

## Termux, Termux:API & Node-RED

Termux.....	2
Installation du paquet termux.....	2
Termux:API.....	3
Installation du paquet termux-api.....	3
Implémentations actuelles de l'API.....	3
!!! Activer les autorisations des API !!!.....	3
Node-RED sur Android.....	4
Installation via Termux.....	4
Visualiser l'éditeur de Node-RED sur un PC.....	4
Récupérer l'IP du SmartPhone.....	4
Ouvrir le navigateur d'un PC connecté au même réseau WiFi du SmartPhone.....	4
Taper le lien : <a href="http://IP_Android:1880">http://IP_Android:1880</a> .....	4
Cas pratiques :.....	5
1) Gérer la charge d'un Smartphone S10e.....	5
Comment activer le contrôle de la charge.....	5
2) Envoyer un SMS sur son smartphone en cas de panne de courant à son domicile.....	6
Dans Node-RED, installer la palette « node-red-contrib-termux-api ».....	6
Limite à l'utilisation de la propriété « msg.payload.plugged » et du contrôle de charge d'un smartphone !!.....	7
Utilisation de la propriété « Wifi Info » indépendante de la valeur du courant lors d'un contrôle de charge.....	8
Récupérer l'adresse IP du smartphone.....	8
Utilisation de « exec » & « termux-sms-send ».....	9
Utilisation termux-send-sms.....	9
Utilisation conjointe de « Wifi Info » et « termux-send-sms » pour détecter une panne de courant et envoyer un seul SMS.....	10
3) Envoyer un SMS à son smartphone lors d'une panne de courant et lors du rétablissement de celui-ci.....	11
Utilisation d'un fichier contenant IP smartphone (détecté ttes les 5 minutes) pour envoyer un seul SMS en cas de panne et un SMS lors du rétablissement du courant.....	11
!!!Remarque importante à propos de « termux-wake-(un)lock » !!!.....	12
Documentation.....	12

# Termux

## Sources

- <https://termux.dev/en/>
- [https://wiki.termux.com/wiki/Main\\_Page](https://wiki.termux.com/wiki/Main_Page)

Termux est un émulateur de terminal Android et une application d'environnement Linux qui fonctionne directement, sans les droits root. Un système de base minimal est installé automatiquement, des paquets supplémentaires sont disponibles en utilisant le gestionnaire de paquets.

## Installation du paquet termux

- Télécharger le module complémentaire Termux depuis F-Droid

Termux possède quelques fonctionnalités supplémentaires. Vous pouvez les ajouter en installant des modules complémentaires :



### [Termux:API](#)

Access Android.

### [Termux:Boot](#)

Run script(s) when your device boots.

.....

# Termux:API

## Installation du paquet termux-api

- Télécharger le module complémentaire Termux:API depuis F-Droid
- au départ de termux , utiliser la commande : `pkg install termux-api`

Important : Ne pas mélanger les installations de Termux et Addons entre Google Play et F-Droid.

## Implémentations actuelles de l'API

### [termux-battery-status](#)

Get the status of the device battery.

.....

### [termux-sensor](#)

Get information about types of sensors as well as live data.

.....

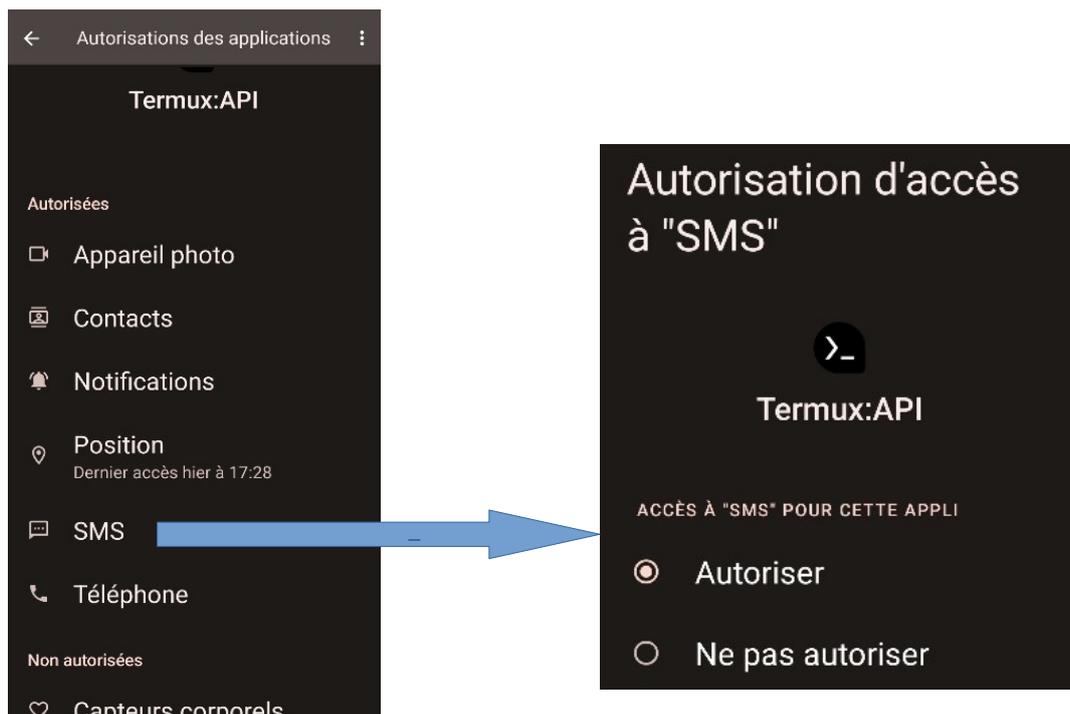
### [termux-sms-send](#)

Send a SMS message to the specified recipient number(s).

## !!! Activer les autorisations des API !!!

pour utiliser ces implémentations, **il est nécessaire d'activer les autorisations d'accès à SMS dans le smartphone !**

**Paramètres > Applications > Termux:API > Autorisations >**



# Node-RED sur Android

Source : <https://nodered.org/docs/getting-started/android>

## Installation via Termux

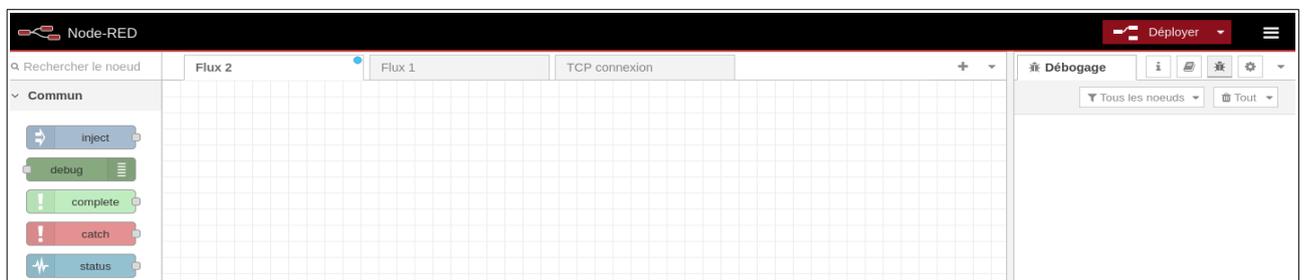
```
apt update
apt upgrade
apt install coreutils nano nodejs
npm i -g --unsafe-perm node-red
node-red
```

## Visualiser l'éditeur de Node-RED sur un PC

### Récupérer l'IP du SmartPhone

Ouvrir le navigateur d'un PC connecté au même réseau WiFi du SmartPhone

Taper le lien : [http://IP\\_Android:1880](http://IP_Android:1880)



## Cas pratiques :

### 1) Gérer la charge d'un Smartphone S10e

#### Comment activer le contrôle de la charge

Paramètres > batterie > contrôle de la charge

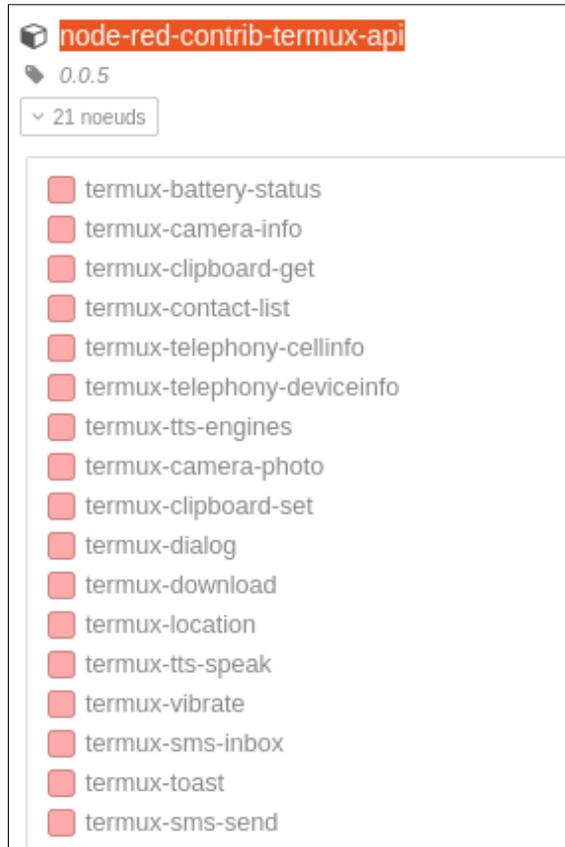
Au delà de ce seuil, le courant de charge mesuré à la sortie du chargeur est nul !

Le courant reste nul jusqu'à ce que le % restant descend à 69 %, ensuite le courant est rétabli !



## 2) Envoyer un SMS sur son smartphone en cas de panne de courant à son domicile

Dans Node-RED, installer la palette « node-red-contrib-termux-api »



**Limite à l'utilisation de la propriété « msg.payload.plugged » et du contrôle de charge d'un smartphone !!**

Dans Node-RED on observe :

- un courant de charge de 1148 mA à 79 %

```
msg.payload : Object
  ▼ object
    health: "GOOD"
    percentage: 79
    plugged: "PLUGGED_AC"
    status: "CHARGING"
    temperature: 29.799999237060547
    current: 1148
```

- la propriété msg.payload.plugged affiche « PLUGGED\_AC »

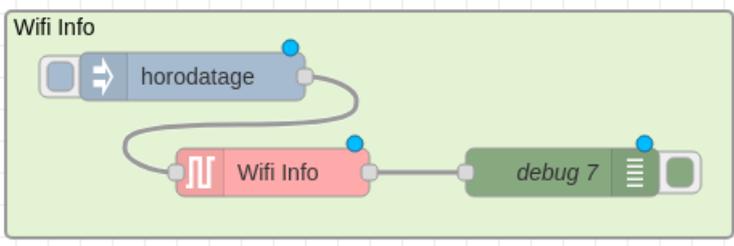
- un courant négatif -778 mA à 80 %

```
▼ object
  health: "GOOD"
  percentage: 80
  plugged: "UNPLUGGED"
  status: "DISCHARGING"
  temperature: 29.799999237060547
  current: -778
```

- la propriété msg.payload.plugged affiche « UNPLUGGED »

**La propriété « msg.payload.plugged » utilisée dans le flow Node-RED ne permet pas à elle seule de distinguer une panne de courant d'un dépassement de seuil !!**

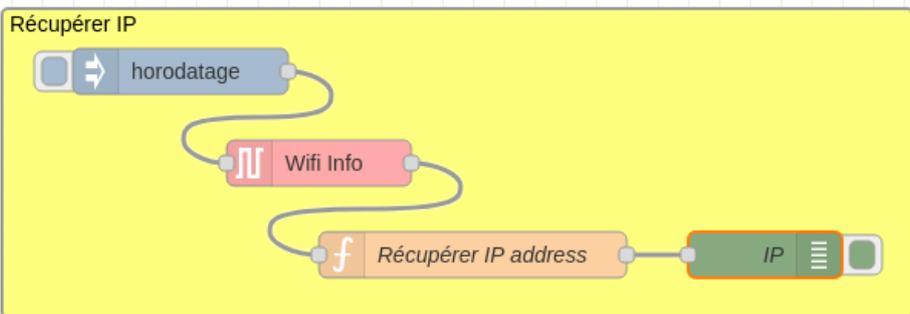
## Utilisation de la propriété « Wifi Info » indépendante de la valeur du courant lors d'un contrôle de charge



The flowchart, titled "Wifi Info", is contained within a light green rounded rectangle. It starts with a blue "horodatage" block, followed by a red "Wifi Info" block, and ends with a green "debug 7" block. All three blocks are connected in a sequence with blue lines.

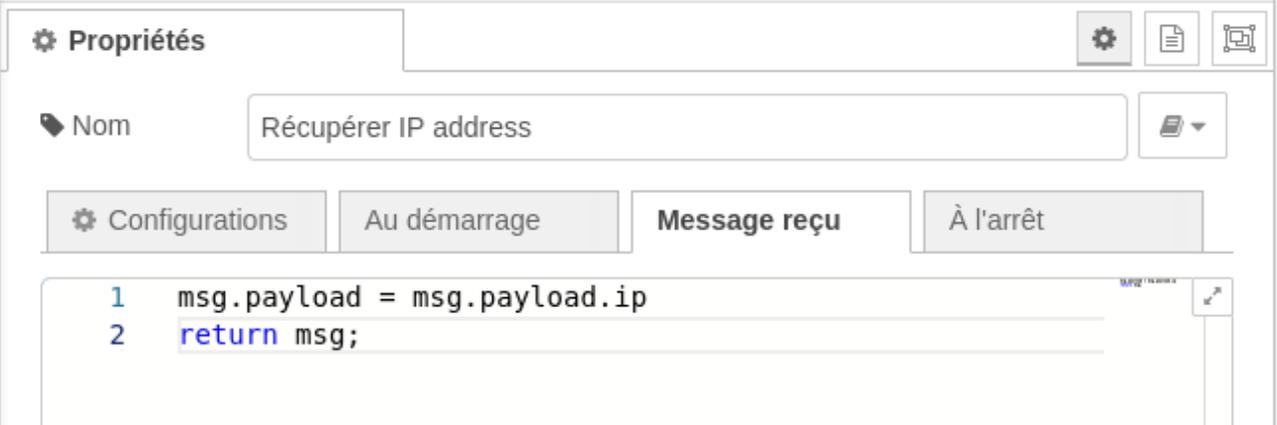
```
msg.payload : Object
  object
    bssid: "a6:91:b1:5f:7f:b1"
    frequency_mhz: 2412
    ip: "192.168.129.56"
    link_speed_mbps: 144
    mac_address: "02:00:00:00:00:00"
    network_id: 0
    rssi: -45
    ssid: "Proximus-Home-2810"
    ssid_hidden: false
    supplicant_state: "COMPLETED"
```

## Récupérer l'adresse IP du smartphone



The flowchart, titled "Récupérer IP", is contained within a yellow rounded rectangle. It starts with a blue "horodatage" block, followed by a red "Wifi Info" block, and ends with an orange "Récupérer IP address" block and a green "IP" block. All four blocks are connected in a sequence with blue lines.

```
28/01/2025 19:06:26 noeud: IP
msg.payload : string[14]
"192.168.129.56"
```



The "Propriétés" window for the "Récupérer IP address" node. It shows the node name, configuration tabs, and a code editor with the following code:

```
1 msg.payload = msg.payload.ip
2 return msg;
```

## Utilisation de « exec » & « termux-sms-send »

### termux-sms-send

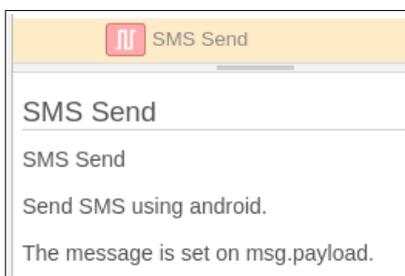
Send a SMS message to the specified recipient number(s).

#### Usage

```
termux-sms-send -n number[,number2,number3,...][text]
```

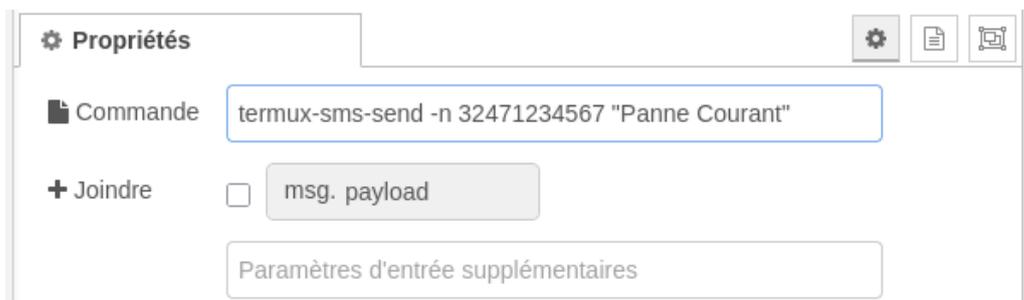
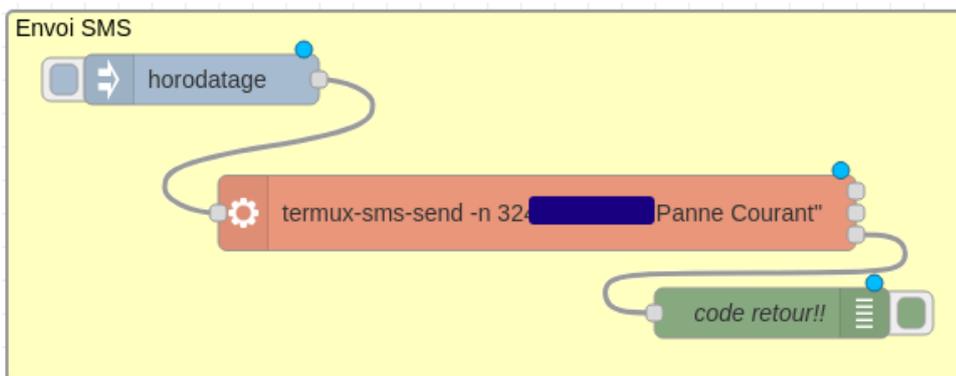
The text to send is either supplied as arguments or read from stdin if no arguments are given.

Pourquoi utiliser « termux-sms-send » de Termux-API, au lieu de « sms-send » de la palette «node-red-contrib-termux-api» ??

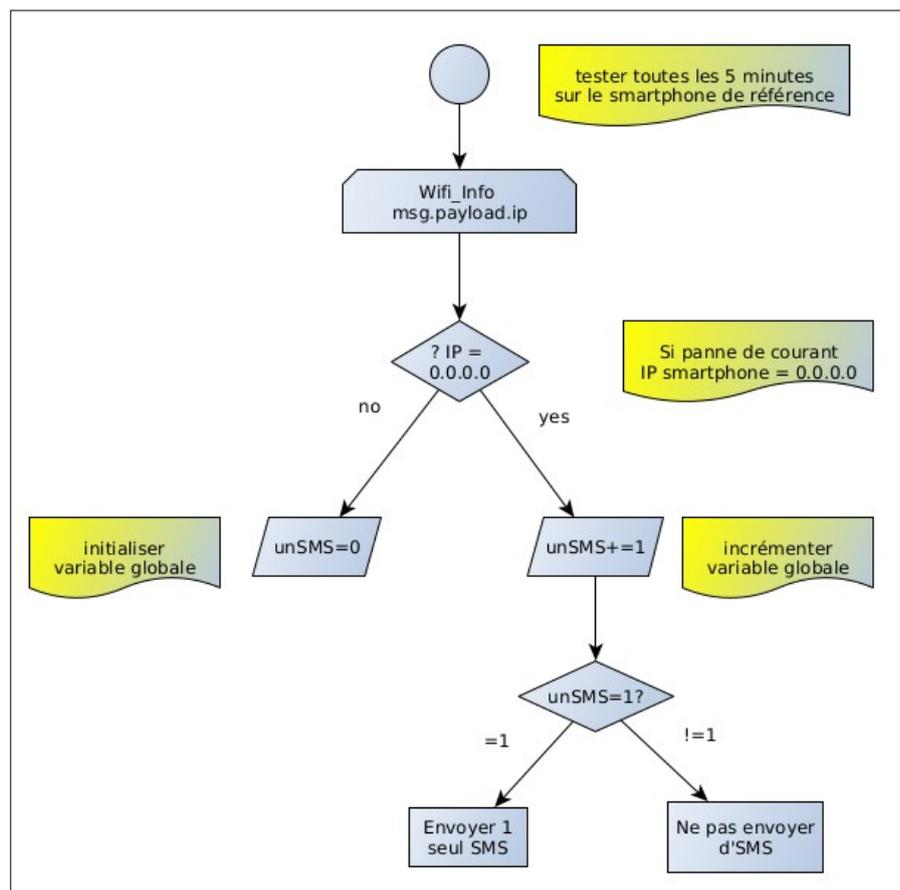
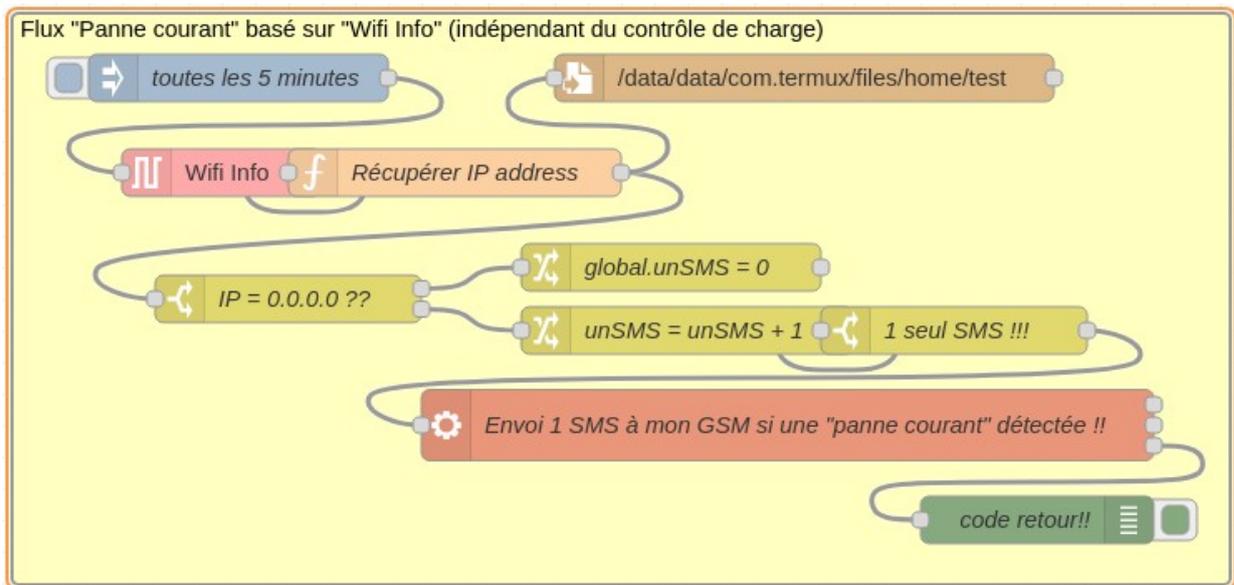


La documentation de « sms-send » ne précise pas comment l'utiliser....

## Utilisation termux-send-sms



## Utilisation conjointe de « Wifi Info » et « termux-send-sms » pour détecter une panne de courant et envoyer un seul SMS



### Modifier le nœud switch

The screenshot shows the configuration window for a switch node. It has a 'Propriétés' tab. The 'Nom' field contains 'IP = 0.0.0.0 ??'. The 'Propriété' dropdown is set to 'msg. payload'. There are two rule entries: the first is '!= 0.0.0.0' with a count of 1, and the second is '= 0.0.0.0' with a count of 2.

### Modifier le nœud change

The screenshot shows the configuration window for a change node. It has a 'Propriétés' tab. The 'Nom' field contains 'global.unSMS = 0'. The 'Règles' section has one rule: 'Définir global.unSMS sur la valeur 0'.

### Modifier le nœud change

The screenshot shows the configuration window for a change node. The 'Nom' field contains 'unSMS = unSMS + 1'. The 'Règles' section has three rules: 1) 'Définir msg. payload sur la valeur global.unSMS' with a 'Copie profonde de la valeur' checkbox. 2) 'Définir global.unSMS sur la valeur J: msg.payload + 1'. 3) 'Définir msg. payload sur la valeur global.unSMS' with a 'Copie profonde de la valeur' checkbox.

### Modifier le nœud switch

The screenshot shows the configuration window for a switch node. The 'Nom' field contains '1 seul SMS si 1 fois 0 !!!'. The 'Propriété' dropdown is set to 'msg. payload'. There is one rule entry: '= 1' with a count of 1.

The screenshot shows the configuration window for an exec node. It has buttons for 'Supprimer', 'Annuler', and 'Terminer'. The 'Propriétés' tab is active. The 'Commande' field contains 'termux-sms-send -n 32 [redacted] "Panne Courant"'. The 'Joindre' checkbox is unchecked, with 'msg. payload' as the value. The 'Sortie' dropdown is set to 'lorsque la commande est terminée - mode exec'. The 'Temps mort' field is set to 'Facultat' seconds. The 'Masquer la console' checkbox is unchecked. The 'Nom' field contains 'Envoi 1 SMS à mon GSM si une "panne courant" détectée !'.

### 3) Envoyer un SMS à son smartphone lors d'une panne de courant et lors du rétablissement de celui-ci

**Utilisation d'un fichier alimenté ttes les 5 minutes par l' IP smartphone pour envoyer un seul SMS en cas de panne et un SMS lors du rétablissement du courant**

Via script bash créé dans le smartphone

....

# ne faire ce test que si le fichier « test » existe et qu'il contient >=2 lignes

d=\$(tail -n1 fic\_ip | cut -d'.' -f4) # recupere derniere ligne du fichier fic\_ip et la derniere colonne

ad=\$(tail -n2 fic\_ip | head -n1 | cut -d'.' -f4) recupere avant derniere ligne du fichier fic\_ip et la derniere colonne

echo \$(( \$ad - \$d )) # calcule la différence

si le résultat est positif ⇒ panne courant

si résultat nul ⇒ ne rien faire

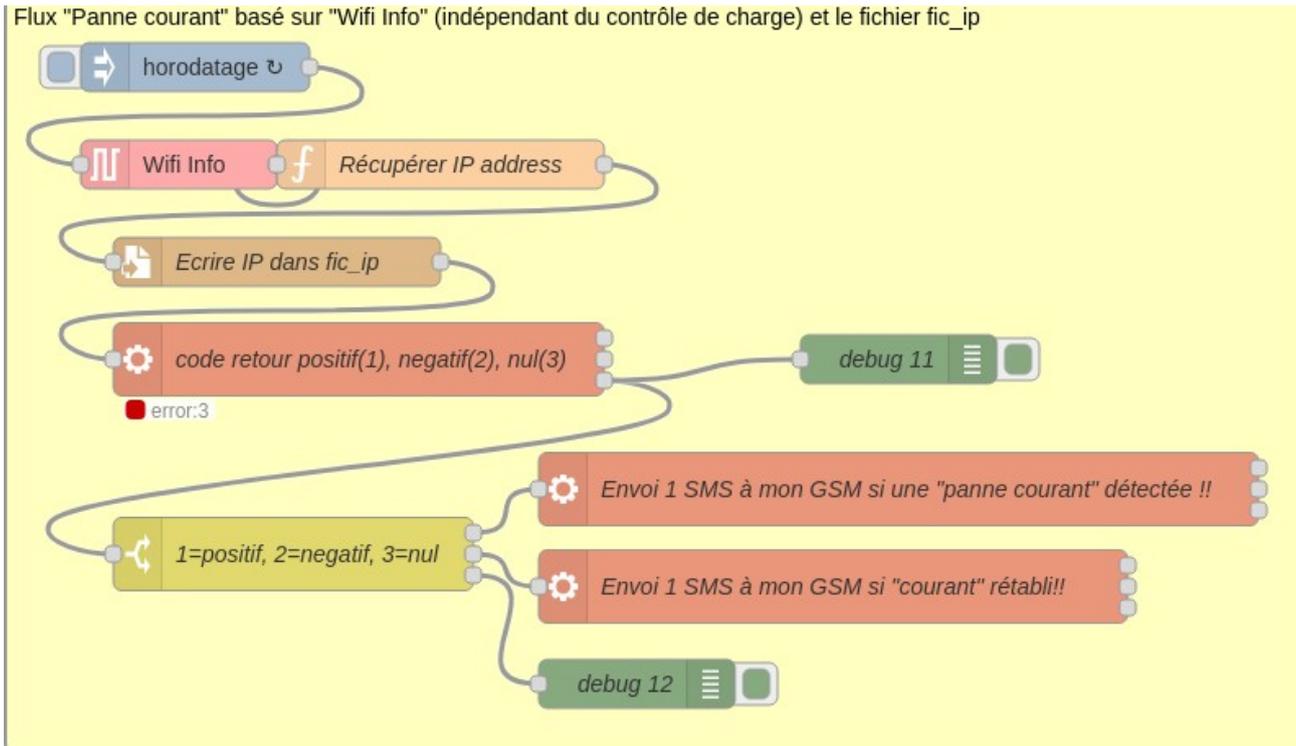
si résultat négatif ⇒ courant rétabli

script EnvoiSMS.sh avec argument fic\_ip ⇒ ./EnvoiSMS fic\_ip

```
#!/bin/bash
if [[ -f "$1" && $(cat "$1"|wc -l) -gt 1 ]]
then
    d=$(tail -n1 "$1" | cut -d'.' -f4)
    ad=$(tail -n2 "$1" | head -n1 | cut -d'.' -f4)

    if [[ $((ad - d)) -gt 0 ]]
    then
        #echo positif
        exit 1
    elif [[ $((ad - d)) -lt 0 ]]
    then
        #echo negatif
        exit 2
    else
        #echo nul
        exit 3
    fi
else
    echo "pas de fichier"
fi
```

Flux "Panne courant" basé sur "Wifi Info" (indépendant du contrôle de charge) et le fichier fic\_ip



**Modifier le noeud Écrire le fichier**

Supprimer Annuler Terminer

**Propriétés**

Nom du fichier:

Action:

Ajouter une nouvelle ligne (\n) à chaque charge ?

Créer un répertoire s'il n'existe pas ?

Encodage:

Nom:

Astuce : Le nom du fichier doit être un chemin absolu, sinon il sera relatif au répertoire de travail du processus Node-RED.

**Modifier le noeud exec**

Supprimer Annuler Terminer

**Propriétés**

Commande:

Joindre:  msg. payload

Sortie:

Temps mort:  secondes

Masquer la console:

Nom:

## **!!!Remarque importante à propos de « termux-wake-(un)lock » !!!**

« termux-wake-lock » doit être utilisé pour empêcher Android de se mettre en veille et permettre ainsi un fonctionnement continu de vos flows Node-RED !

La commande « termux-wake-lock » sera donc lancée dans termux avant la commande « node-red » !

## **Documentation**

<https://termuxtools.com/termux-commands-list-for-basic/>