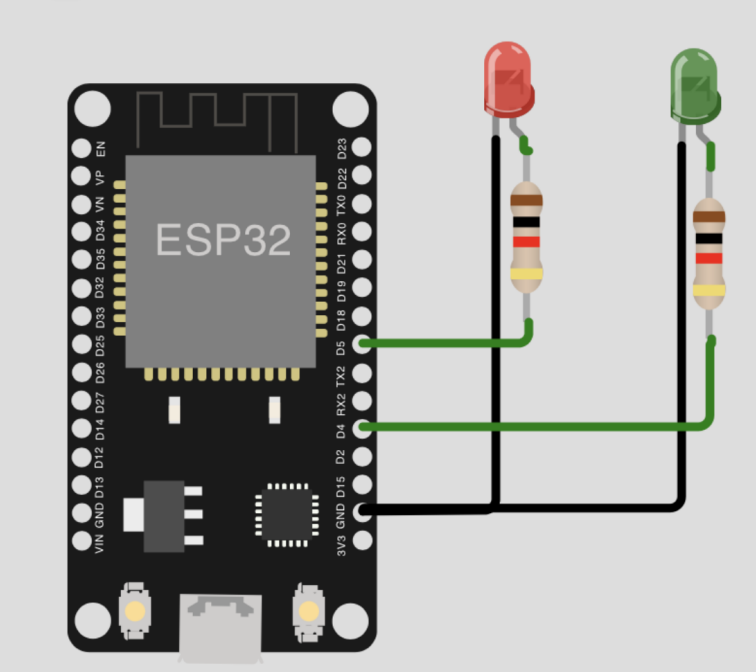
Hello World

Un microcontrolleur n'ayant pas d'écran, le premier programme que l'on essaye sur un microcontrolleur consiste le plus souvent à faire clignoter une led. C'est le « Hello World » des microcontrolleurs. Le montage traditionnel, avec une (ou plusieurs) led externe, est le suivant :



Ici deux leds sont reliées à un microcontrolleur : les cathodes à la masse, et l'anode de la led rouge à la broche D5 et celle de la led verte à la broche D4. Sans résistance, il est garanti que les leds ne survivent pas, et il est à craindre que le microcontrolleur soit lui aussi endommagé. (Bien sûr, pas de problème avec la simulation, et Wokwi ne met hélas pas en évidence le risque que l'on fait courir au matériel.)

Pour allumer l'une ou l'autre des diodes, il suffit de configurer le microcontrolleur pour que les broches D4 et D5 soient des sorties et de les placer à un état haut. Pour les éteindre, bien sûr on place la ou les broches à un état bas. Exemple (à tester directement dans la console) :

```
>>> from machine import Pin
>>> ledRouge = Pin(5, Pin.OUT)
>>> ledRouge.on()
>>> ledRouge.off()
```

Il est rare qu'un microcontrolleur soit équipé de sorties analogiques (c-à-d de convertisseur DAC : digital to analog converter) ce qui permettrait d'avoir une sortie de niveau variable, mais les microcontrolleurs contournent souvent la difficulté en émettant des signaux rectangulaires, c-à-d des signaux périodiques alternant entre le niveau haut et le niveau bas. On parle de signal modulé en largeur d'impulsion (PWM : pulse width modulation en anglais). Voici quelques lignes de code pour faire pulser la led verte (5 fois)

```
>>> from machine import PWM
>>> ledVerte = Pin(4, Pin.OUT)
>>> pwm = PWM(ledVerte, freq= 1000)

>>> def pulseV():
      for i in range(5):
        for j in range(102):
            pwm.duty(j*10)
            sleep(0.01)
        for j in range(102,-1,-1):
            pwm.duty(j*10)
            sleep(0.01)

>>> pulseV()
```

La méthode `duty()` de l'objet `pwm` ici créé admet un argument entier, entre 0 (qui correspond à un signal bas) et 1023 (qui correspond à un signal constant haut). La fréquence du signal rectangulaire généré est donné à l'initialisation de l'objet `pwm` et elle est ici fixée à 1000Hz.

Vous trouverez le projet Wokwi présenté ci-dessus à l'adresse https://wokwi.com/projects/347682562173305428

Exercice 1. Familiarisez-vous avec les fonctions du module `time` (en particulier `sleep`, `sleep_ms` ou `sleep_us`, mais aussi `ticks_us`, `ticks_diff`)

Ecrire alors une fonction d'entête `def metronome(pulsations)` qui admet en argument un entier et qui fait clignoter la led rouge à raison de `pulsations` par minutes (et ce sans fin, on l'arrêtera éventuellement avec ctrl-C)

Exercice 2. Reprendre le code source pour supprimer la modulation en largeur d'impulsion de la led verte (supprimer les lignes 6 et 7 relatives à PWM) et écrire une fonction d'entête `def deuxPourTrois(pulsations)` qui fait clignoter les leds rouges et vertes à la fréquence `2*pulsations` par minute pour la led rouge, et `3*pulsations` par minute pour la led verte (en faisant en sorte qu'elles s'allument en même temps `pulsations` fois par minute.