

## Algemene informatie

ESP32 is een microcontroller met geïntegreerde wifi, gelanceerd door Espressif, die over sterke eigenschappen beschikt en integreert rijke randapparatuur. Het kan worden ontworpen en bestudeerd als een gewone Single-Chip Module (SCM), of verbonden met internet en gebruikt als een Internet of Things-apparaat.

ESP32 kan worden ontwikkeld met behulp van het Arduino-platform, wat het zeker gemakkelijker zal maken voor mensen die dat willen hebben Arduino geleerd om onder de knie te krijgen. Bovendien is de code van ESP32 volledig open source, dus beginners kan snel leren hoe u slimme huishoudelijke IoT producten kunt ontwikkelen en ontwerpen, waaronder slimme gordijnen, ventilatoren, lampen en klokken.

Over het algemeen bestaan ESP32 projecten uit code en schakelingen. Maak je geen zorgen, zelfs als je nog nooit code en schakelingen hebt geleerd. We zullen geleidelijk de basiskennis van programmeertaal C en elektronische schakelingen introduceren, van eenvoudig tot moeilijk. Onze producten bevatten alle elektronische componenten en modules die nodig zijn om deze projecten voltooien. Het is vooral geschikt voor beginners.

We verdelen elk project in vier delen, namelijk Componentenlijst, Componentkennis, Schakeling en Code. Met de Componentenlijst kunt u sneller materiaal voor het experiment voorbereiden. Componentkennis maakt dit mogelijk u om snel nieuwe elektronische modules of componenten te begrijpen, terwijl Schakeling u helpt het werkingsprincipe van de schakeling. En met Code kunt u eenvoudig het gebruik van ESP32 en de accessoireset onder de knie krijgen. Nadat u alle projecten in deze tutorial hebt afgerond, kunt u deze componenten en modules ook zelf maken producten zoals slimme huishoudens, slimme auto's en robots om uw creatieve ideeën om te zetten in prototypes en nieuwe en innovatieve producten.

Als u bovendien problemen of vragen heeft met deze tutorial of toolkit, vraag dan gerust naar onze snelle en gratis technische ondersteuning via [support@freenove.com](mailto:support@freenove.com)

## ESP32-WROVER

ESP32-WROVER heeft in totaal twee antennepakketten gelanceerd, respectievelijk PCB-on-board-antenne en IPEX-antenne. De ingebouwde PCB-antenne is een geïntegreerde antenne in de chipmodule zelf, dus gemakkelijk mee te nemen en te ontwerpen. De IPEX-antenne is een metalen antenne afgeleid van de geïntegreerde antenne van de chipmodule zelf, die wordt gebruikt om het signaal van de module te versterken.

PCB-on-board-antenne

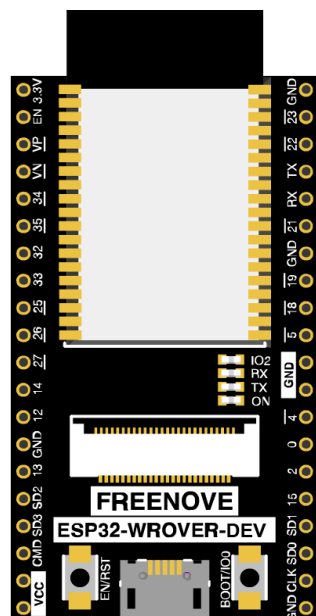


IPEX-antenne

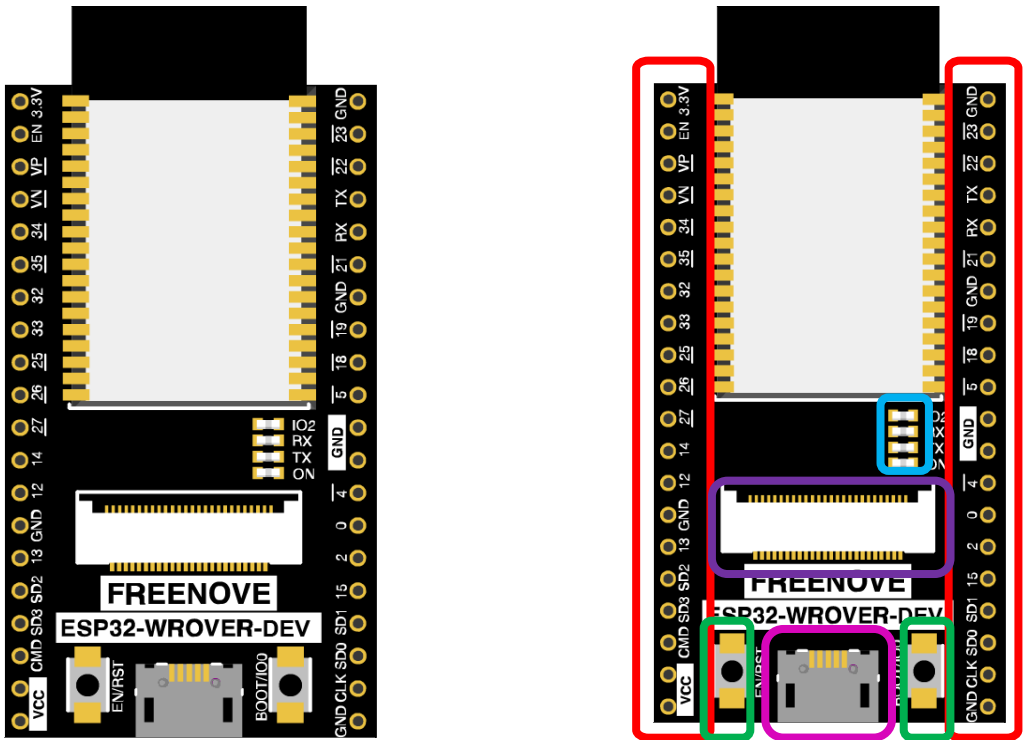


In deze zelfstudie wordt de ESP32-WROVER ontworpen op basis van de ESP32-WROVER-module met PCB-antenne aan boord.






ESP32-WROVER



De hardware-interfaces van ESP32-WROVER zijn als volgt verdeeld:



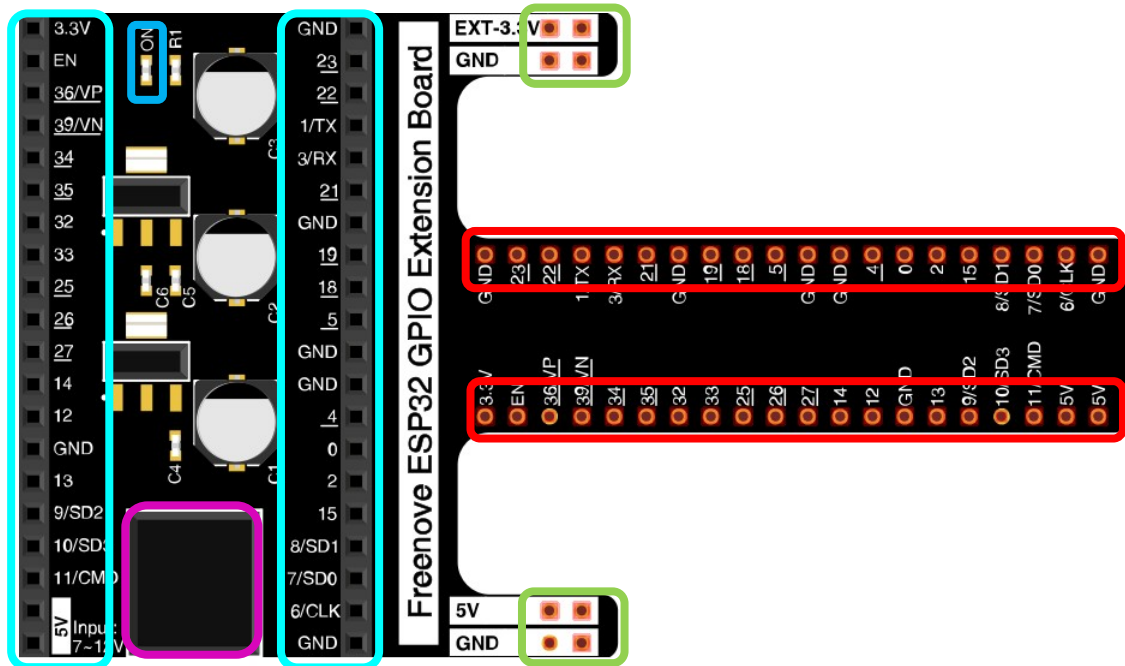
Vergelijk de linker- en rechterafbeeldingen. We hebben de bronnen op de ESP32-WROVER in verschillende kleuren verpakt om uw begrip van de ESP32-WROVER te vergemakkelijken.

Kleur	Overeenkomstige onderdelen
	GPIO pin
	LED-indicator
	Camera interface
	Resetknop, selectieknop voor opstartmodus
	USB poort




Voor meer informatie kunt u terecht op:  
[https://www.espressif.com.cn/sites/default/files/documentation/esp32-WROVER-1\\_wroom-1u\\_datasheet\\_en.pdf](https://www.espressif.com.cn/sites/default/files/documentation/esp32-WROVER-1_wroom-1u_datasheet_en.pdf)

## Uitbreidingskaart van de ESP32-WROVER

En we ontwerpen ook een uitbreidingskaart, zodat je de ESP32 gemakkelijker kunt gebruiken volgens het meegeleverde schakelschema. Hieronder volgen hun foto's. De hardware-interfaces van ESP32-WROVER zijn als volgt verdeeld:



We hebben de onderdelen op de ESP32-WROVER in verschillende kleuren verpakt om uw begrip van de ESP32-WROVER te vergemakkelijken.

Kleur	Overeenkomstige onderdelen
	GPIO pin
	LED-indicator
	GPIO-interface van ontwikkelbord
	Stroom geleverd door de uitbreidingskaart
	Externe voeding

- In ESP32 is GPIO een interface om randapparatuur te besturen.
- In de volgende projecten gebruiken we standaard alleen een USB-kabel om de ESP32-WROVER van stroom te voorzien.
- In de hele tutorial gebruiken we het uitbreidingsbord niet om ESP32-WROVER van stroom te voorzien. Dus 5V en 3,3V (incl EXT 3.3V) op de uitbreidingskaart worden geleverd door ESP32-WROVER.
- We kunnen ook de DC-aansluiting van het uitbreidingskaart gebruiken om de ESP32-WROVER van stroom te voorzien. Op deze manier is 5v en EXT 3.3v op de uitbreidingskaart worden geleverd door een externe voedingsbron.

## Opmerkingen voor GPIO

### Camera Pinnen

Wanneer u de camera van onze ESP32-WROVER gebruikt, controleer dan de pinnen ervan. Pins met onderstreepte cijfers worden gebruikt door de camerafunctie. Als u daarnaast andere functies wilt gebruiken, gebruik deze dan niet.

## Software op Ubuntu

### Installatie

Arduino IDE 2

## Hoofdstuk 0 LED

Dit hoofdstuk is het startpunt van de reis om elektronische ESP32-projecten te bouwen en te verkennen. We beginnen met een eenvoudig "Blink" -project.

### Project 0.1 Blink

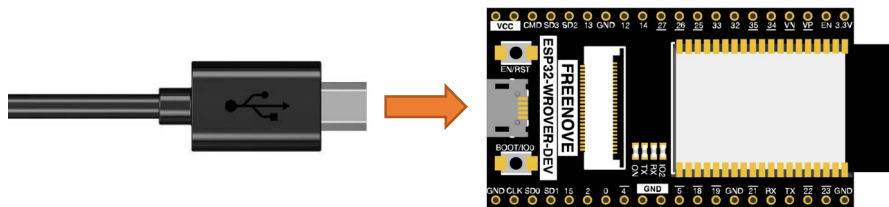
In dit project zullen we ESP32 gebruiken om het knipperen van een gemeenschappelijke LED te regelen.

#### Componentenlijst

ESP32-WROVER x1	USB kabel
-----------------	-----------

#### Stroom

ESP32-WROVER heeft een voeding van 5 V nodig. In deze tutorial moeten we ESP32-WROVER via een USB-kabel op de computer aansluiten om hem van stroom te voorzien en te programmeren. We kunnen ook een andere 5V-stroombron gebruiken om hem van stroom te voorzien.



In de volgende projecten gebruiken we standaard alleen een USB-kabel om ESP32-WROVER van stroom te voorzien.

## Schets

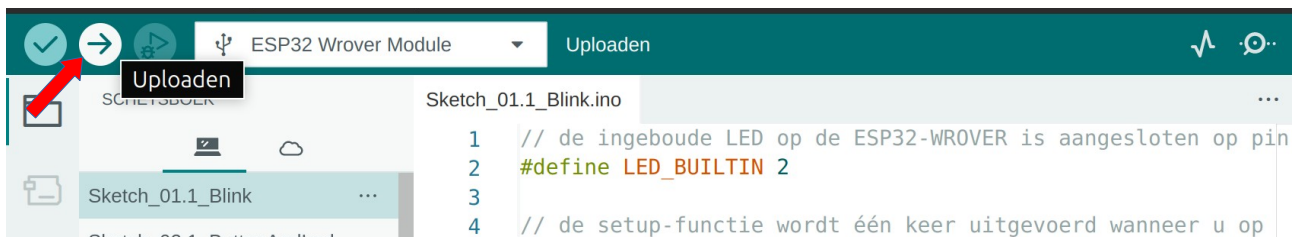
Volgens de schakeling gaat de LED AAN als pin GPIO2 van de ESP32-WROVER hoog is. Omgekeerd, wanneer de pin GPIO2 laag is, gaat de LED UIT. Daarom kunnen we GPIO2 doorlopend hoog en laag aansturen om de LED te laten knipperen.

Upload de volgende schets:

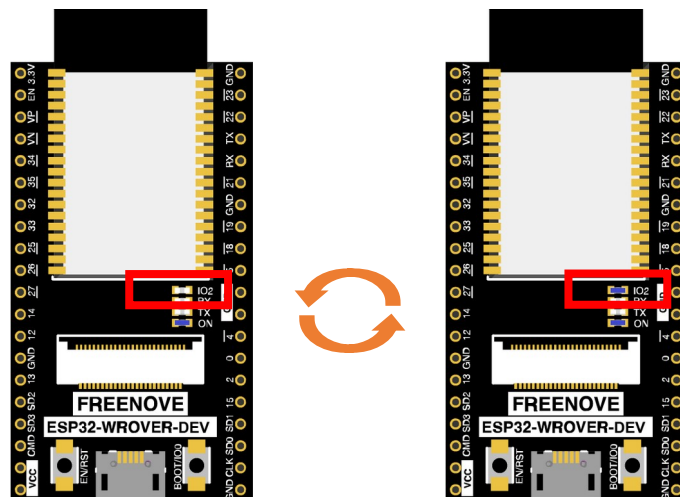
**Freenove\_Ultimate\_Starter\_Kit\_for\_ESP32/Sketches/Sketch\_01.1\_Blink.**

### Sketch\_01.1\_Blink

Open schets *Sketch\_01.1\_Blink* en klik *Uploaden*



De code wordt geüpload naar de ESP32-WROVER en de LED op het bord begint doorlopend te knipperen.



Dit is de programmacode:

```
1 // de ingebouwde LED op de ESP32-WROVER is aangesloten op pin 2
2 #define LED_BUILTIN 2
3
4 // de setup-functie wordt één keer uitgevoerd wanneer je op
5 // reset drukt of het bord van stroom voorziet
6 void setup() {
7   // initialiseer digitale pin LED_BUILTIN als uitvoer.
8   pinMode(LED_BUILTIN, OUTPUT);
9 }
10
11 // de loop-functie loopt voor altijd door. Op het einde begint ze gewoon terug opnieuw
12 void loop() {
13   digitalWrite(LED_BUILTIN, HIGH); // LED aan door het spanningsniveau HIGH (hoog) te maken
14   delay(1000); // wacht een seconde
15   digitalWrite(LED_BUILTIN, LOW); // LED uit door het spanningsniveau LOW (laag) te maken
16   delay(1000); // wacht een seconde
17 }
```



De Arduino IDE code bevat meestal twee basisfuncties: `void setup()` en `void loop()`. Nadat het bord is gereset, wordt eerst de functie `setup()` uitgevoerd en vervolgens de functie `loop()`. De functie `setup()` wordt doorgaans gebruikt om code te schrijven om de hardware te initialiseren. En de functie `loop()` wordt gebruikt om code te schrijven om bepaalde functies te bereiken. De `loop()` functie wordt herhaaldelijk uitgevoerd. Wanneer de uitvoering het einde van `loop()` bereikt, springt deze naar het begin van `loop()` om opnieuw te worden uitgevoerd.

## Reset

Een resetbewerking zorgt ervoor dat de code vanaf het begin wordt uitgevoerd. Als u de stroom inschakelt, het uploaden van de code voltooit en op de resetknop drukt, wordt de resetbewerking geactiveerd.

## Code

GPIO2 van ESP32-WROVER is verbonden met de ingebouwde LED, dus de LED-pin is gedefinieerd als 2.

```
2 #define LED_BUILTIN 2
```

Dit betekent dat na deze coderegel alle LED\_BUILTIN's als 2 worden behandeld.

In de `setup()` functie stellen we eerst de LED\_BUILTIN in als uitvoermodus, waardoor de poort een hoog of laag niveau kan uitvoeren.

```
7 // initialiseer digitale pin LED_BUILTIN als uitvoer.
8 pinMode(LED_BUILTIN, OUTPUT);
```

Stel vervolgens in de functie `loop()` de LED\_BUILTIN in op een hoog uitgangsniveau, zodat de LED gaat branden.

```
13 digitalWrite(LED_BUILTIN, HIGH); // zet de LED aan door het spanningsniveau HIGH te maken
```

Wacht 1000 ms, dat is 1 seconde. De functie `delay()` wordt gebruikt om de besturingskaart even te laten wachten voordat de volgende instructie wordt uitgevoerd. De parameter geeft het aantal milliseconden aan waarop moet worden gewacht.

```
14 delay(1000); // wacht een seconde
```

Stel vervolgens de LED\_BUILTIN in op een laag uitgangsniveau en het LED-lampje gaat uit. Eén seconde later wordt de uitvoering van de functie `loop()` voltooid.

```
15 digitalWrite(LED_BUILTIN, LOW); // zet de LED uit door het spanningsniveau LOW te maken
16 delay(1000); // wacht een seconde
```

De functie `loop()` wordt voortdurend uitgevoerd, dus de LED blijft knipperen.

## Referentie

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

Configureert de opgegeven pin om zich als invoer of uitvoer te gedragen.

### Parameters

**pin:** het pinnummer waarvan u de modus wilt instellen.

**modus:** INPUT, OUTPUT, INPUT\_PULLDOWN of INPUT\_PULLUP.

```
void digitalWrite (int pin, int value);
```

Schrijft de waarde HOOG of LAAG (1 of 0) naar de gegeven pin die eerder als uitgang moet zijn ingesteld.

### Parameters

**pin:** het pinnummer waarvan u de waarde wilt instellen.

**value:** de waarde van de pin. HIGH, LOW, 1, 0.

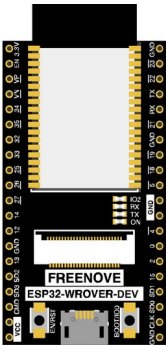
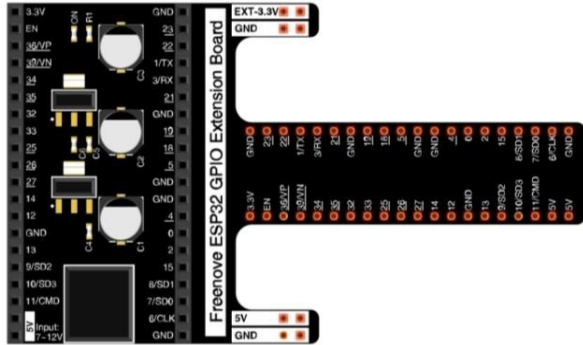
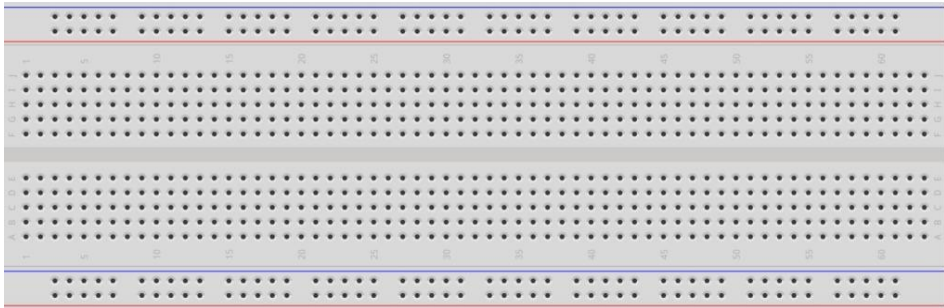



## Hoofdstuk 1 LED

Dit hoofdstuk is het startpunt van de reis om elektronische ESP32-projecten te bouwen en te verkennen. We beginnen met een eenvoudig "Blink" -project.

### Project 1.1 Blink

In dit project zullen we de ESP32 gebruiken om het knipperen van de gemeenschappelijke LED te regelen

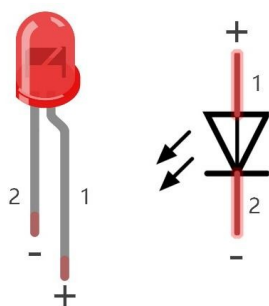
#### Componentenlijst

<div>ESP32-WROVER x1</div> <div></div>	<div>GPIO Uitbreidingskaart x1</div> <div></div>	
<div>Breadboard (Experimenteerbord) x1</div> <div></div>		
<div>LED x1</div> <div></div>	<div>Weerstand 220Ω x1</div> <div></div>	<div>Verbindingsdraad M/M x2</div> <div></div>

## Componentenkennis

### LED

Een LED is een soort diode. Alle diodes werken alleen als de stroom in de juiste richting vloeit en twee polen heeft. Een LED zal alleen werken (oplichten) als de langere pin (+) van de LED is aangesloten op de positieve uitgang van een stroombron en de kortere pin is aangesloten op de negatieve (-). Negatieve uitgang wordt ook wel aarde (GND) genoemd. Dit type component staat bekend als "diodes" (denk aan eenrichtingsverkeer). Alle gangbare 2-draadsdiodes zijn in dit opzicht hetzelfde. Diodes werken alleen als de spanning van de positieve elektrode hoger is dan die van de negatieve elektrode. Er is maar een klein spanningsbereik voor de meeste gangbare diodes van 1,9 en 3,4 V. Als u meer dan 3,3 V gebruikt, wordt de LED beschadigd en doorgebrand.

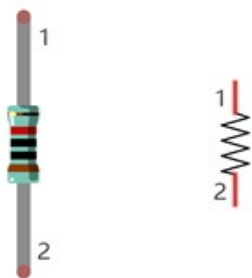


LED	Voltage (U)	Maximum stroom (I)	Aanbevolen stroom (I)
Roob	1,9V – 2,2V	20mA	10mA
Groen	2,9V – 3,4V	10mA	5mA
Blauw	2,9V – 3,4V	10mA	5mA

Let op: LED's kunnen niet rechtstreeks op een voeding worden aangesloten, wat meestal uitmondt in een beschadigd onderdeel. Een weerstand met een gespecificeerde weerstandswaarde moet in serie worden aangesloten op de LED die u wilt gebruiken.

### Weerstand

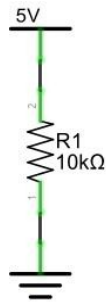
Weerstanden gebruiken Ohm ( $\Omega$ ) als meeteenheid voor hun weerstand (R).  $1\text{M}\Omega=1000\text{k}\Omega$ ,  $1\text{k}\Omega=1000\Omega$ . Een weerstand is een passieve elektrische component die de hoeveelheid stroom in een elektronisch circuit beperkt of regelt. Aan de linkerkant zien we een fysieke weergave van een weerstand, en aan de rechterkant is het symbool dat wordt gebruikt om de aanwezigheid van een weerstand in een schakelschema of schema weer te geven.



De gekleurde banden op een weerstand zijn een verkorte code die wordt gebruikt om de weerstandswaarde ervan te identificeren. Raadpleeg de bijlage van deze tutorial voor meer informatie over de kleurcodes van weerstanden. Bij eenzelfde spanning (U) zal de stroomsterkte (I) verkleinen als de weerstand (R) vergoogd wordt.

De relatie tussen stroom, spanning en weerstand kan worden uitgedrukt met deze formule:  $I=U/R$ , bekend als de wet van Ohm, waarbij I = stroom, U = spanning en R = weerstand. Als u de waarden van twee van deze kent, kunt u de waarde van de derde berekenen.

In het volgende diagram is de stroom door R1:  $I=U/R=5V/10k\Omega=0,0005A=0,5mA$ .

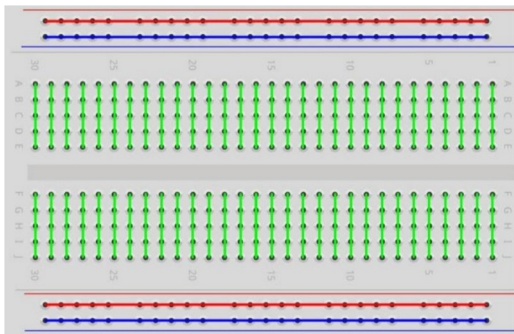


**WAARSCHUWING:** Sluit de twee polen van een voeding nooit aan op iets met een lage weerstandswaarde (d.w.z. een metalen voorwerp of blootliggende draad). Dit is kortsluiting en resulteert in een hoge stroomsterkte die de voeding en elektronische componenten kan beschadigen.

**Opmerking:** in tegenstelling tot LED's en diodes hebben weerstanden geen polen en zijn ze *niet-polair* (het maakt niet uit in welke richting je ze in een circuit steekt, het werkt hetzelfde).

### Breadboard

Hier hebben we een klein breadboard als voorbeeld van hoe de rijen gaten (contactdozen) elektrisch zijn bevestigd. De linker afbeelding toont de manier om pinnen aan te sluiten. De rechter foto toont de praktische interne structuur.



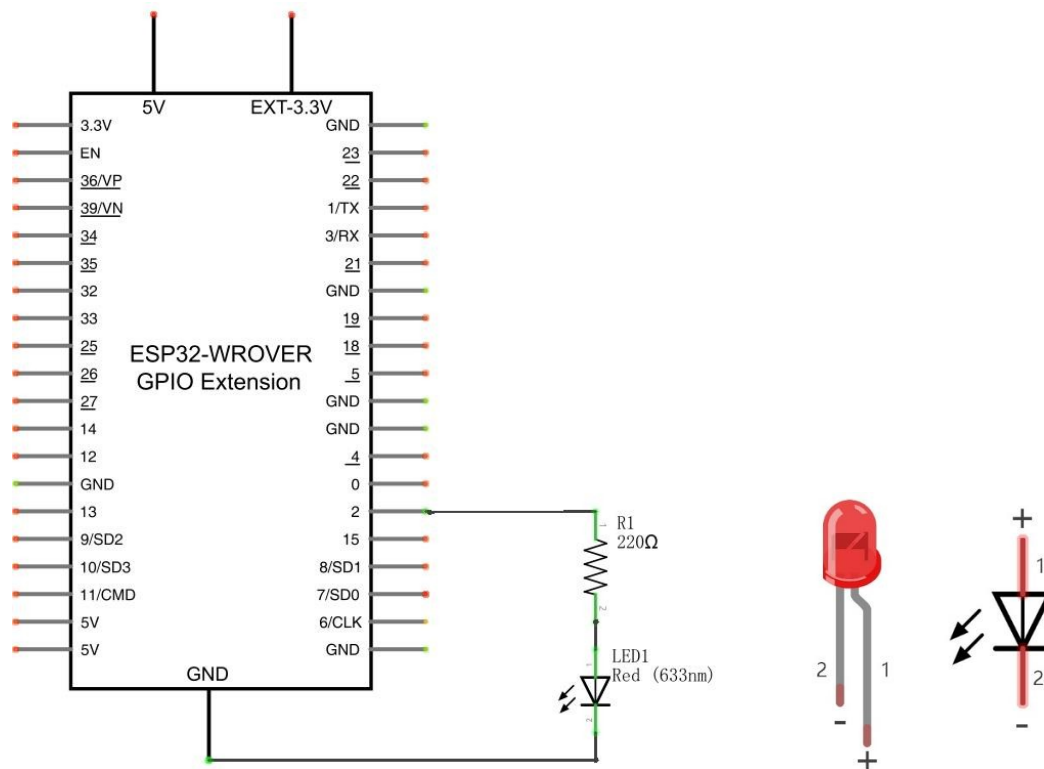
### Schakeling

Koppel eerst alle stroom los van de ESP32-WROVER. Bouw vervolgens de schakeling volgens het schematisch- en hardware diagram. Nadat de schakeling is gebouwd en correct is geverifieerd, sluit u de pc aan op ESP32-WROVER.

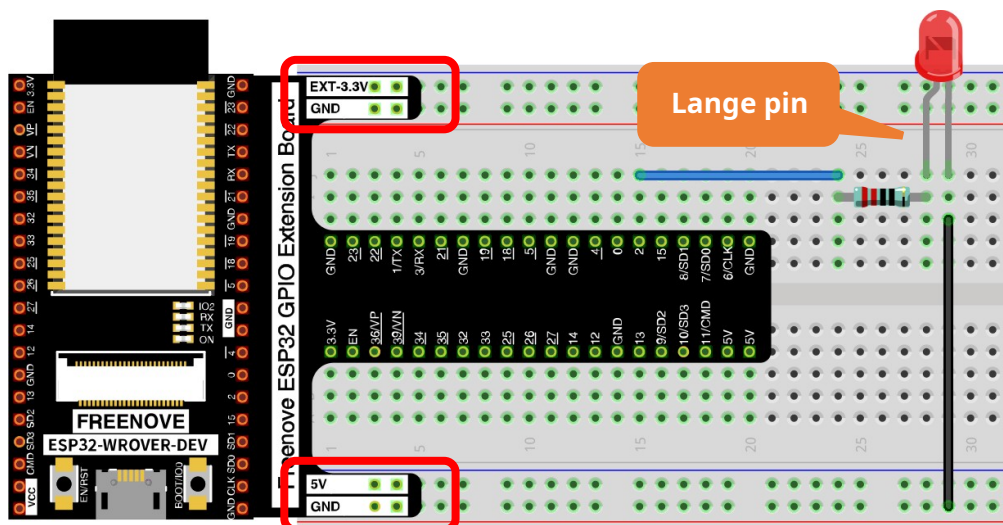
**LET OP:** Vermijd eventuele kortsluiting (vooral bij het aansluiten van 5V of GND, 3,3V en GND)!

**WAARSCHUWING:** Een kortsluiting kan een hoge stroomsterkte in uw schakeling veroorzaken, overmatige hitte van de componenten genereren en permanente schade aan uw hardware veroorzaken!

## Schematisch diagram



## Hardware verbindingen



**Draai de ESP32-WROVER niet 180°!**

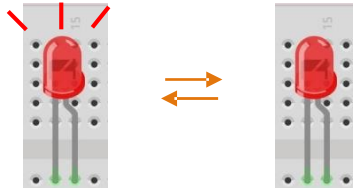
## Schets

Volgens de schakeling gaat de LED AAN als GPIO2 van de ESP32-WROVER hoog is. Omgekeerd, wanneer GPIO2 laag is, gaat de LED UIT. Daarom kunnen we GPIO2 circulair hoog en laag maken om de LED te laten knipperen.

Upload, zoals in het vorige hoofdstuk, de volgende schets:

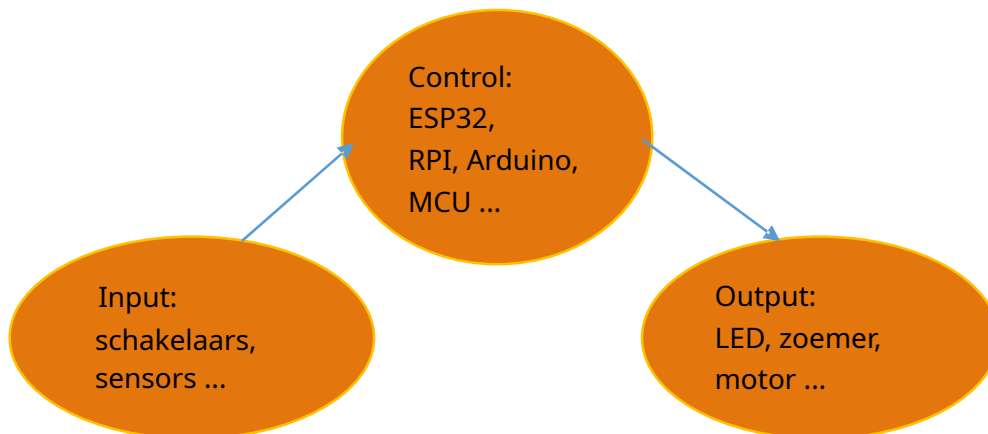
***Freenove\_Ultimate\_Starter\_Kit\_for\_ESP32/Sketches/Sketch\_01.1\_Blink.***

De LED op het breadboard zal nu knipperen



## Hoofdstuk 2 Knop & LED

Normaal gesproken zijn er drie essentiële onderdelen in een volledig automatisch besturingsapparaat: INPUT, OUTPUT en CONTROL. In het laatste gedeelte was de LED het uitvoergedeelte en ESP32 het besturingsgedeelte. In praktische toepassingen laten we niet alleen LED's knipperen, maar laten we een apparaat ook de omgeving waarnemen, instructies ontvangen en vervolgens de juiste actie ondernemen, zoals LED's oplichten, een zoemer laten inschakelen, enzovoort.

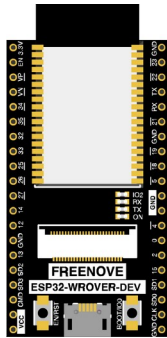
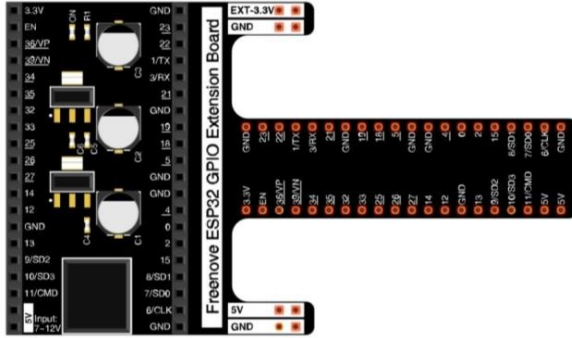
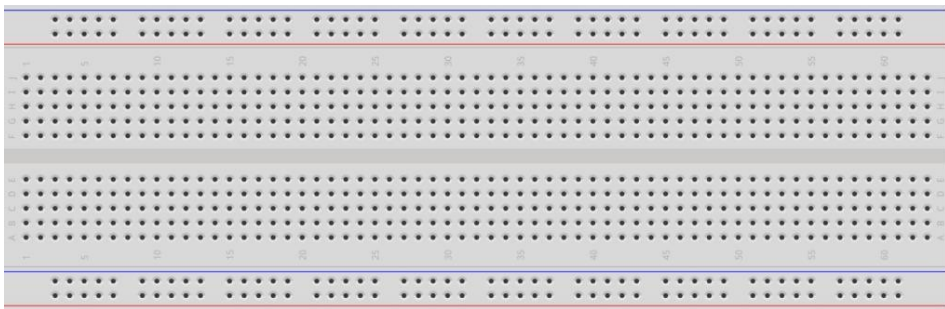







Vervolgens gaan we een eenvoudig besturingssysteem bouwen om een LED via een drukknop te besturen.

### Project 2.1 Knop & LED

In dit project zullen we de LED-status regelen via een drukknop. Wanneer de knop wordt ingedrukt, gaat onze LED aan, en wanneer deze wordt losgelaten, gaat de LED uit.

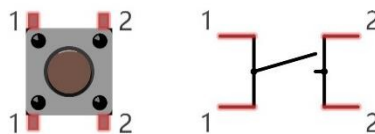
## Componentenlijst

<div>ESP32-WROVER x1</div> <div></div>	<div>GPIO Uitbreidingsbord x1</div> <div></div>			
<div>Breadboard x1</div> <div></div>				
<div>Verbindingsdraad M/M x4</div> <div></div>	<div>LED x1</div> <div></div>	<div>Weerstand 220Ω x1</div> <div></div>	<div>Weerstand 10kΩ x2</div> <div></div>	<div>Drukknop x1</div> <div></div>

## Componentenkennis

### Drukknop

Dit type drukknopschakelaar heeft 4 pinnen (2-polige schakelaar). Twee pinnen aan de linkerkant zijn verbonden en zowel de linker- als de rechterkant zijn hetzelfde zoals te zien is in de afbeelding:

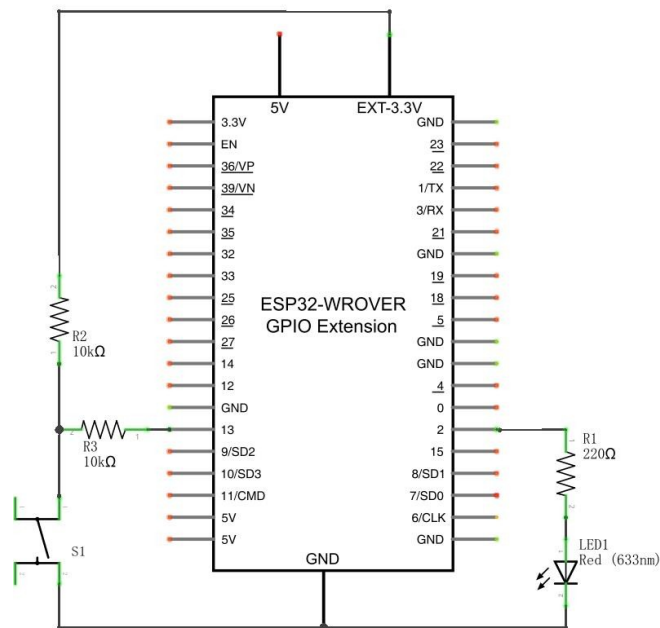


Wanneer de knop op de schakelaar wordt ingedrukt, is de schakeling gesloten (LED is dan ingeschakeld).

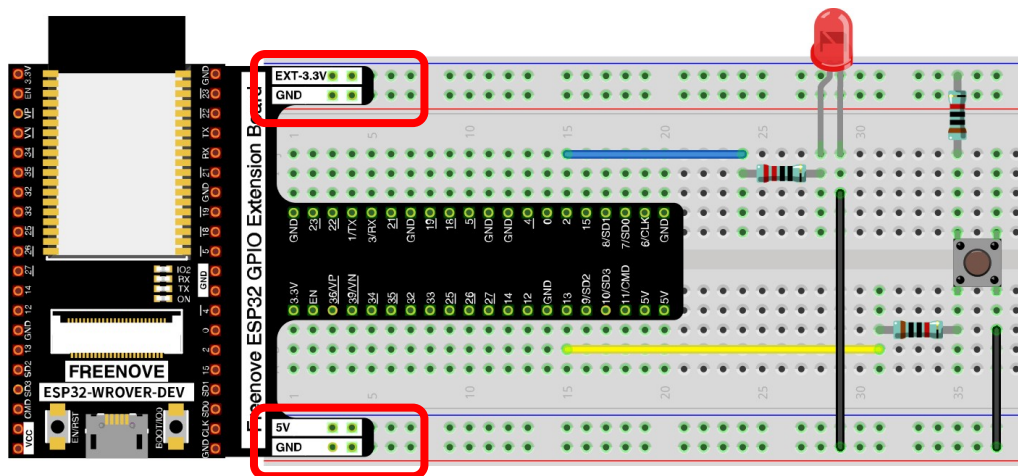


## Schakeling

Schematisch diagram



## Hardware verbindingen



## Schets

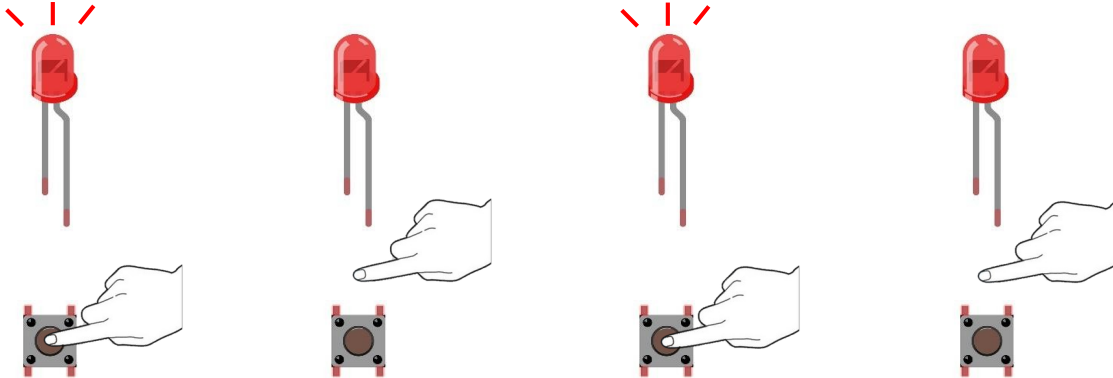
Dit project is bedoeld om te leren hoe je een drukknop gebruikt om een LED te bedienen. We moeten eerst de status van de schakelaar lezen en vervolgens bepalen of de LED moet worden ingeschakeld in overeenstemming met de status van de schakelaar.

Upload de volgende schets:

## Freenove Ultimate Starter Kit for ESP32/Sketches/Sketch 02.1 ButtonAndLed

### Sketch\_02.1\_ButtonAndLed

De LED zal nu aan gaan wanneer je op de drukknop drukt. Als je de drukknop los laat, zal de LED terug uit gaan.



Dit is de programmacode:

```
1 // de ingebouwde LED op de ESP32-WROVER is aangesloten op pin 2
2 #define PIN_LED 2
3 // de drukknop wordt aangesloten op pin 13
4 #define PIN_BUTTON 13
5
6 // de setup-functie wordt één keer uitgevoerd wanneer u op
7 // reset drukt of het bord van stroom voorziet
8 void setup(){
9   // initialiseer digitale pin PIN_LED als uitvoer.
10  pinMode(PIN_LED, OUTPUT);
11  // initialiseer digitale pin PIN_BUTTON als invoer.
12  pinMode(PIN_BUTTON, INPUT);
13 }
14
15 // de loop-functie loopt voor altijd door. Op het einde begint ze gewoon terug opnieuw
16 void loop(){
17   if (digitalRead(PIN_BUTTON) == LOW){ // als de invoer van de drukknop laag is (ingedrukt)
18     digitalWrite(PIN_LED, HIGH); // zet de LED aan door het spanningsniveau HIGH te maken
19   } else {
20     digitalWrite(PIN_LED, LOW); // zet de LED uit door het spanningsniveau LOW te maken
21   }
22 }
```

In de schakeling zijn de LED en de drukknop respectievelijk verbonden met GPIO2 en GPIO13, dus definieer PIN\_LED en PIN\_BUTTON respectievelijk als 2 en 13.

```
1 // de LED is aangesloten op pin 2 (de ingebouwde LED)
2 #define PIN_LED 2
3 // de drukknop is aangesloten op pin 13
4 #define PIN_BUTTON 13
```

Gebruik in de loop() functie digitalRead(buttonPin) om de status van de knop te bepalen. Wanneer de knop wordt ingedrukt, geeft de functie een laag (LOW) niveau terug, het resultaat van if is waar en schakel vervolgens de LED in. Schakel anders de LED uit.

```
16 void loop(){
17   if (digitalRead(PIN_BUTTON) == LOW){ // als de invoer van de drukknop laag is (ingedrukt)
18     digitalWrite(PIN_LED, HIGH); // zet de LED aan door het spanningsniveau HIGH te maken
19   } else {
20     digitalWrite(PIN_LED, LOW); // zet de LED uit door het spanningsniveau LOW te maken
21   }
22 }
```

## Referentie

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

Deze functie geeft de uitgelezen waarde terug van de opgegeven pin. Het zal "HIGH" of "LOW" (1 of 0) zijn, afhankelijk van het logische niveau op de pin.

**Parameters**

**pin:** het pinnummer waarvan u de waarde wilt uitlezen.

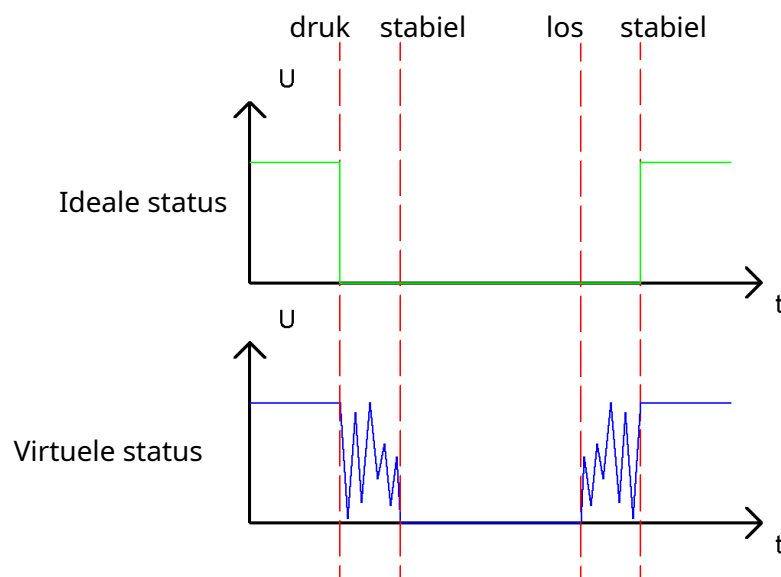
## Project 2.2 MINI tafellamp

We zullen ook een drukknop, LED en ESP32 gebruiken om een MINI-tafellamp te maken, maar deze zal anders werken: druk op de knop, de LED gaat AAN, en als je nogmaals op de knop drukt, gaat de LED UIT. De actie van de AAN-schakelaar is niet langer kortstondig (zoals bij een deurbel), maar blijft AAN zonder dat u voortdurend op de knopschakelaar hoeft te drukken.

Laten we eerst iets leren over de drukknopschakelaar.

### Debounce voor de drukknop

Op het moment dat een drukknop wordt ingedrukt, verandert deze niet onmiddellijk van de ene toestand naar de andere. Als gevolg van kleine mechanische trillingen zal er een korte periode van voortdurend in- en uitschakelen zijn, voordat het volledig een andere toestand bereikt, die te snel is voor mensen om te detecteren maar niet voor microcontrollers. Hetzelfde geldt wanneer de drukknop wordt losgelaten. Dit ongewenste fenomeen staat bekend als stuiten of 'bounce'.



Als we de status van de drukknop direct kunnen detecteren, zijn er dus meerdere indrukken en loslaten in één drukcyclus. Dit zal de snelle werking van de microcontroller misleiden en veel foute beslissingen veroorzaken. Daarom moeten we de impact van het stuiten elimineren. Onze oplossing: meerdere keren de staat van de knop beoordelen. Alleen wanneer de knopstatus gedurende een bepaalde periode stabiel (consistent) is, kan dit aangeven dat de knop daadwerkelijk in de AAN-status staat (wordt ingedrukt). Dit project heeft dezelfde componenten en circuits nodig als we in de vorige sectie hebben gebruikt.

### Schets

Upload de volgende schets:

***Freenove\_Ultimate\_Starter\_Kit\_for\_ESP32/Sketches/Sketch\_02.2\_Tablelamp***

## Sketch\_02.2\_Tablelamp

Druk nu op de knop, de LED gaat aan, druk nogmaals en de LED gaat terug uit.

Dit is de programmacode:

```
1  #define PIN_LED 2
2  #define PIN_BUTTON 13
3
4  void setup() {
5      pinMode(PIN_LED, OUTPUT);
6      pinMode(PIN_BUTTON, INPUT);
7  }
8
9  void loop() {
10     if (digitalRead(PIN_BUTTON) == LOW) {
11         delay(20);
12         if (digitalRead(PIN_BUTTON) == LOW) {
13             reverseGPIO(PIN_LED);
14         }
15         while (digitalRead(PIN_BUTTON) == LOW);
16         delay(20);
17     }
18 }
19
20 void reverseGPIO(int pin) {
21     digitalWrite(pin, !digitalRead(pin));
22 }
```

Wanneer u de status van de drukknop beoordeelt en deze wordt gedetecteerd als "ingedrukt", wacht dan een bepaalde tijd (20ms) voordat u deze opnieuw detecteert om het effect van 'bounce' te elimineren. Wanneer u dit bevestigt, schakelt u de LED aan en uit. Vervolgens begint het te wachten tot de ingedrukte knop wordt losgelaten, en wacht het een bepaalde tijd (20ms) om het effect van 'bounce' te elimineren nadat de knop is losgelaten.

```
10 if (digitalRead(PIN_BUTTON) == LOW) {
11     delay(20);
12     if (digitalRead(PIN_BUTTON) == LOW) {
13         reverseGPIO(PIN_LED);
14     }
15     while (digitalRead(PIN_BUTTON) == LOW);
16     delay(20);
17 }
```

De functie `reverseGPIO()` (die we zelf een eigen naam hebben gegeven) betekent het lezen van de statuswaarde van de opgegeven pin, het omkeren van de waarde (!) en het opnieuw schrijven naar de pin om de functie van het omdraaien van de status van de pin te bereiken.

```
20 void reverseGPIO(int pin) {
21     digitalWrite(pin, !digitalRead(pin));
22 }
```

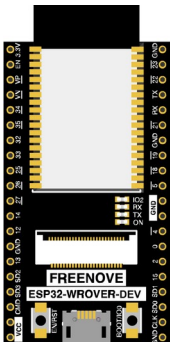
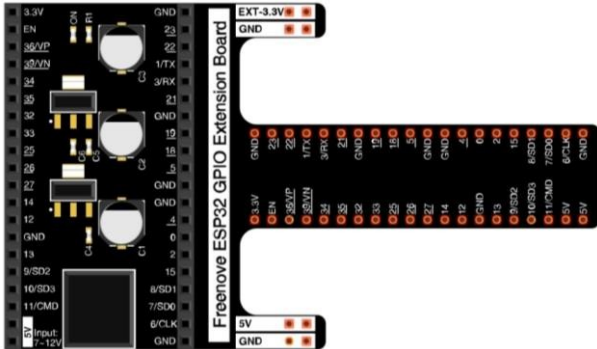
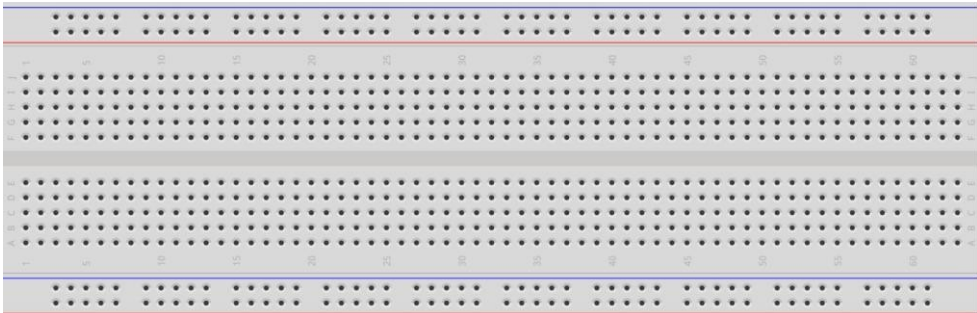



## Hoofdstuk 3 LED bar

We hebben geleerd hoe je een knipperende LED kunt aansturen, nu gaan we leren hoe je een aantal LED's kunt aansturen.

### Project 3.1 Flowing Light

In dit project gebruiken we een aantal LED's om een vloeiend licht te creëren.

#### Componentenlijst

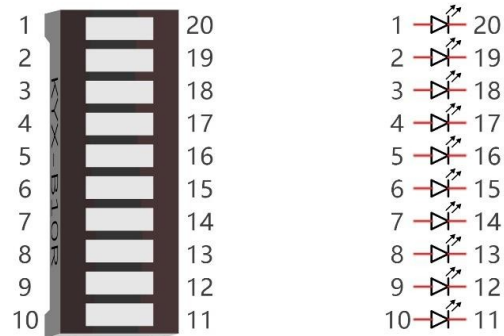
<div>ESP32-WROVER x1</div> <div></div>	<div>GPIO Uitbreidingsbord x1</div> <div></div>	
<div>Breadboard x1</div> <div></div>		
<div>Verbindingsdraad M/M x10</div> <div></div>	<div>LED bar graph x1</div> <div></div>	<div>Weerstand 220Ω x10</div> <div></div>

## Componentenkennis

Laten we de basisfuncties van deze componenten leren kennen, zodat we ze beter kunnen gebruiken en begrijpen.

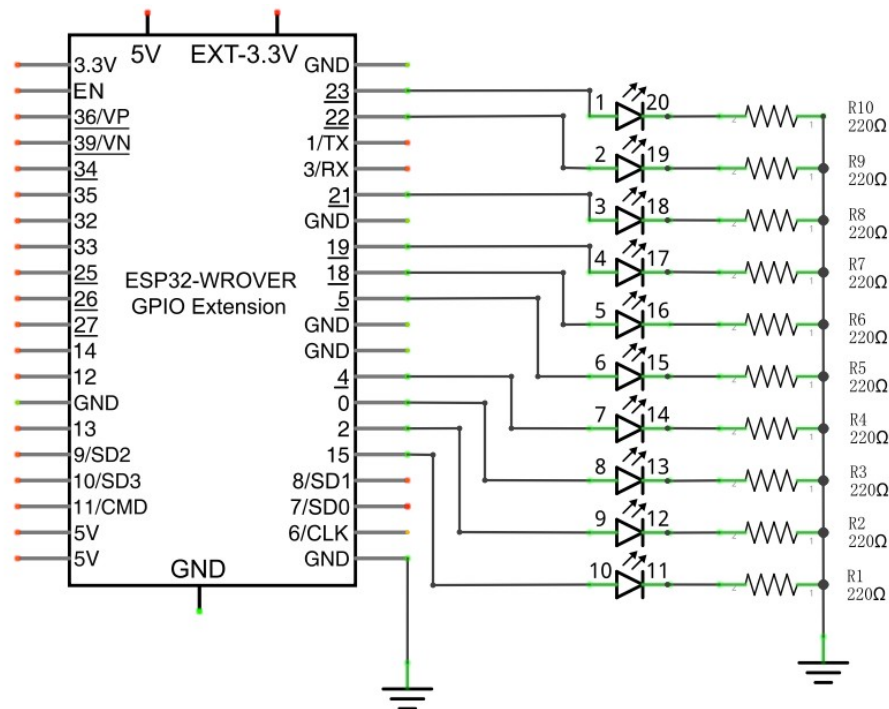
### LED bar

Een LED bar heeft 10 LED's (segmenten) geïntegreerd in één compact onderdeel. De twee rijen pinnen aan de onderkant zijn gekoppeld om elke LED te identificeren, net als de eerder gebruikte enkele LED.

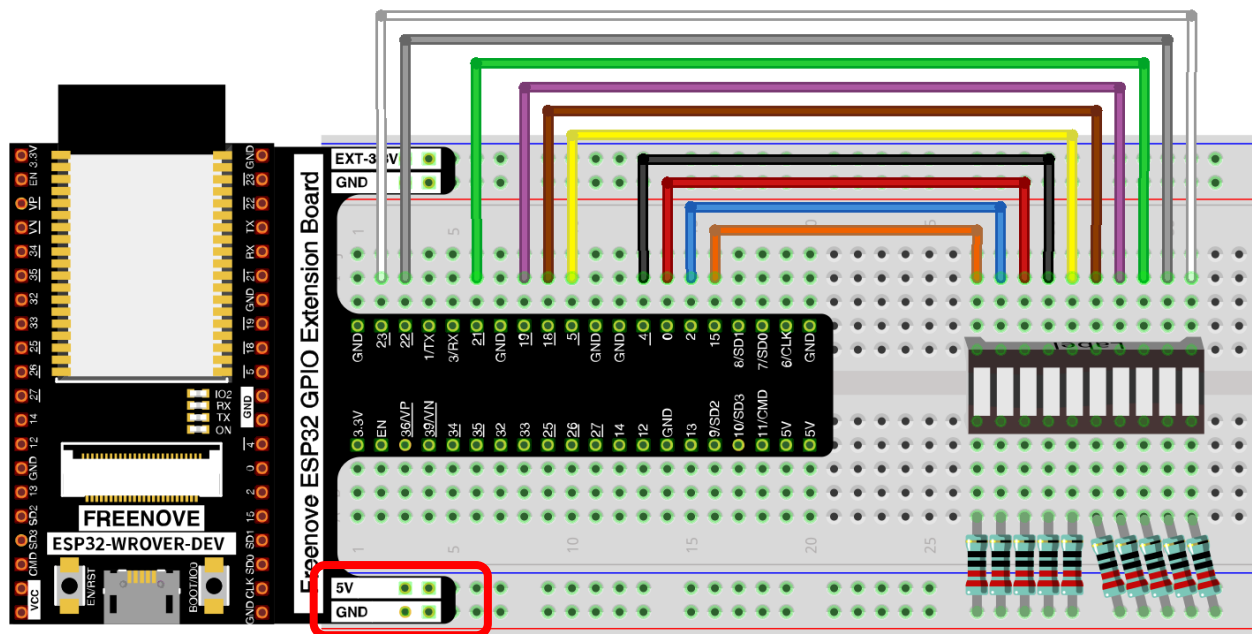


## Schakeling

## Schamatisch diagram



## Hardware verbindingen



Als de LED bar niet werkt, probeer het dan 180° te draaien. Het label is willekeurig.



## Schets

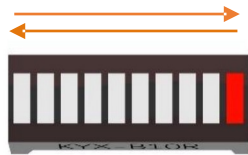
Dit project is ontworpen om een stromend waterlamp te maken. Wat zijn deze acties: Schakel eerst LED #1 AAN en vervolgens UIT. Zet vervolgens LED #2 AAN en vervolgens UIT en herhaal hetzelfde voor alle 10 de LED's totdat de laatste LED UIT gaat. Dit proces wordt herhaald om de “bewegingen” van stromend water te bereiken.

### Sketch\_03.1\_FlowingLight

Upload de volgende schets:

***Freenove\_Ultimate\_Starter\_Kit\_for\_ESP32/Sketches/Sketch\_03.1\_FlowingLight***

De LED bar licht op van links naar rechts en van rechts naar links.



Dit is de programmacode:

```
1 byte ledPins[] = {15, 2, 0, 4, 5, 18, 19, 21, 22, 23};
2 int ledCounts;
3
4 void setup() {
5     ledCounts = sizeof(ledPins);
6     for (int i = 0; i < ledCounts; i++) {
7         pinMode(ledPins[i], OUTPUT);
8     }
9 }
10
11 void loop() {
12     for (int i = 0; i < ledCounts; i++) {
13         digitalWrite(ledPins[i], HIGH);
14         delay(100);
15         digitalWrite(ledPins[i], LOW);
16     }
17     for (int i = ledCounts - 1; i > -1; i--) {
18         digitalWrite(ledPins[i], HIGH);
19         delay(100);
20         digitalWrite(ledPins[i], LOW);
21     }
22 }
```

Gebruik een array om 10 GPIO-poorten te definiëren die zijn aangesloten op een LED bar voor eenvoudigere bediening.

```
1 byte ledPins[] = {15, 2, 0, 4, 5, 18, 19, 21, 22, 23};
```

Gebruik in setup() sizeof() om de grootte van een array op te halen, wat in ons geval het aantal LED's is, en configureer vervolgens de GPIO-poort naar de uitvoermodus.

```
5 ledCounts = sizeof(ledPins);
6 for (int i = 0; i < ledCounts; i++) {
7     pinMode(ledPins[i], OUTPUT);
8 }
```

Gebruik vervolgens in `loop()` twee `for` lussen om stromend waterlicht van links naar rechts en van rechts naar links te realiseren.

```
12 for (int i = 0; i < ledCounts; i++) {  
13     digitalWrite(ledPins[i], HIGH);  
14     delay(100);  
15     digitalWrite(ledPins[i], LOW);  
16 }  
17 for (int i = ledCounts - 1; i > -1; i--) {  
18     digitalWrite(ledPins[i], HIGH);  
19     delay(100);  
20     digitalWrite(ledPins[i], LOW);  
21 }
```

## Hoofdstuk 4 Analooq en PWM

In eerder onderzoek wisten we dat één knop twee statussen heeft: ingedrukt en losgelaten, en dat de LED een aan en uit status heeft. Hoe kunnen we dan naar een middenstatus gaan? Hoe kan ik een tussenstatus uitvoeren om de LED "halfhelder" te laten zijn? Dat is wat we gaan leren.

Laten we eerst eens kijken hoe we de helderheid van een LED kunnen regelen.

### Project 4.1 Breathing LED

Ademhalingslicht, dat wil zeggen dat de LED geleidelijk van uit naar aan wordt gezet, en geleidelijk van aan naar uit, net als bij "ademen". Dus, hoe regel je de helderheid van een LED? Om dit doel te bereiken, zullen we PWM gebruiken.

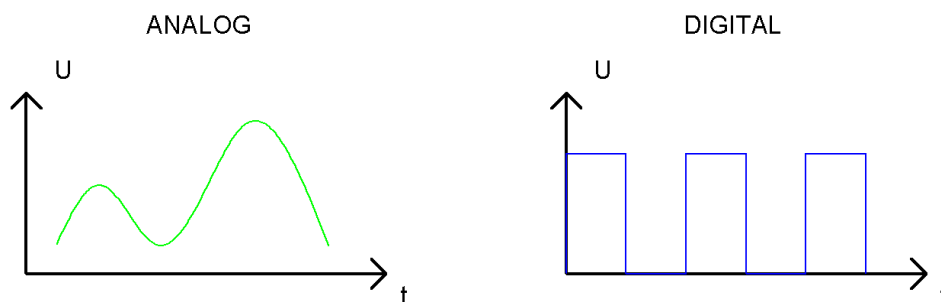
### Componentenlijst

We gebruiken dezelfde Componentenlijst als Project 1.1 Blink

### Gerelateerde kennis

#### Analooq & digitaal

Een analooq signaal is een continu signaal in zowel tijd als waarde. Integendeel, een digitaal signaal of een discreet tijdsignaal is een tijdreeks die bestaat uit een reeks grootheden. De meeste signalen in het leven zijn analoge signalen. Een bekend voorbeeld van een analooq signaal is dat de temperatuur gedurende de dag voortdurend verandert en niet plotseling van 0°C naar 10°C kan veranderen. Digitale signalen kunnen echter onmiddellijk in waarde veranderen. Deze verandering wordt uitgedrukt in cijfers als 1 en 0 (de basis van binaire code). Hun verschillen kunnen gemakkelijker worden gezien als ze worden vergeleken in de onderstaande grafiek.

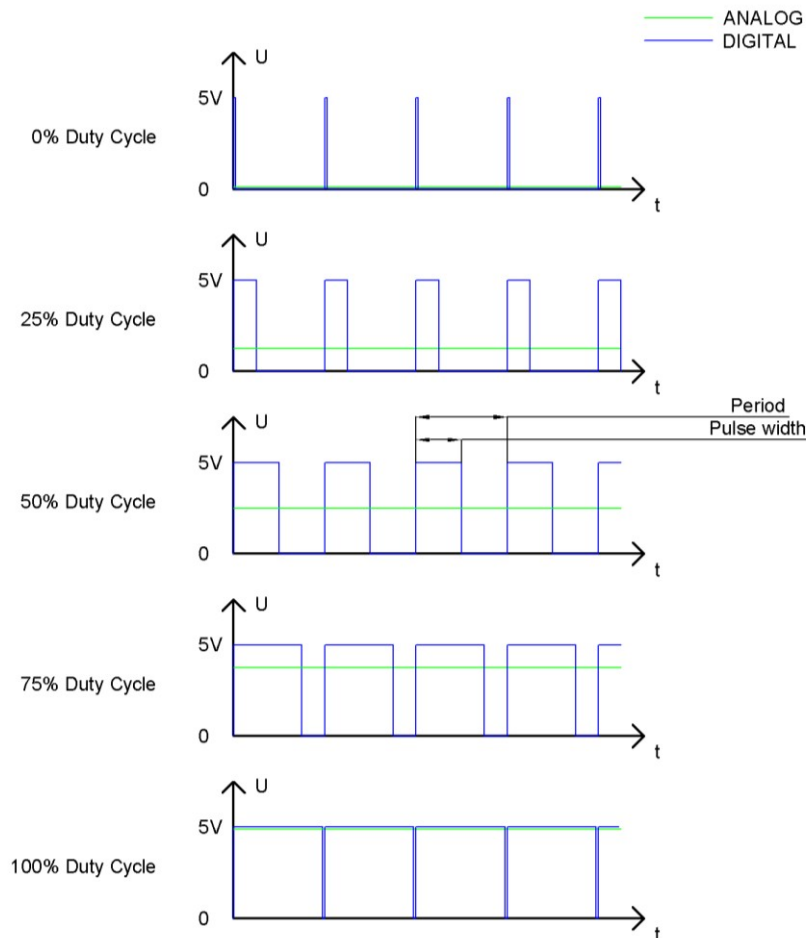


In de praktijk gebruiken we vaak binair als digitaal signaal, dat wil zeggen een reeks nullen en enen. Omdat een binair signaal slechts twee waarden heeft (0 of 1), heeft het een grote stabiliteit en betrouwbaarheid. Ten slotte kunnen zowel analoge als digitale signalen in elkaar worden omgezet.

#### PWM

PWM, Pulse-Width Modulation, (pulsbreedte modulatie) is een zeer effectieve methode om digitale signalen te gebruiken om analoge circuits te besturen. Gemeenschappelijke processors kunnen analoge signalen niet rechtstreeks uitvoeren. PWM-technologie maakt het erg gemakkelijk om deze conversie (vertaling van digitale naar analoge signalen) te realiseren. PWM-technologie maakt gebruik van digitale pinnen om

bepaalde frequenties van vierkante golven te verzenden, dat wil zeggen de uitvoer van hoge en lage niveaus, die afwisselend een tijdje aanhouden. De totale tijd voor elke reeks hoge en lage niveaus ligt over het algemeen vast, wat de periode wordt genoemd (opmerking: het omgekeerde van de periode is de frequentie). De tijd van uitgangen op hoog niveau wordt over het algemeen "pulsbreedte" genoemd, en de duty-cycle (werkcyclus) is het percentage van de verhouding van de pulsduur, of pulsbreedte (PW) tot de totale periode (T) van de golfvorm. Hoe langer de uitgangen van hoge niveaus aanhouden, hoe langer de duty-cycle en hoe hoger de overeenkomstige spanning in het analoge signaal zal zijn. De volgende afbeeldingen laten zien hoe de analoge signaalspanningen variëren tussen 0V-5V (hoog niveau is 5V), overeenkomend met de pulsbreedte 0%-100%:



Hoe langer de PWM-dutycycle is, hoe hoger het uitgangsvermogen zal zijn. Nu we deze relatie begrijpen, kunnen we PWM gebruiken om de helderheid van een LED of de snelheid van de DC-motor enzovoort te regelen. Uit het bovenstaande blijkt duidelijk dat PWM niet echt analoog is en dat de effectieve waarde van de spanning equivalent is aan de overeenkomstige analoog. Daarom kunnen we het uitgangsvermogen van de LED en andere uitgangsmodule regelen om verschillende effecten te bereiken.

### ESP32 and PWM

In de ESP32 heeft de LEDC(PWM)-controller 16 afzonderlijke kanalen, die elk onafhankelijk de frequentie, de werkcyclus en zelfs de nauwkeurigheid kunnen regelen. In tegenstelling tot traditionele PWM-pinnen zijn de PWM-uitgangspinnen van ESP32 configureerbaar, met een of meer PWM-uitgangspinnen per kanaal. De relatie tussen de maximale frequentie en bitprecisie wordt weergegeven in de volgende formule, waarbij de maximale waarde van *bit* 31 is

$$\text{Freq}_{\max} = \frac{80.000.000}{1 \ll bit}$$

8

Genereer bijvoorbeeld een PWM met een precisie van 8 bits ( $2^8 = 256$ . Waarden variëren van 0 tot 255) met een maximale frequentie van  $80.000.000/256 = 312.500$  Hz.

## Schakeling

De Schakeling is dezelfde als Project 1.1 Blink.

## Schets

Dit project is ontworpen om PWM-uitvoer GPIO2 te maken waarbij de pulsbreedte toeneemt van 0% naar 100% en vervolgens geleidelijk wordt verlaagd van 100% naar 0%.

### Sketch\_04.1\_BreathingLight

Upload de volgende schets:

**Freenove\_Ultimate\_Starter\_Kit\_for\_ESP32/Sketches/Sketch\_04.1\_BreathingLight**

Je zult zien dat de LED geleidelijk van uit naar aan wordt gezet en vervolgens aan uit uit aan, net als bij ademen.



Dit is de programmacode:

```
1  #define PIN_LED 2 //definieer de led pin
2  #define CHN 0 //definieer het pwm kanaal
3  #define FRQ 1000 //definieer de pwm frequentie
4  #define PWM_BIT 8 //definieer de pwm nauwkeurigheid
5
6  void setup() {
7      ledcSetup(CHN, FRQ, PWM_BIT); //pwm kanaal instellen
8      ledcAttachPin(PIN_LED, CHN); //verbind de led pin met het pwm kanaal
9  }
10
11 void loop() {
12     for (int i = 0; i < 255; i++) { //laat de led langzaam aan gaan
13         ledcWrite(CHN, i);
14         delay(10);
15     }
16     for (int i = 255; i > -1; i--) { //laat de led langzaam uit gaan
17         ledcWrite(CHN, i);
18         delay(10);
19     }
20 }
```

De PWM-pin-uitvoermodus van ESP32 is niet hetzelfde als de traditionele controller. Het bestuurt elke parameter van PWM door het PWM-kanaal te besturen. Er kan een willekeurig aantal GPIO's worden aangesloten op het PWM-kanaal om PWM uit te voeren. In `setup()` configureert u eerst een PWM-kanaal en stelt u de frequentie en precisie in.

```
7 ledcSetup(CHN, FRQ, PWM_BIT); //pwm kanaal instellen
```

Vervolgens wordt de GPIO gekoppeld aan het PWM-kanaal.

```
8 ledcAttachPin(PIN_LED, CHN); //verbind de led pin met het pwm kanaal
```

Vervolgens wordt de GPIO geassocieerd met het PWM-kanaal. In de `loop()` zijn er twee `for` lussen. De eerste maakt de `PIN_LED` PWM-uitgang van 0% naar 100% en de tweede zorgt ervoor dat de `PIN_LED` PWM-uitgang van 100% naar 0% gaat. Hierdoor kan de LED geleidelijk oplichten en doven.

```
12 for (int i = 0; i < 255; i++) { //laat de led langzaam aan gaan
13     ledcWrite(CHN, i);
14     delay(10);
15 }
16 for (int i = 255; i > -1; i--) { //laat de led langzaam uit gaan
17     ledcWrite(CHN, i);
18     delay(10);
19 }
```

U kunt ook de snelheid van de statusverandering van de LED aanpassen door de parameter van de `delay()` functie in de `for` lus te wijzigen.

## Referentie

```
double ledcSetup(uint8_t chan, double freq, uint8_t bit_num)
```

Stel de frequentie en nauwkeurigheid van een PWM-kanaal in.

### Parameters

**chan**: kanaalindex. Waardebereik: 0-15

**freq**: frequentie, dit kan een decimaal zijn.

**bit\_num**: precisie van waarden.

```
void ledcAttachPin(uint8_t pin, uint8_t channel)
```

Koppel **pin** aan PWM kanaal **channel**.

```
void ledcDetachPin(uint8_t pin)
```

Ontkoppel **pin** van het PWM kanaal waaraan het gekoppeld is.

```
void ledcWrite(uint8_t channel, uint32_t duty)
```

Schrijft de pulsbreedte waarde **duty** naar PWM-kanaal **channel**.

## Project 4.2 Meteor Flowing Light

Nadat we over PWM hebben geleerd, kunnen we het gebruiken om een LED-bar te regelen en een koeler stromend licht te realiseren. De componentenlijst, de schakeling en de hardware komen exact overeen met Project 3.1 Flowing Light.

### Schets

Een meteorlicht zal worden geïmplementeerd met PWM.

#### Sketch\_04.2\_FlowingLight2

Upload de volgende schets:

#### ***Freenove\_Ultimate\_Starter\_Kit\_for\_ESP32/Sketches/Sketch\_04.2\_FlowingLight2***

De LED-balk zal geleidelijk oplichten en van links naar rechts uitgaan, licht dan weer op van rechts naar links en gaat geleidelijk uit.

```

1  const byte ledPins[] = { 15, 2, 0, 4, 5, 18, 19, 21, 22, 23 }; //definieer led pins
2  const byte channels[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }; //definieer pwm kanalen
3  const int dutys[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1023, 512, 256, 128, 64, 32, 16, 8,
4                      4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }; //definieer de pwm werkcyclussen
5  int ledCounts;
6  int delayTimes = 50; //de snelheid ,hoe kleiner, hoe sneller
7
8  void setup() {
9      ledCounts = sizeof(ledPins); //aantal led pinnen onthouden
10     for (int i = 0; i < ledCounts; i++) { //pwm kanalen opzetten
11         ledcSetup(channels[i], 1000, 10);
12         ledcAttachPin(ledPins[i], channels[i]);
13     }
14 }
15
16 void loop() {
17     for (int i = 0; i < 20; i++) { //van de ene naar de andere kant vloeien
18         for (int j = 0; j < ledCounts; j++) {
19             ledcWrite(channels[j], dutys[i + j]);
20         }
21         delay(delayTimes);
22     }
23     for (int i = 0; i < 20; i++) { //in omgekeerde richting vloeien
24         for (int j = ledCounts - 1; j > -1; j--) {
25             ledcWrite(channels[j], dutys[i + (ledCounts - 1 - j)]);
26         }
27         delay(delayTimes);
28     }
29 }

```

Eerst hebben we 10 GPIO-, 10 PWM-kanalen en 30 pulsbreedtewaarden gedefinieerd.

```

1  const byte ledPins[] = { 15, 2, 0, 4, 5, 18, 19, 21, 22, 23 }; //definieer led pins
2  const byte channels[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }; //definieer pwm kanalen
3  const int dutys[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1023, 512, 256, 128, 64, 32, 16, 8,
4                      4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }; //definieer de pwm werkcyclussen

```

In de setup() -functie worden de volgende stappen uitgevoerd:

- De variabele ledCounts wordt ingesteld op de lengte van het array ledPins, wat het aantal LED's aangeeft.
- Voor elke LED wordt een PWM-kanaal ingesteld.
- Voor elk PWM-kanaal wordt de PWM-frequentie ingesteld op 1000 Hz en de resolutie op 10 bits.
- Voor elk PWM-kanaal wordt de LED-pin gekoppeld aan het PWM-kanaal.

```
9 ledCounts = sizeof(ledPins); //aantal led pinnen onthouden
10 for (int i = 0; i < ledCounts; i++) { //pwm kanalen opzetten
11     ledcSetup(channels[i], 1000, 10);
12     ledcAttachPin(ledPins[i], channels[i]);
13 }
```



In de `loop()`-functie worden de volgende stappen uitgevoerd:

- Voor elke LED wordt de PWM-werkcyclus ingesteld op de waarde van het array `dutys` op de index `i + j`.
- De LED's worden zichtbaar gemaakt door de PWM-werkcyclus te schrijven.
- Er wordt een vertraging ingesteld om de animatie te vertragen.
- De loop wordt 20 keer herhaald.

De array `dutys` bevat de PWM-werkcyclussen voor de LED's. De PWM-werkcyclus is een waarde tussen 0 en 1023. Een waarde van 0 betekent dat de LED uit is, een waarde van 1023 betekent dat de LED altijd aan is.

In de eerste helft van de `loop()`-functie worden de LED's van links naar rechts gedimmed. In de tweede helft van de `loop()`-functie worden de LED's van rechts naar links gedimmed.

De snelheid van de animatie wordt bepaald door de waarde van de variabele `delayTimes`. Een kleinere waarde betekent een snellere animatie.