Manuel d’opérations

Référence : pi4x4-operations.A.1.0.1
Date : 21/02/19

Table des matières

ISystème d’exploitation et paramétrage....4
I.1/boot/config :....4
I.2Paquetages supplémentaires....4
I.2.1Gestion de la luminosité de l’écran....4
I.2.2Clavier virtuel....4
I.2.3Multi-touch....4
IIUtilitaires de gestion de la batterie et du bouton poussoir....5
II.1Fonctionnement de la gestion de batterie....6
II.2Affichage du niveau de la batterie dans l’interface graphique....7
II.3Bouton poussoir....8
IIIConfiguration de l’horloge temps réel....8
III.1Fonctionnement du GPS....9
III.2Paramétrage pour l’IMU, température, pression....10

I Système d'exploitation et paramétrage

Le fonctionnement de la tablette Pi4x4 et des utilitaires décrits à la suite ont été testés sous le système d'exploitation **Raspian Stretch** et l'interface graphique **PIXEL**.

L'installation du système est classique. Quelques paramètres doivent être réglés ensuite et quelques paquetages supplémentaires doivent être installés (éventuellement optionnellement).

I.1 Fichier /boot/config.txt

enable_uart=1

Activation de la ligne série. Ceci est utilisé lors de la mise sous tension du Raspberry. Ce paramètre peut être positionné avec la commande **raspi-config**. Il est important pour que la tablette puisse démarré (sans ça, quand on appui sur le bouton, la Raspberry s'allume puis s'éteint presque immédiatement).

start_x=1

lcd_rotate=2

Retournement de l'éran à 180° (sinon l'écra est à l'envers).

dtoverlay=i2c-rtc,ds3231

Pour l'horloge temps réel.

I.2 Paquetages supplémentaires

I.2.1 Gestion de la luminosité de l'écran

apt install rpi-backlight

I.2.2 Clavier virtuel

apt install xvkdb

I.2.3 Multi-touch

Le multi-touch est géré par l'utilitaire **twofing** (http://plippo.de/p/twofing)

Pour l'installer :

- **apt-get update && apt-get install build-essential libx11-dev libxi-dev x11proto-randr-dev libxrandr-dev libxtst-dev xserver-xorg-input-evdev**

- **get** http://plippo.de/dwl/twofing/twofing-0.1.2.tar.gz (vérifiez la version en cours sur le site)
- **tar -xvzf twofing-0.1.2.tar.gz**
- **cd twofing-0.1.2**
- **make && sudo make install**
- Créez le fichier de règles **/etc/udev/rules.d/70-touchscreen-raspberrypi.rules**

  et ajoutez :

  KERNEL=="event*",ATTRS{name}=="FT5406 memory based driver",SYMLINK+="twofingtouch",MODE="0440"
- Modifiez la confid X11 dans le fichier **/usr/share/X11/xorg.conf.d/40-libinput.conf**
  Ajoutez les lignes suivantes à la fin :

```
Section "InputClass"
  Identifier "calibration"
  Driver "evdev"
  MatchProduct "FT5406 memory based driver"
  Option "EmulateThirdButton" "1"
  Option "EmulateThirdButtonTimeout" "750"
  Option "EmulateThirdButtonMoveThreshold" "30"
```

- Calibration **twofinger -debug** (sortir par ctrl-C)
- Pour que twofing démarre automatiquement lors de l'ouverture de session ajoutez **@/usr/bin/twofing** dans le fichier **.config/lxsession/LXDE-pi/autostart** dans le home de l'utilisateur.

II Utilitaires de gestion de la batterie et du bouton poussoir

- Les fichiers suivants sont nécessaires pour mettre en œuvre la gestion de la batterie et du bouton poussoir. Ils doivent être copiés dans le répertoire **/usr/local/pi4x4**.

- power_management.service
- power_management.py
- push_button.py
- bat_status.py
- battery-100.png

- battety-80.png
- battery-60.png
- battery-40.png
- battery-20.png
- plugged.png
- power_management_config.sample

- Copiez **power_management.service** dans **/lib/systemd/system/power_management.service**
- **systemctl enable power_management**
- Le package **rpi-backlight** doit être installé (**apt install rpi-backlight**)

II.1 Fonctionnement de la gestion de batterie

Lorsque le système démarre le service **power_management.service** est démarré et lance le script **power_management.py**. Celui-ci lance à son tour le script **push_button.py** et se met en boucle de lecture des valeurs de tension de la batterie que lui envoi le circuit POOM. La dernière valeur est enregistrée dans le fichier **/dev/shm/power_management_status** (en mémoire).

Le script **power_management.py** utilise des valeurs par défaut pour son paramétrage. Ces valeurs peuvent être modifiées grâce au fichier de configuration **/etc/power_management_config** dont un exemple se trouve dans le fichier **/usr/local/pi4x4/power_management_config.sample**.

Le script **power_management.py** est une version adaptée du script pi_power.py du projet pi_power (https://github.com/craic/pi_power)

Paramètres	Signification	Valeur par défaut
battery_min_voltage	Définit la tension minimale de la batterie c'est-à-dire la tension en-dessous de laquelle la puissance n'est plus suffisante et la machine se plante. Cette valeur correspond à 0 % de charge	3.0
battery_max_voltage	Définit la tension maximale réelle. Normalement une batterie 18650 complètement chargée à une tension de 4,2V. Cependant la tension réellement constatée est inférieure. Cette valeur correspond à 100 % de charge.	4.05
battery_max_voltage_with_power	Cette valeur correspond à la tension 100 % lorsque l'alimentation électrique est branchée.	4.2
shutdown_when_low_level	Yes ou no. Indique si un shutdown doit	yes

	être fait quand la valeur fraction_battery_min est atteinte	
fraction_battery_min	Pourcentage de batterie à partir duquel un shutdown automatique est lancé si shutdown_when_low_level est à yes.	0.04
shutdown_delay	Nombre de secondes d'attente avant de faire le shutdown.	60
log_battery_level	Yes ou no. Les informations de charge de la batterie sont enregistrées dans le fichier indiqué dans le paramètre power_management_log_path	no
power_management_log_path	Chemin du fichier dans lequel les informations de charge sont enregistrées	/dev/shm/power_management.log.csv

Format des fichiers power_management_status et power_management.log.csv.

Ces fichiers ont le même format CSV. Dans **power_management_status** il n'y a qu'une seule ligne correspondant aux dernières valeurs enregistrées. Dans **power_management.log.csv** il y a les mêmes informations mais de façon cumulées et datées.

Le format de ces fichiers est le suivant :

- Date/heure (uniquement dans **power_management.log.csv)**
- nombre de secondes depuis le boot (par intervalle de 30 secondes)
- tension de la batterie
- tension de la source électrique lorsque la tablette est alimentée par le secteur
- Fraction de puissance restante
- Source d'alimentation (batterie ou secteur)
-

II.2 Affichage du niveau de la batterie dans l'interface graphique

Cette affichage est géré par le script **/usr/local/pi4x4/bat_status.py**. Il est lancé au démarrage de l'interface graphique de l'utilisateur. Il affiche en haut à droite de l'écran une icône représentant le niveau de charge ou le branchement sur le secteur.

Pour cela la ligne suivante doit être rajouté dans le fichier **.config/lxsession/LXDE-pi/autostart** dans le home de l'utilisateur :

@/usr/bin/python /usr/local/pi4x4/bat_status.py &

Ce script lit le fichier **/dev/shm/power_management_status** pour connaître l'état de la batterie. Si la charge restante passe en-dessous de 16% la luminosité de l'écran est réduite à un tiers en passant la commande **rpi-backlight -b 85** .

Cette opération est réalisée qu'une seule fois, c'est-à-dire que l'utilisateur peut remonter le niveau de luminosité s'il le souhaite.

II.3 Bouton poussoir

Lorsque la machine est éteinte une simple pression sur le bouton poussoir allume la machine.

Pour arrêter la machine proprement une pression de 2 secondes est interprétée comme un ordre de shutdown.

C'est l'utilitaire **push_button.py**, lancé au démarrage par **power_management.py**, qui gère l'arrêt. Il doit donc toujours être actif. Il surveille la broche 17 qui est connectée sur le bouton-poussoir via le circuit POOM et le POWERBOOST. Lorsque le bouton-poussoir est pressé plus de 2 secondes l'utilitaire passe la commande shutdown.

III Configuration de l'horloge temps réel

(voir aussi : https://www.raspberrypi-spy.co.uk/2015/05/adding-a-ds3231-real-time-clock-to-the-raspberry-pi/)

Une fois le câblage du module réalisé :

1. **i2cdetect -y 1** pour vérifier que l'horloge est bien vue sur le bus I2C.
2. Dans **/etc/config.txt** ajouter :
   dtoverlay=i2c-rtc,ds3231
3. La première fois, mettre le système à l'heure manuellement :
   date -s « 18 april 2018 10:17:00 »
4. Puis enregistrer la date dans le module : **hwclock -w**
5. La commande **hwclock** doit afficher l'heure
6. Afin que le module soit pris en compte à chaque reboot et que l'horloge système soit mise à l'heure en fonction du contenu de la clock hardware, ajouter dans **rc.local** : **hwclock -s**

III.1 Fonctionnement du GPS

Activation du port série :

tapez la commande : **raspi-config**

1. Sélectionnez : **Interfacing Options**
2. Puis : **P6 Serial**
3. A la question : **Would you like a login shell to be accessible over serial?**
   Répondre **NON**
4. A la question : **Would you like the serial port hardware to be enabled?**
   Répondre **YES**.
5. Rebooter
6. Passez la commande **dmesg | grep tty** qui doit répondre :
   3f201000.serial: ttyAMA0 at MMIO 0x3f201000 (irq = 87, base_baud = 0) is a PL011 rev2
   [0.933824] 3f215040.serial: ttyS0 at MMIO 0x0 (irq = 220, base_baud = 31250000) is a 16550
7. La commande **cat /dev/serial0** doit renvoyer des lignes du type :
   *(49) $GPRMC,012247.261,V,,,,,0.00,0.00,060180,,,N*45*
   *(37) $GPVTG,0.00,T,,M,0.00,N,0.00,K,N*32*
   *(39) $GPGGA,012248.261,,,,,0,0,,,M,,M,,*40*
   *(30) $GPGLL,,,,,,012248.261,V,N*72*
8. Installer les utilitaires GPS et le damon.
   sudo apt-get install gpsd-clients gpsd -y
   sudo systemctl enable gpsd.socket
9. Modifiez le fichier **/etc/default/gpsd** avec la ligne : **DEVICES=”/dev/serial0″**
10. rebooter
11. Tapez **gpsmon**.
    Suivant l’endroit où on est, le GPS mettra plus ou moins de temps pour faire un Fix (réception de 4 satellites). Un fois qu’un fix est obtenu la page de **gpsmon** indiquera les coordonnées.