

ESP32 DAY1

Simple Digital Output/Input

The use of tangible, embedded microcontroller-based interactive systems that can sense the world around them and/or control outputs such as lights, displays and motors.

<https://www.microsoft.com/en-us/research/project/physical-computing/>

Physical Computing

Introduction to Arduino & ESP32



- 2005
- Arduino 是一個基於簡單易用的韌體與軟體開源開放原始碼電子平台
- 始於義大利的Ivrea 互動設計研究所，是設計給沒有電子電路與程式設計背景的學生使用，讓設計師與藝術家能用非常低的時間成本，快速製作出閃爍的LED、馬達、感測器...等藝術作品或是產品原型。
- 優點:
 1. 開放原始碼硬體及軟體
 2. 價格便宜
 3. 支援很多感測器元件或模組
 4. 網路上有許多函式庫程式庫



<https://makeradvisor.com/esp32-development-boards-review-comparison/>

什麼是ESP32?

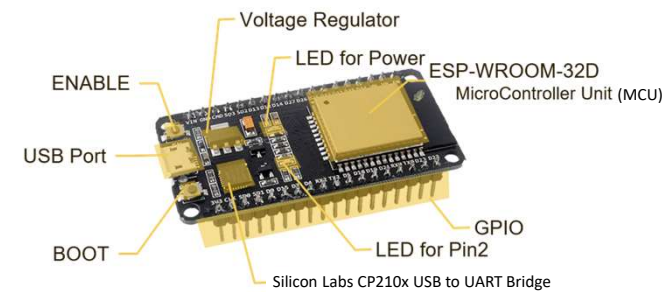
1. 它是一顆 MCU (Micro-Controller Unit)
2. 由上海樂鑫科技開發 (Espressif)
3. 支援Arduino開放性架構
4. 前身是ESP8266系列
5. 內建WiFi及低功耗藍芽BLE，可用腳位也擴增到約26根，大部分腳位同時支援類比及數位，內建霍爾、溫度、觸控感測器，主頻高達240MHz的雙核心 CPU，傳輸界面SPI、I2C、UART等強大的功能。



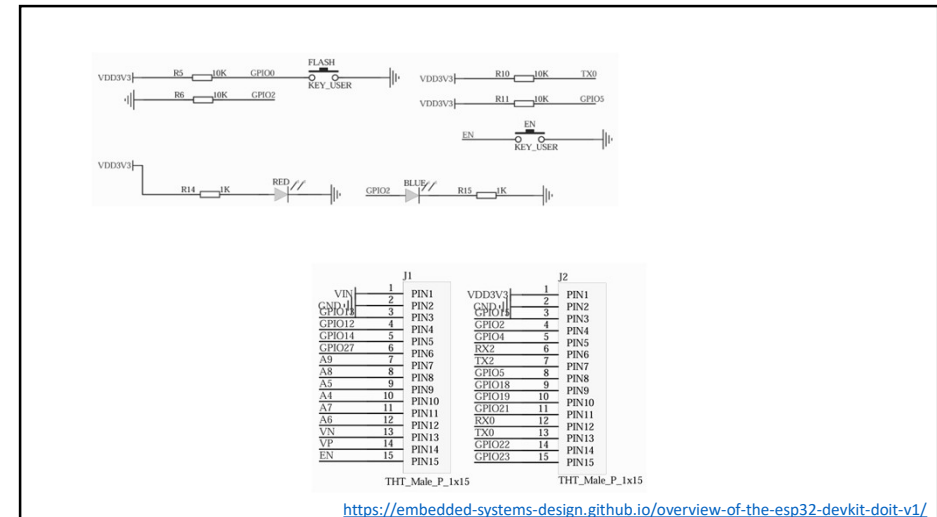
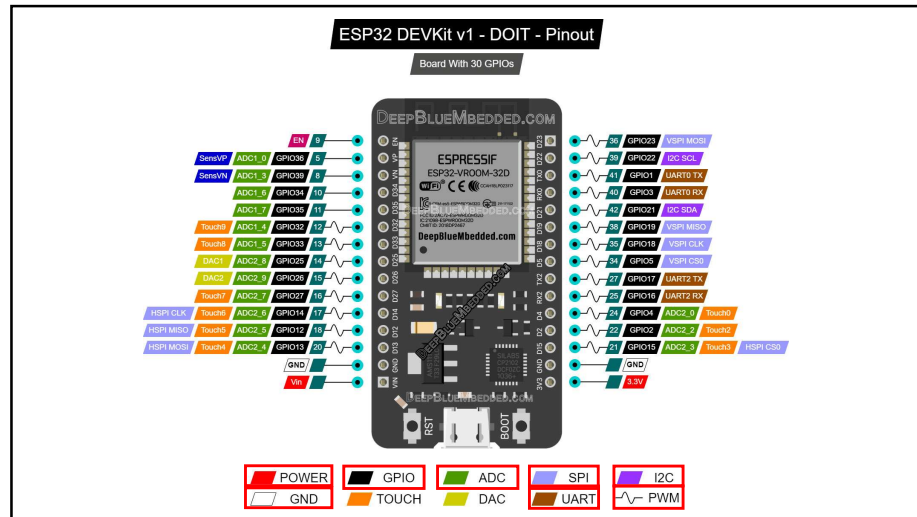
<https://ithelp.ithome.com.tw/articles/10269937?sc=IThomeR>

三者比較表			
	Arduino UNO	ESP8266系列	ESP32系列
CPU	AVR ATmega328P	Tensilica Xtensa LX106	Tensilica Xtensa LX106
核心	單核 20 MHz	單核 80/160 MHz	雙核 160/240 MHz
資料寬度	8 位元	32 位元	32 位元
SRAM	16 KB	160 KB	512 KB
Flash 空間	32 KB	1.4 MB	4-32 MB
GPIO	13	8	18
ADC(類比)	8	1	18
PWM	6	8	16
類比解析度	0~1023	0~1023	0~4095
I2C 組數	1	1	2
SPI 組數	1	1	3
I2S 組數	1	1	2
UART 組數	1	1	3
OTA 更新	x	支援	支援
內建 WiFi	x	802.11 b/g/n	802.11 b/g/n
內建藍芽	x	x	BLE 4.2
內建觸控電容	x	x	10 組
內建溫度感測	x	x	1
內建霍爾感測	x	x	1

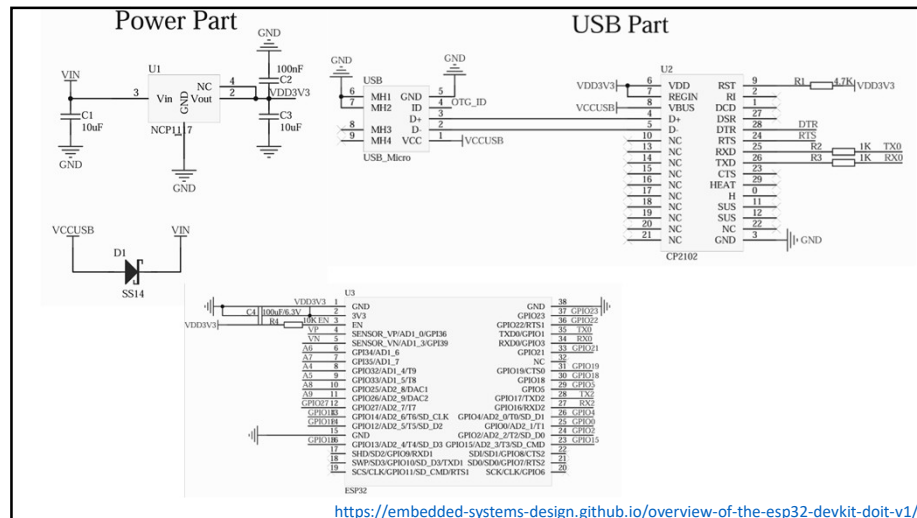
DOIT DEVKIT V1



<https://ithelp.ithome.com.tw/articles/10269937>



<https://embedded-systems-design.github.io/overview-of-the-esp32-devkit-doit-v1/>

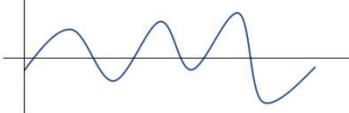


<https://embedded-systems-design.github.io/overview-of-the-esp32-devkit-doit-v1/>

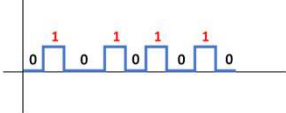
Simple Digital Output (DO)

AI: 何謂數位訊號? 何謂類比訊號?

類比訊號 (Analog)



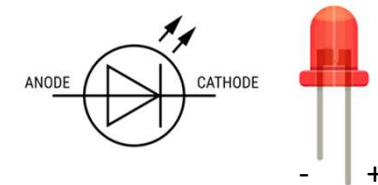
數位訊號 (Digital)



- <https://share.gemini.google/KonKXt3mNYZe>
- <https://copilot.microsoft.com/shares/YksE371ob5DEkPjcDKTmk>
- <https://claude.ai/share/97317fe3-d9ec-4a76-91ea-2616379d136d>
- <https://chatgpt.com/share/6a922949-92d8-83ee-bb7a-c9fbd1894b33>
- https://grok.com/share/bGVnYWN5_381ca543-5eea-4f9a-af12-be1057bbfab2
- <https://www.perplexity.ai/computer/tasks/433e0f20-9c5d-4187-983f-0d0e3098540f>

LED

- 發光二極體(Light Emitting Diode)
- 二極體是由p型與n型兩種不同類型的半導體組合而成的
- 二極體可發揮讓電流朝單一方向流動的整流作用(rectifying) Light-Emitting Diode (LED)
- 重點：
 - 長腳 → 正極
 - 短腳 → 負極
- 常見的紅色中LED



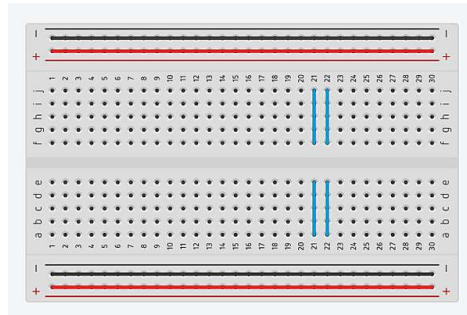
General Purpose Input/Output (GPIO)

- GPIO 是種具有彈性且可以藉由軟體控制(software-controlled) 的數位訊號
- 常見於開發版邊緣, 以針腳 (Pin) 的形式呈現
 - 這些針腳即是開發版與外界溝通的重要橋樑
 - 一可以想像是開關, user透過軟體可以自由打開關閉, 可以當LED輸出, 也可以當按鈕的輸入, 控制連續的High、Low變化產生PWM, 當USB、USART...等等通訊協定的變化
- 每個 GPIO 可以被當成 input, output, analog 或 alternative function
 - alternative function 是指其他的的功能, 如 I2C, SPI, USART, PWM, Clock 等。如何控制則取決於外部設備 (peripheral), 所以總結來說它是一個通用型的, 泛用型的輸入、輸出腳位

We often need current-limiting resistors for LEDs

LED Color	Typical Vf (@ 20 mA)	Operating Voltage Range	Primary Semiconductor Material
Infrared (IR)	1.2 V – 1.4 V	1.2 V – 1.6 V	GaAs / AlGaAs
Red	1.9 V – 2.0 V	1.8 V – 2.2 V	AlGaAs / GaAsP
Orange / Amber	2.0 V – 2.1 V	1.9 V – 2.3 V	GaAsP / AlInGaP
Yellow	2.1 V – 2.2 V	2.0 V – 2.4 V	AlInGaP / GaAsP
Yellow-Green (Standard)	2.1 V – 2.3 V	2.0 V – 2.5 V	GaP
Pure / Emerald Green	3.2 V	3.0 V – 3.6 V	InGaN
Blue	3.2 V	3.0 V – 3.6 V	InGaN
White	3.2 V	3.0 V – 3.6 V	InGaN (with phosphor coating)
Ultraviolet (UV)	3.4 V – 3.8 V	3.2 V – 4.2 V	InGaN / AlGaIn

麵包板



黑線這排互相連接

紅線這排互相連接

每5個點相互連接

分隔島

每5個點相互連接

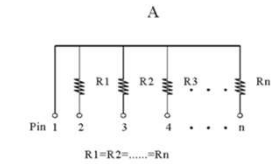
黑線這排互相連接

紅線這排互相連接

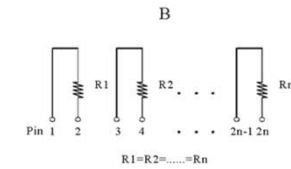
330Ω 10PA 排阻



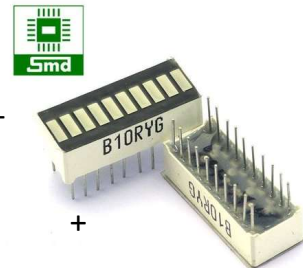
A-Type排阻接線圖



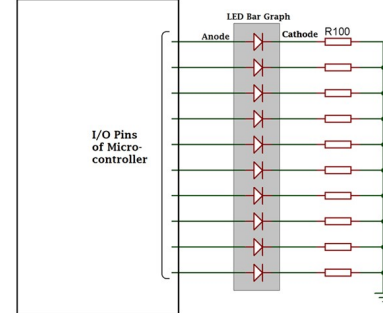
B-Type排阻接線圖



LED Bar



Micro-Controller



Why do we need these resistors?

- Protecting the LEDs from Burning Out
- Protecting the Microcontroller's I/O Pins

<https://components101.com/displays/led-bar-graph>

<https://share.gemini.google/QnOf7hXBemjz>

			LED Bar	排阻	
EN		23		Common	3V3
36 GPI		22		4	
39 GPI TX0		1		5	
34 GPI RX0		3	-	6	
35 GPI		21	-	7	
32		19	-	8	
33		18	-	8	
25		5	-	7	
26		17	-	6	
27		16	-	5	
14		4	-	4	
12		2	-	3	
13		15	-	2	
GND		GND		Common	3V3
Vin		3V3			

Functions you may want to use

• Digital IO functions

- `pinMode(pin, mode)`
pin: the Arduino pin number to set the mode of.
mode: `INPUT`, `OUTPUT`, `INPUT_PULLUP`, or `INPUT_PULLDOWN`.
- `digitalWrite(pin, value)`
pin: the Arduino pin number.
value: `HIGH` or `LOW`.
- `digitalRead(pin)`
pin: the Arduino pin number you want to read [return: `HIGH` or `LOW`]

• Time function

- `delay(ms)`

<https://docs.espressif.com/projects/arduino-esp32/en/latest/api/gpio.html>
<https://docs.arduino.cc/language-reference/en/functions/digital-io/pinMode/>
<https://docs.arduino.cc/language-reference/en/functions/digital-io/digitalwrite/>
<https://docs.arduino.cc/language-reference/en/functions/digital-io/digitalread/>

Simple Digital Input (DI)

Lab: a scanning LED

t0	
t1	
t2	
t3	
t4	
t5	
t6	
t7	
t8	
t9	
t10	
t11	
t12	
t13	
t14	
t15	
t16	
t17	
t18	

Cylon Robot

<https://www.youtube.com/watch?v=SGibEb4IH6k>



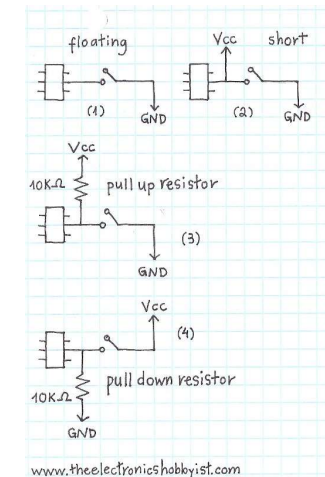
Knight Rider

https://knight-rider.fandom.com/wiki/Anamorphic_Equalizer
<https://en.wikipedia.org/wiki/KITT>



Digital Input (DI)

- GPIO pins can be used for output & input.
- In the diagrams on the RHS (Right Hand Side):
 - Do not use (1) or (2)
 - (3) reads HIGH when the switch is open, and reads LOW when the switch is closed. (active low switch)
 - (4) reads LOW when the switch is open, and reads HIGH when the switch is closed. (active high switch)
- In ESP32, a voltage above $\frac{3}{4} V_{cc}$ is considered is HIGH, and below $\frac{1}{4} V_{cc}$ is considered is LOW.
- In ESP32, GPIO input can be configured as `INPUT_PULLUP` or `INPUT_PULLDOWN`. The internal resistor is 45kOhm.



<https://vinduino.blogspot.com/2013/07/active-high-vs-active-low-button.html>

Homework. Use a switch to control the scanning

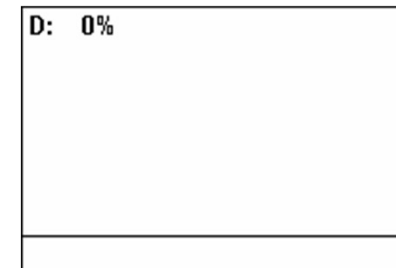
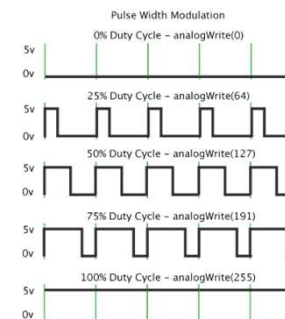
- Use switch to turn on/off LED_BUILTIN
- Hold the switch to pause scanning
 - When the switch is open, the LED will continue its scan
- Use the switch to switch modes
 - All on
 - All off
 - Scan

Advanced Digital Output / PWM

Using interrupts

<https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/>
<https://zebrakent.pixnet.net/blog/post/36306796>
<https://garrysblog.com/2021/03/20/reliably-debouncing-rotary-encoders-with-arduino-and-esp32/>
<https://www.az-delivery.de/en/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/debounced-rotary-encoder>

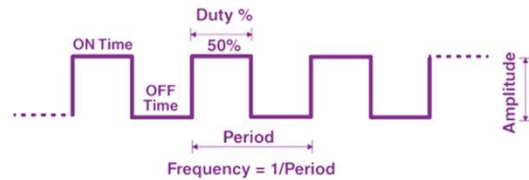
PWM: Pulse Width Modulation



<https://zh.wikipedia.org/zh-tw/%E5%8D%A0%E7%A9%BA%E6%AF%94>
<https://docs.arduino.cc/learn/microcontrollers/analog-output/>

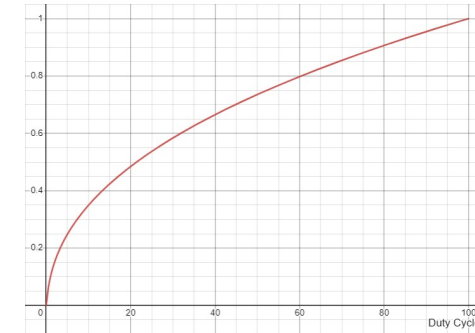
PWM parameters

- **PWM Frequency:** PWM Frequency is the count of PWM interval periods per second, expressed in Hertz (Hz). Mathematically, the frequency is equal to the inverse of the interval period's length ($\text{PWM_Frequency} = 1 / \text{PWM_Interval_Period}$).
- **Resolution:** number of possible cycles in one PWM_Interval_Period.
- **Duty cycle:** number of cycles that the PWM pin should be ON/HIGH in one PWM_Interval_Period.



<https://byjus.com/physics/pulse-width-modulation/>
<https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html>

Gamma Correction



<https://codeinsecurity.wordpress.com/2023/07/17/the-problem-with-driving-leds-with-pwm/>

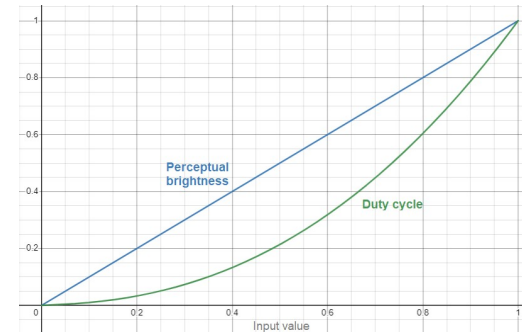
LEDC APIs

- `ledcAttach(pin, frequency, resolution);`
 - `ledcAttachChannel(pin, frequency, resolution, channel);`
 - `ledcDetach(pin);`
 - `ledcWrite(pin, duty_cycle);`
 - `ledcFade(pin, start_duty, target_duty, max_fade_time_ms);`
- channel: LEDC channel number [0-15]*
frequency: set the PWM frequency
resolution: set the signal's duty cycle resolution (in bits)
pin: GPIO pin number
duty_cycle: zero to 2^{(resolution)-1}
start_duty: start duty cycle for fading
target_duty: target duty cycle for fading
max_fade_time_ms: the maximum fading time to reach the target duty cycle.

<https://lastminuteengineers.com/esp32-pwm-tutorial/>
<https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html>

ESP32 SoC	Number of LEDC channels	resolutions
ESP32	16	1-20
ESP32-S2	8	1-14
ESP32-S3	8	1-14
ESP32-C3	6	1-14
ESP32-C6	6	1-14
ESP32-H2	6	1-14

Gamma Correction



<https://codeinsecurity.wordpress.com/2023/07/17/the-problem-with-driving-leds-with-pwm/>

$$(\text{duty cycle}) = (\text{perception})^{2.2}$$

Perception	Duty Cycle
0.05	0.14%
0.1	0.63%
0.15	1.54%
0.2	2.90%
0.25	4.74%
0.3	7.07%
0.35	9.93%
0.4	13.32%
0.45	17.26%
0.5	21.76%
0.55	26.84%
0.6	32.50%
0.65	38.76%
0.7	45.63%
0.75	53.10%
0.8	61.21%
0.85	69.94%
0.9	79.31%
0.95	89.33%
1	100.00%

Homework

1. Use switch to control your scanning
2. Use another switch to turn on all LEDs
3. Use 3rd switch to display 10 LEDs with different brightness.