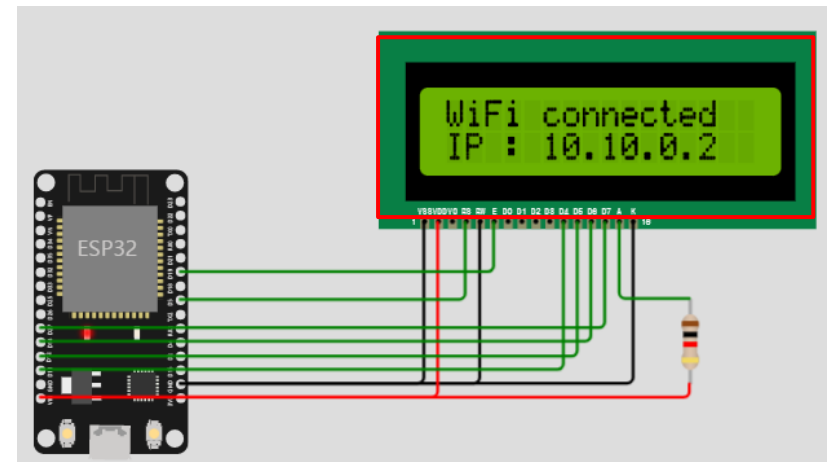
void loop() {
    printLocalTime();
    delay(1000);
}
```

# [Project 5] WiFi 연결하기, LCD 출력

- ◆ WLAN 연결 후 LCD에 IP 주소 출력



WLAN 연결 중

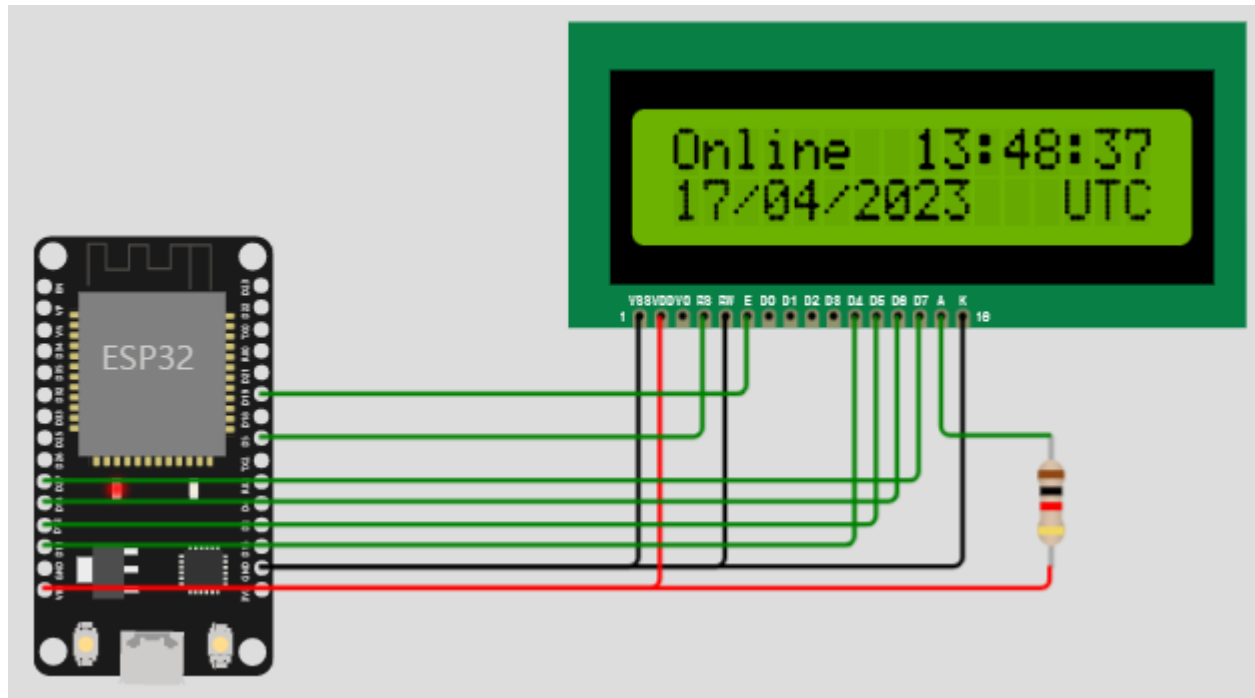


WLAN 연결 완료



# [Project 6] 현재 날짜와 시간 LCD 출력

- ◆ WLAN 연결 후 NTP 서버에서 날짜와 시간을 읽어서 아래와 같이 출력



---

# 질의 응답