Programmation et Robotique

Les cartes Programmables ?

Une carte programmable est une carte sur laquelle est intégré des composants électronique dont un ou plusieurs microcontrôleurs. Un microcontrôleur permet de contrôler la carte avec un programme informatique. Ce programme peut être écris en divers langage de programmation.

Sans programme la carte programmable ne peut pas fonctionner. Vous devez donc relier votre carte à un ordinateur pour y injecter un programme.

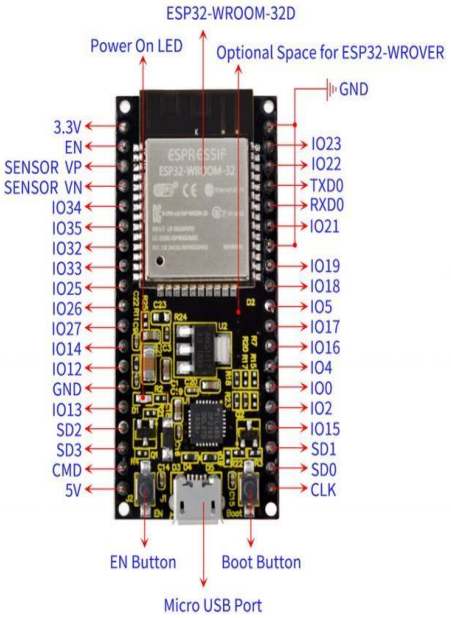
Une fois le programme dans la carte, vous pouvez l'utiliser en toute autonomie sans ordinateur.

Exemples des cartes Programmables :

Présentation de la carte ESP32

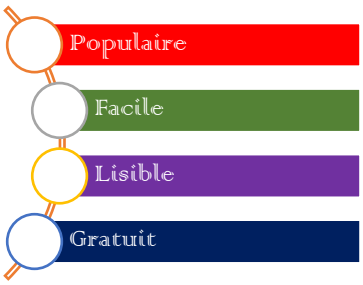
L'ESP32, comme toutes les cartes à microcontrôleur, permet de piloter un système de manière interactive à partir du programme que l'on aura défini et mis dans sa mémoire. Elle possède en effet une connectivité assez complète.

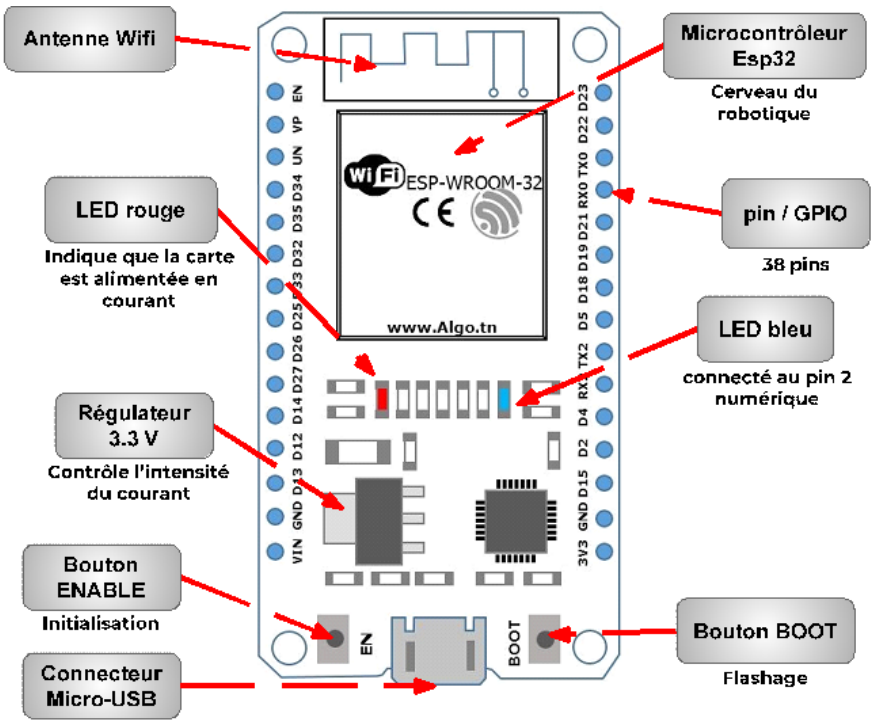
L'ESP32 développé par la société Espressif, est une carte de développement à faible coût dédié à l'internet des objets (IoT) et les applications embarquées. C'est un (SoC) system on a chip doté de communications sans fil Wifi et Bluetooth.



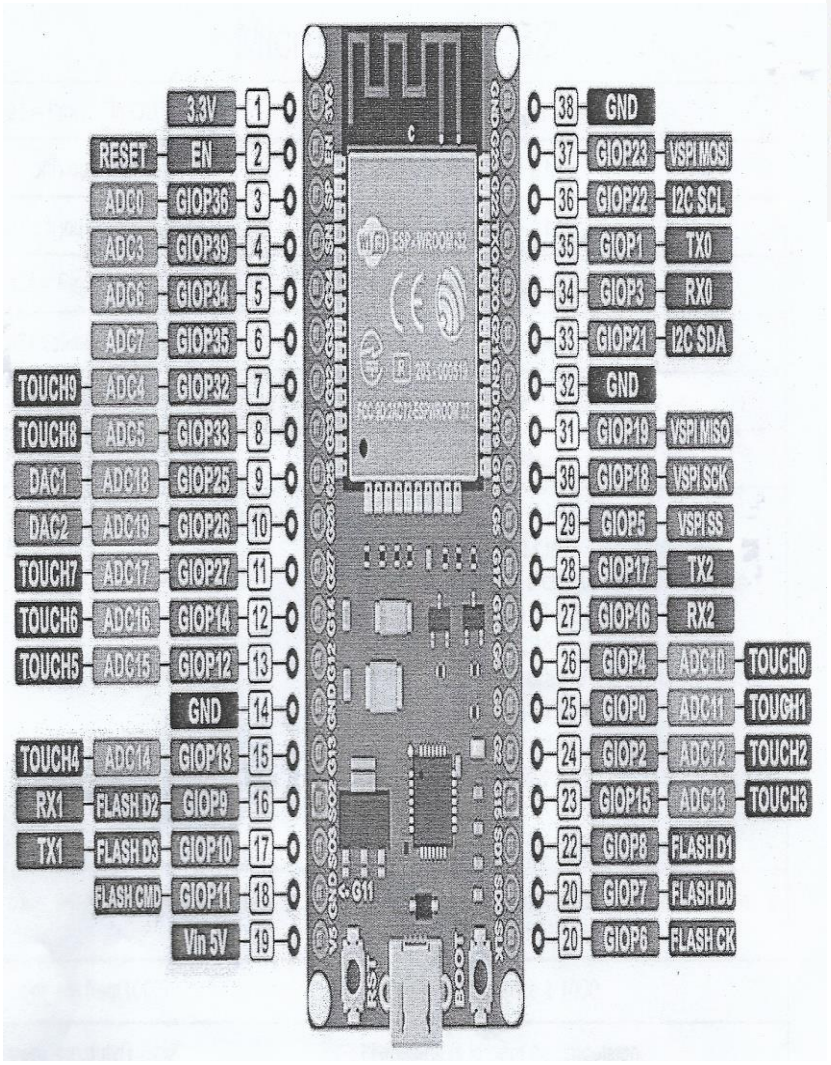
Quelques langages de de programmations

Pourquoi on a choisi Python :

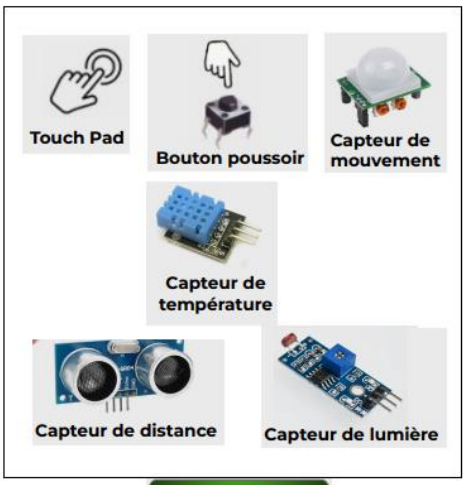




Les composants de la carte ESP32



Les Pins de la carte ESP32



Activité 1 : Clignoter une diode LED

Objectif :

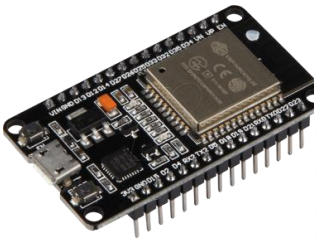
Dans cette activité on se propose de piloter une LED avec une carte ESP32 et de la faire clignoter à une vitesse d'un clignotement par seconde.

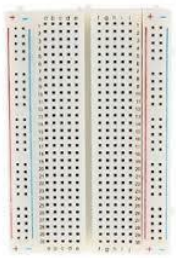
On aura besoin de…


LED


Résistance


Fils électrique


Carte ESP32

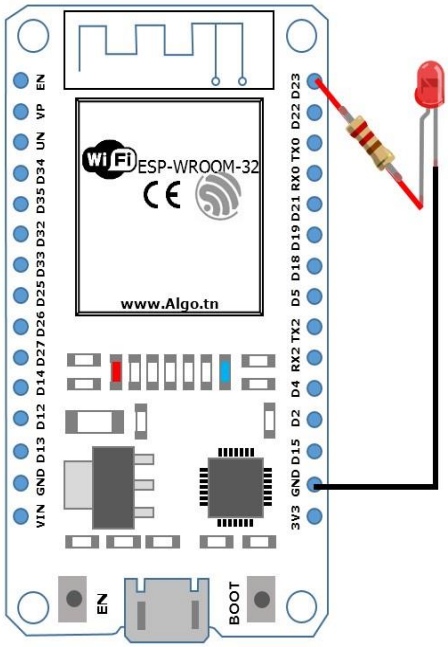

Plaquette d'essai

Qu'est-ce qu'une LED ?

Une LED est une diode qui émet de la lumière. Comme toute diode elle comporte une anode (borne +) qui est la patte la plus longue et une cathode (borne -) qui est la patte la plus courte.

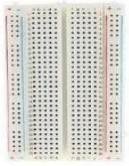
Pour qu'une LED s'allume il faut obligatoirement relier sa patte - à la borne - de l'alimentation et sa patte + à la borne + de l'alimentation. Si la LED est branchée à l'envers elle ne s'allumera pas (le courant ne la traversera pas).

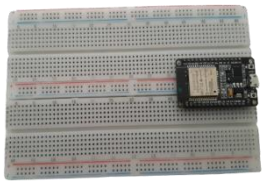
De plus elle ne doit pas être traversée par un courant trop fort, c'est pour cela qu'il est indispensable de brancher une résistance en série avec la LED. On notera qu'une résistance n'a pas de sens de branchement.



I- Se préparer

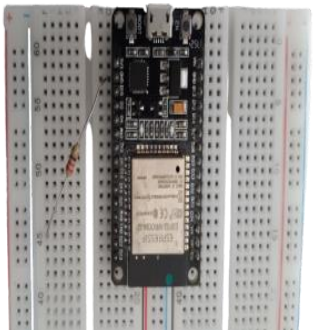
1- Brancher la carte ESP32 à a la plaquette d'essai.





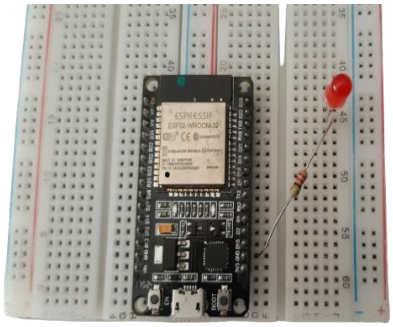
II- Allumer une LED

2- Brancher la résistance au Pin GND de la carte ESP32

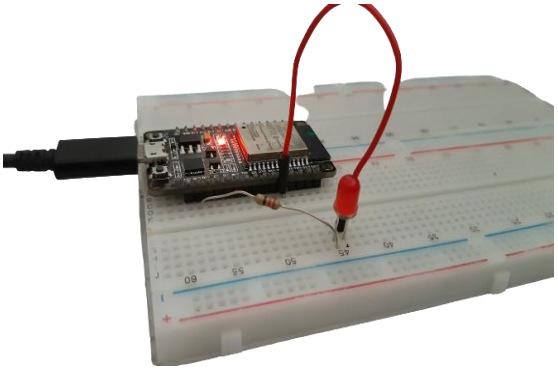


La carte ESP32 possède de nombreux PIN différents. Chacun d'entre eux est étiqueté et sert à connecter différentes pièces.

3- Brancher la LED a la résistance :



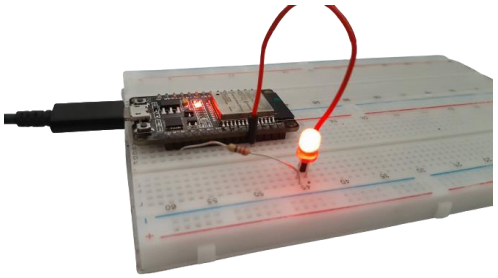
4- Relier la LED à la carte ESP32 par le fil au PIN23



Pour que l'ordinateur puisse envoyer des instructions à la carte ESP32, on doit les connecter avec un câble USB.

5- Relier la carte ESP32 au PC par le câble USB

Coté PC



Coté Carte ESP32

III- Programmer l'allumage d'une LED

1. *Importer la classe PIN de la bibliothèque machine*

……

L'objet PIN est utiliser pour contrôler les entrées / sorties (I/O)

2. *Configurer le PIN 23 en sortie :*

……

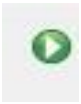
3. *Allumer la LED :*

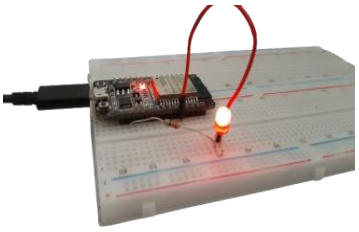
……

Ou bien

……

4. *Tester l'allumage :*

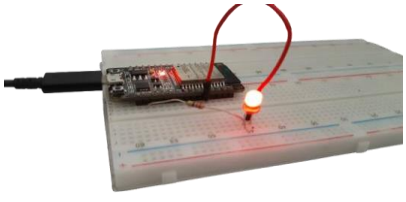
Pour tester le programme clique sur 

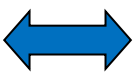


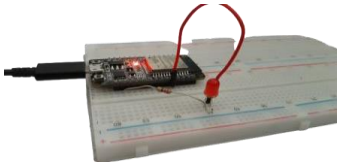
5. *Eteindre la LED :*

……

6. *Tester le programme :*







L'exécution du programme est très rapide la LED s'allume et s'éteint très vite !!!

7. *Ajouter un temps après chaque état (une seconde) :*

……..

8. *Importer la classe sleep de la bibliothèque time :*

……..

9. *Faire clignoter la LED :*

……..
……..
……..
……..
……..
……..
……..

10. *Programme final*

……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..
……..

10. *Explications du programme*

Le programme qui permet de piloter le montage est le suivant :

```
from machine import Pin # importation de la classe Pin de la bibliothèque “machine”

from time import sleep # importation de la classe sleep de bibliothèque “time”

led = Pin(23, Pin.OUT) # régler la pin 23 comme sortie

while True: # répétez indéfiniment

    led.value(1) # Allumer La diode LED

ou bien : led.on( )

    sleep(1)  #Patientez 1 seconde

    led.value(0) #éteindre la diode LED

ou bien  : led.off( )

    sleep(1) #Patientez 1 seconde
```

`Pin(23,Pin.OUT)` règle la Pin numérique numéro 23 de la carte ESP32 en mode sortie `Pin.OUT`, c'est à dire qu'il sera possible de programmer sur cette borne (`nommée led`) l'absence (0V) ou la présence (+5V) de courant.

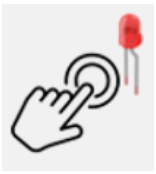
L’instruction `while True` permet d’exécuter le bloc d’instructions en dessous indéfiniment en boucle. La séquence de commandes utilisées est la suivante :

- `led.value(1)` : du courant est envoyé sur la borne `led`(associer à la Pin23), du coup la LED est alimentée et elle s'allume,
- L’instruction `sleep(1)` fait « patienter » le programme une seconde, durant ce temps la LED reste allumée,
- La commande `led.value(0)` provoque l'absence de courant sur la borne `led`(associer à la Pin23) : la LED s'éteint,
- Le programme patiente encore 1 s, pendant ce temps la LED est éteinte.

Comme précisé plus haut, `while True` permet d’exécuter le bloc d’instructions indéfiniment en boucle. Dès que toutes les instructions du bloc en dessous de while ont été exécutées, elles s'exécutent de nouveau. Ainsi le programme provoque le clignotement de la LED avec une fréquence d’une seconde.

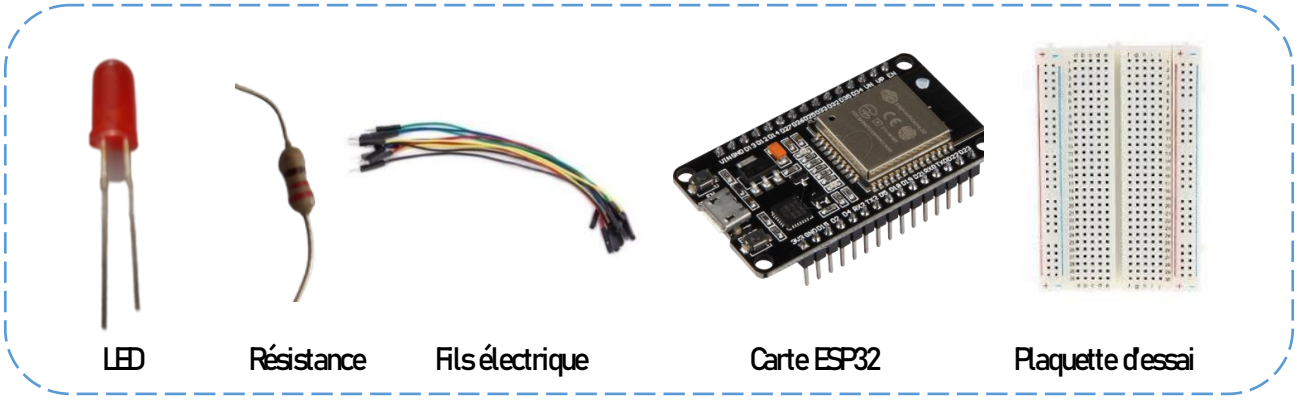
Activité 2 : Utilisation des Pins tactiles (TouchPad)

Objectif :



Dans cette activité on se propose d'allumer une diode LED par le toucher du pin GPIO4.

On aura besoin de…



Identification des broches tactiles

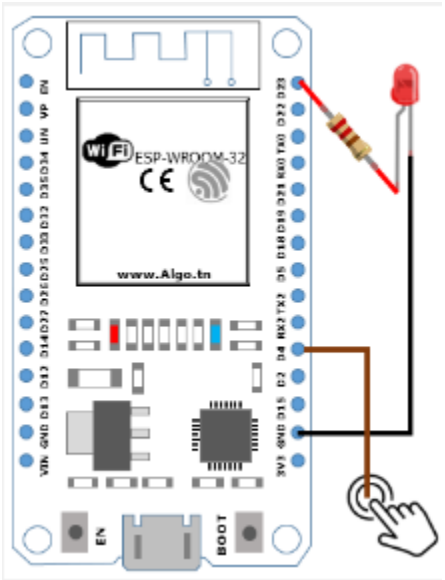
Les broches qui peuvent être utilisées pour détecter un contact avec le doigt sont:

GPIO 0 (si disponible), GPIO 2, GPIO 4, GPIO 12, GPIO 13, GPIO 14, GPIO 15, GPIO 27, GPIO 32 et GPIO 33.

Programme :

```
from machine import Pin,TouchPad

led=Pin(23,Pin.OUT)  # régler la pin 23 comme sortie
touch=TouchPad(Pin(4))  # régler la touchpin à la pin 4
seuil=100 #configurer le seuil pour lequel le pin est considéré comme touché
while True:
    touch.config(seuil)
    if touch.read( )<100: # vérifier si la valeur sur le touchpin < seuil
        led.on( )      # Allumer la lampe LED
    else:
        led.off( )     # Eteindre la lampe LED
```



Montage

Activité 3 : Allumer une lampe LED avec un bouton poussoir

Objectif :

Allumer une diode LED connectée au pin23 avec un bouton poussoir connecté au GPIO15.

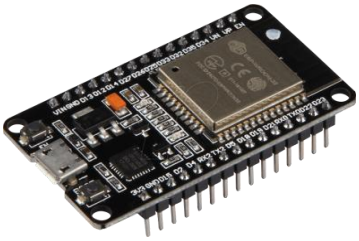
On aura besoin de…

LED

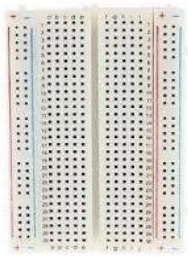
Résistance

Fils électrique

Bouton poussoir



Carte ESP32

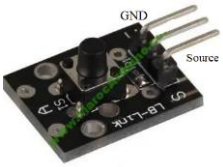


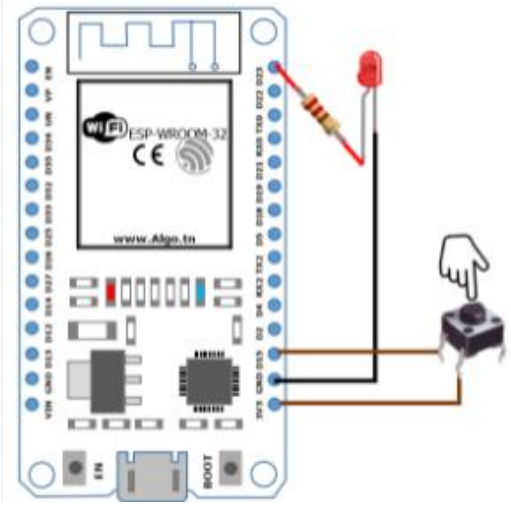
Plaquette d'essai

Programme :

```
from time import sleep
from machine import Pin

led=Pin(23,Pin.OUT)
p15=Pin(15,Pin.IN,Pin.PULL_UP)
while True:
    if p15.value( )==0:
        led.on( )
    else :
        led.off( )
```





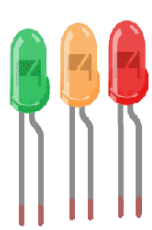
Montage

Activité 4 : Feux de circulation des voitures

Objectif :

Dans ce projet on va réaliser un feu de signalisation routière contrôlée par la carte ESP32 où la led Rouge s'allume pendant 3 secondes, puis s'éteint et la led Verte s'allume pendant 3 secondes et s'éteint à son tour pour que la led orange s'allume pendant 1 seconde. Puis le programme repart au début et recommence.

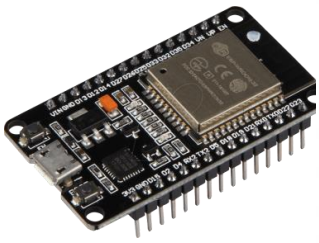
On aura besoin de…



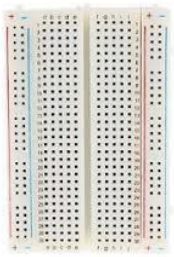
3 LEDs

3 Résistances

Fils électrique



Carte ESP32

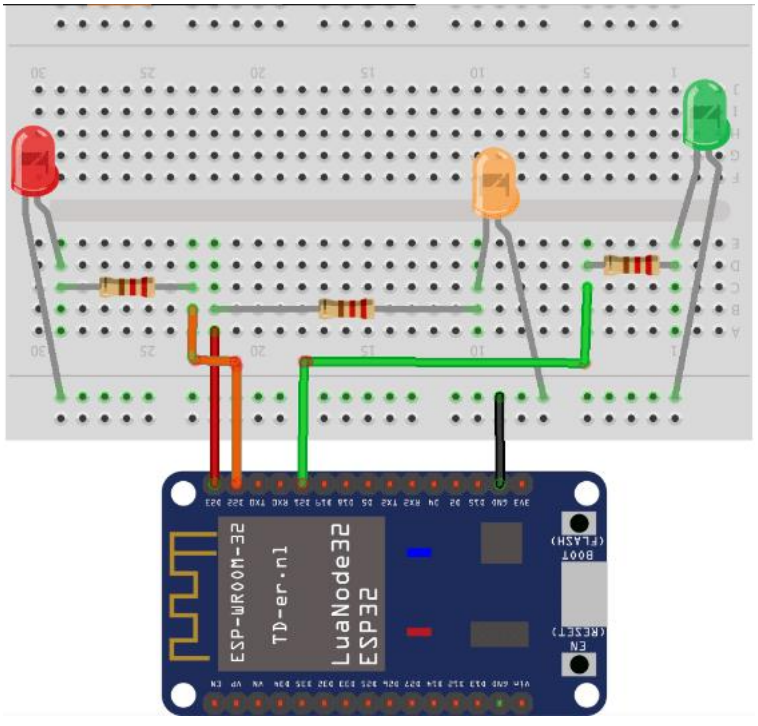


Plaquette d'essai

Programme :

```
from time import sleep
from machine import Pin
led_jaune=Pin(23,Pin.OUT) # Règle la broche D23 de la carte ESP32 en mode sortie
led_rouge=Pin(22,Pin.OUT) # Règle la broche D22 de la carte ESP32 en mode sortie
led_verte=Pin(21,Pin.OUT) # Règle la broche D23 de la carte ESP32 en mode sortie

while True:
    led_jaune.value(1) # Allumer LED jaune
    led_rouge.value(0)
    led_verte.value(0)
    sleep(1) # Attendre 1s
    led_jaune.value(0) # Éteindre LED jaune
    led_rouge.value(1) # Allumer LED rouge
    led_verte.value(0) # Éteindre LED verte
    sleep(3) # Attendre 3s
    led_jaune.value(0) # Éteindre LED jaune
    led_rouge.value(0) # Éteindre LED rouge
    led_verte.value(1) # Allumer LED verte
    sleep(3) # Attendre 3s
```



Montage

Activité 5 : Contrôler un buzzer par la carte ESP32

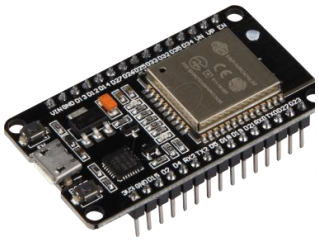
Objectif :

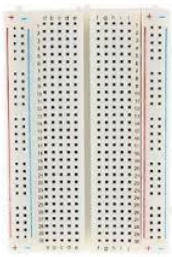
Le but de cette activité est de faire sonner un buzzer avec la carte ESP32.

On aura besoin de…


Buzzer

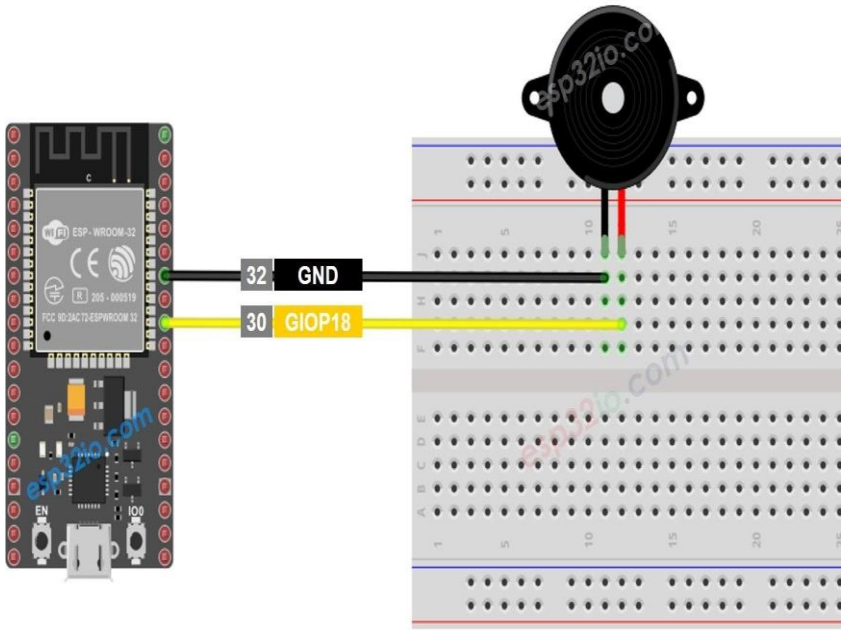

Fils électrique


Carte ESP32


Plaquette d'essai

Montage

Le montage consiste à relier la borne (+) du buzzer à la broche D18 et la borne (-) à la GND de la carte ESP32.



Programme :

```
From time import sleep
from machine import Pin

buzzer=Pin(18,Pin.OUT) # Règle la broche D18 de la carte ESP32 en mode sortie

while True:
    buzzer.value(1) #le buzzer sonne
    sleep(2) # Attendre 2s
    buzzer.value(0) # le buzzer arrête de sonner
    sleep(2) # Attendre 2s
```

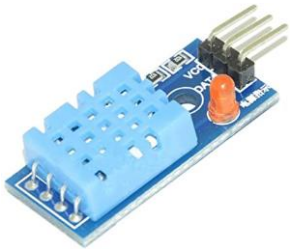
Activité 6 : Capteur de température et d'humidité

Capteur DHT11

Objectif :

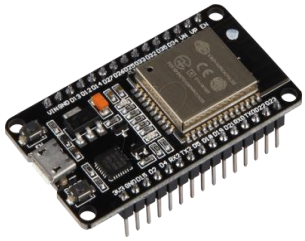
Afficher les lectures de température et d'humidité du capteur DHT11.

On aura besoin de…

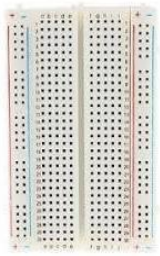


Capteur DHT11

Fils électrique



Carte ESP32

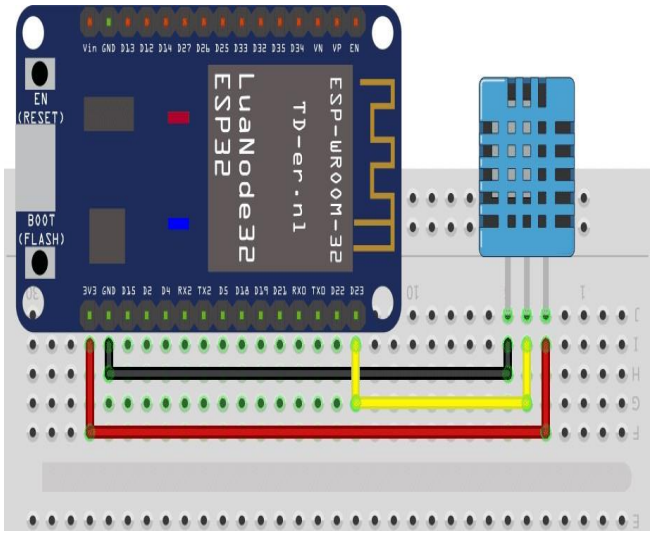


Plaquette d'essai

Montage

Pour réaliser le montage, on connecte:

- la broche DATA à la broche D23 de la carte ESP32
- la broche VCC à la broche 3.3V de la carte ESP32
- la broche GND à la broche GND de la carte ESP32



Programme :

```
from machine import Pin
from time import sleep
import dht
cap=Pin(23, Pin.IN)
d=dht.DHT11(cap)
while True:
    d.measure( )      # mesure de la température et de l'humidité
    t=d.temperature( ) # lecture de la température et l'affectation de sa valeur dans un objet t
    h=d.humidity( )  # lecture de l'humidité et l'affectation de sa valeur dans un objet h
    print("Temperature=", t, "C", "Humidité=", h, "%") #affichage de température et d' humidité
    sleep(1)  #patienter 1 seconde
```

Activité 7 : Contrôler un servomoteur avec la carte ESP32

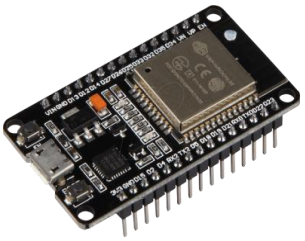
Objectif :

Le but de cette activité est de faire tourner le servomoteur entre 0° et 90°.

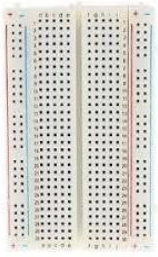
On aura besoin de…

Servomoteur

Fils électrique



Carte ESP32

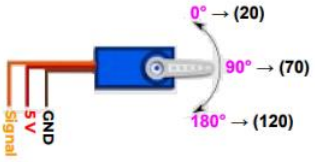


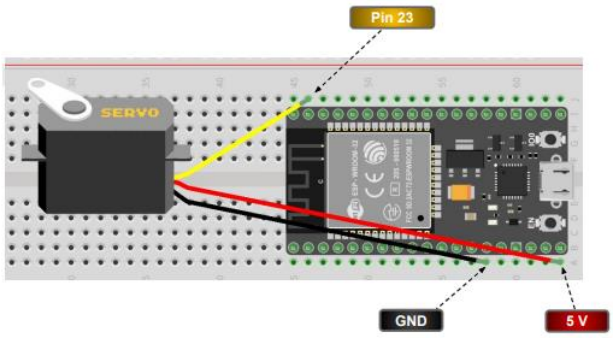
Plaquette d'essai

Montage

Le branchement est presque toujours le même et se fera à l'aide de 3 fils dont voici les principales couleurs que vous pourrez rencontrer :

- Rouge : fil de l'alimentation à relier à la broche 5V de la carte ESP32
- Noir : fil à relier à la broche GND de la carte ESP32
- Jaune : fil de signal de positionnement branché à la broche D23 de la carte ESP32





Programme :

```
from time import sleep
from machine import Pin,PWM   #PWM (Pulse Width Modulation) pour le fonctionnement du servo moteur
servo = PWM(Pin(23),freq=50)  #Définir Pin23 pour le servomoteur avec 50 Hz de PWM
while True:
    servo.duty(20)   #Tourner le servomoteur à 0°
    sleep(2)
    servo.duty(70)   #Tourner le servomoteur à 90°
    sleep(2)
```

Activité 8 : Capteur de distance

Objectif :

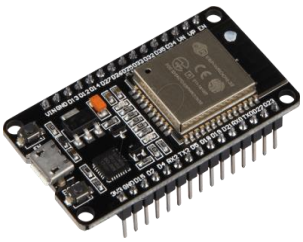
Le but de cette activité est de faire allumer la LED intégré (Pin2) avec un capteur de distance.

Si le capteur détecte un obstacle < 10 cm : la Led s'allume, sinon la Led s'éteint

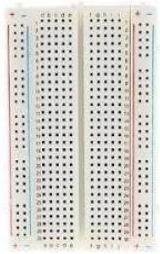
On aura besoin de…

Capteur de distance

Fils électrique



Carte ESP32

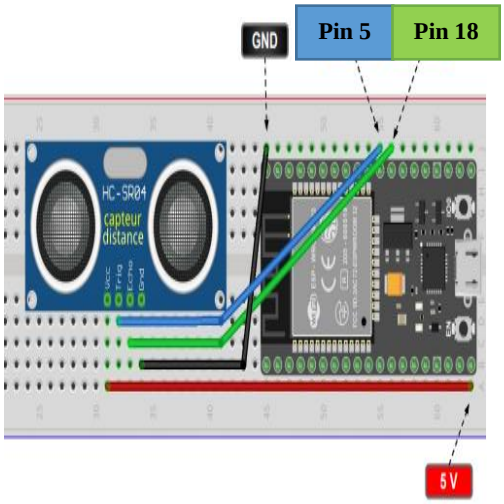


Plaquette d'essai

Montage

Pour réaliser le montage, on connecte:

- L'alimentation 5V de la carte ESP32 va sur la broche VCC du capteur.
- La broche GND de la carte ESP32 va sur la broche GND du capteur.
- La broche D5 de la carte ESP32 va sur la broche TRIGGER du capteur.
- La broche D18 de la carte ESP32 va sur la broche ECHO du capteur.



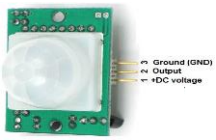
Programme :

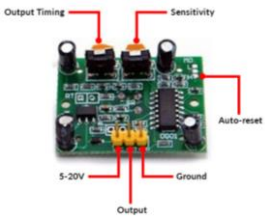
```
from machine import Pin
from time import sleep
from hcsr04 import HCSR04   # Le fichier hcsr04.py doit être chargé dans la carte ESP32
led=Pin(2,Pin.OUT)
sensor = HCSR04(trigger_pin=5 , echo_pin=18 , echo_timeout_us=10000)   # Configuration du capteur hc-sr04
while True :
    distance = sensor.distance_cm( )   # Calculer la distance en cm et la récupérer dans une variable distance
    if distance < 40 :
        led.on( )
    else :
        led.off( )
    sleep(0.5)   # Une pause puis reboucler
```

Si la distance < 10 cm la Led s'allume sinon la Led s'éteint

Remarque: On doit aussi téléverser la Bibliothèque hcsr04.py (l'enregistrer sur la carte ESP32) téléchargée à partir https://raw.githubusercontent.com/rsc1975/micropython-hcsr04/master/hcsr04.py (clic droit sur le lien puis enregistrer le lien sous)

Activité 9 : Capteur de mouvement PIR





Objectif :

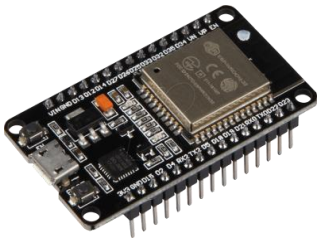
Le but de cette activité est de manipuler un capteur de mouvement PIR. Chaque fois qu'un mouvement est détecté, nous allumons une LED pendant 20 secondes.

On aura besoin de…

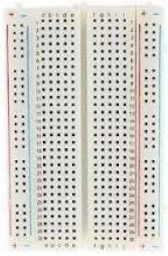
LED

Capteur PIR

Fils électrique



Carte ESP32

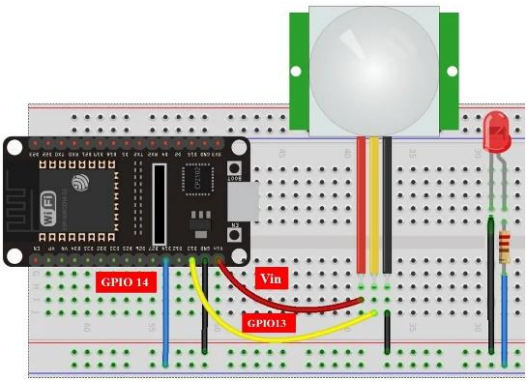


Plaquette d'essai

Montage

Pour réaliser le montage, on connecte:

- L'alimentation 5V (ou 3.3V suivant le modelé du capteur) de la carte ESP32 va sur la broche VCC du capteur.
- La broche GND de la carte ESP32 va sur la broche GND du capteur.
- La broche D13 de la carte ESP32 va sur la broche Output du capteur.
- La broche D14 de la carte ESP32 va sur la broche anode(+) de la diode led et la résistance.



Vous pouvez utiliser tous les GPIO comme interruptions, sauf GPIO6 à GPIO11.

Programme :

```
from machine import Pin
from time import sleep

motion = False

def handle_interrupt(pin):
    global motion
    motion = True
    global interrupt_pin
    interrupt_pin = pin

led = Pin(14, Pin.OUT)
pir = Pin(13, Pin.IN)
pir.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt)
while True:
    if motion:
        print('Motion detected! Interrupt caused by:', interrupt_pin)
        led.value(1)
        sleep(20)
        led.value(0)
        print('Motion stopped!')
        motion = False
```