UTILISATION DU BOITIER CDPI3 Version 4

LANCEUR AU BOOT

Pour pouvoir utiliser ce boitier en autonome sans aucunes liaisons ni écran,
il faut pouvoir choisir à la fin du boot un logiciel à lancer.
Il faut aussi une commande de ShutDown en Local

▰= Espace
Le chemin « /home/pi/Projets-Python » est à adapter.

1 Installer la librairie de la carte Piface2

sudo ▰ apt-get ▰ install ▰ python3-pifacedigitalio ▰ python-pifacedigitalio

2 Créer ce Fichier de service

sudo ▰nano ▰/lib/systemd/system/Lanceur.service

```
[Unit]
Description=Lanceur au Boot
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python3.4 ▰/home/pi/Projets-Python/Lanceur-10.py ▰&
WorkingDirectory=/home/pi/Projets-Python

[Install]
WantedBy=multi-user.target
```

3 Lui donner les droits

sudo ▰chmod ▰644 ▰/lib/systemd/system/Lanceur.service

4 Le démarrer au boot

sudo ▰systemctl ▰enable ▰Lanceur.service
sudo ▰reboot

5 Pour l'enlever du Boot

sudo ▰systemctl ▰mask ▰Lanceur.service

6 Pour le remettre

sudo ▰systemctl ▰unmask ▰Lanceur.service

Les Scripts définis dans le Lanceur doivent être exécutables

sudo ▰chmod ▰+x ▰chemin du script

A LA FIN DU BOOT

Si l'inverseur sur l'Entrée 7 est sur DISTANT (0) :
Affiche « A » sur la Matrice
Fermeture de la Piface
Si position LOCAL (1) :
Affichage d'un '?' sur la matrice

Bouton 0	Fermeture sans rien lancer	
Bouton 1	lance CaptureFlash-31.py	
Bouton 2	Lance SenseHat-20.py	
Bouton 3	Lance SenseHat-21.py	Idem avec le AM2302
Bouton 4	……	
Bouton 5	……	

MODULE HORLOGE RTC

Les fonctions d'horodatage ne peuvent fonctionner correctement
si le boitier est totalement autonome sans liaison internet.
Il faut alors un module horloge sur le RaspBerry (RTC-PI)

Utilise la carte HAT:RTC-PI-PLUS avec les 40 Pins du GPIO
L'adresse I2C 0x68 est compatible avec les capteurs de la SenseHAT

Installation sous Rasbian-Jessie

1 sudo ▰ nano ▰ /boot/config.txt
Ajouter à la fin :
dtoverlay=i2c-rtc,ds1307

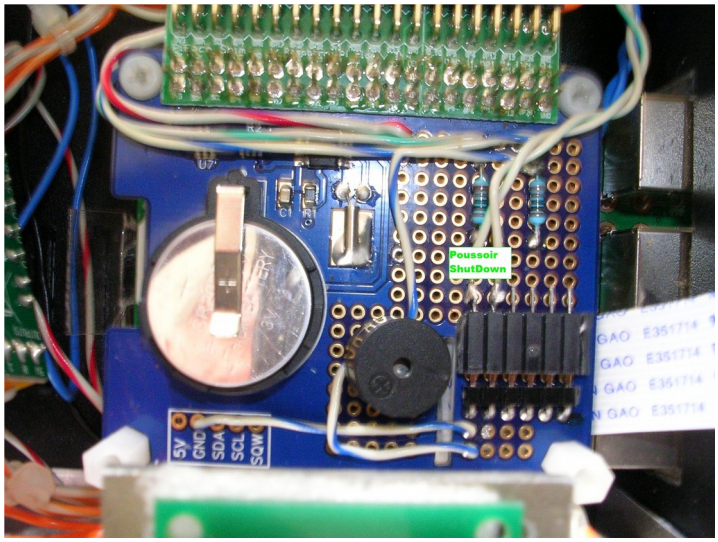
2 sudo ▰ nano ▰ /etc/modules
Ajouter à la fin :
rtc-ds1307

3 sudo ▰ nano ▰ /lib/udev/hwclock-set
Mettre un # au début des 3 lignes suivantes:

```
if [ -e /run/systemd/system ] ; then
    exit 0
fi
```

Ajouts sur la Carte

3 fils relient
3.3v (Broche 1) non utilisé
GPIO17 (Broche 11) relié au poussoir de ShutDown avec une 1K
GPIO18 (Broche 12) relié au Buzzer genre carte-mère PC en PWM à 10 %



Commandes

sudo hwclock
-r pour lire la RTC
-s Mise à l'heure System à partir RTC
-w mise à l'heure RTC à partir System

BOUTON de SHUTDOWN

Ce bouton arrette proprement le PI si appuyé pendant 3 secondes
On peut le Relancer avec le bouton RUN
ATTENTION Ne pas utiliser le RUN si le Raspi n'a pas été arretté corectement avant
Evite simplement de devoir couper l'alimentation du Raspi pour redémarrer
L'appui sur le bouton produit un son avec le Buzzer

Comme pour le Lanceur (Page 1), créer un fichier de service avec le Script ShutD.py ci-dessous

```
#!/usr/bin/env python3.4
# -*- coding: utf-8 -*-
#Lancé au Boot avec ShutD.service
#Bouton appuyé au moins 3 Secondes

# on importe les modules necessaires
import os
import time as time
import RPi.GPIO as GPIO

#GPIO-17 (Broche connecteur 11) pour le Poussoir
#GPIO-18 (Broche connecteur 12) pour le Buzzer
SD=17
Buz=18

# on met RPi.GPIO en mode notation BCM
GPIO.setmode(GPIO.BCM)

# on met le GPIO17 en Entrée et à UP (Broche connecteur 11)
GPIO.setup(SD, GPIO.IN, pull_up_down=GPIO.PUD_UP)
#On met le GPIO18 en sortie et PWM pour le Buzzer
GPIO.setup(Buz,GPIO.OUT)
Buzzer=GPIO.PWM(Buz,600)

def Shutdown():
        debut=time.time()
        intS=0
        maxwait=3
        print ("SHUTDOWN dans 3 secondes")
        while intS < maxwait and GPIO.input(SD) == GPIO.LOW:
                Buzzer.start(10)
                intS=time.time()-debut
                time.sleep(0.1)
        if intS >= maxwait and GPIO.input(SD) == GPIO.LOW :
                GPIO.cleanup()
                print ("SHUTDOWN")
                os.system('sudo halt')
                return 1
        else:
                Buzzer.stop()
                return 0

# on met le bouton en attente d'appui de 1 vers 0
GPIO.wait_for_edge(SD, GPIO.FALLING)
while True:
        if Shutdown():
                exit()
        else:
                #On relance
                Buzzer.stop()
                Buzzer=GPIO.PWM(Buz,600)
                GPIO.wait_for_edge(SD, GPIO.FALLING)
```

CAPTURE PHOTO-VIDEO

Modules necessaires
Le script CaptureFlash utilise picamera et MP4Box
pour convertir automatiquement les vidéos en MP4 à la fin de la capture.

sudo ▰ apt-get ▰ install ▰ gpac
sudo ▰ apt-get ▰ install ▰ python3-picamera

En mode DISTANT, les paramètres sont rentrés au clavier.
En mode LOCAL, utilisation d'un fichier de configuration prédéfinie CaptP.cfg
Avant de démarrer le RaspPI, positionner:

l'inter E6 sur Camera Normale ou IR et brancher la Caméra voulue.
l'inter E7 sur LOCAL ou DISTANT

Si l'Inter E7 sur Mode LOCAL(1):

Affiche « ? » sur la Matrice
Attente bouton sur le Boitier

Ou sur Mode Distant (0) :

Affiche « D » sur la Matrice et Menu DISTANT sur la console
pour ne pas devoir aller au boitier qui peut être loin !!

Menu Commande Locale (Boitier seul)

Utilise le fichier de Configuration /CaptP.cfg qui doit être créé avant en mode Distant (0)

Bouton 0	Fermer (ou CTRL+C)
Bouton 1	Mode Video
Bouton 2	Mode Photo
Bouton 3	Shot Photo avec attente de 5s
Bouton 4	Stream Vidéo
Bouton 5	Test du Flash ou IR selon Camera en cours

Menu Commande Distante (Avec clavier/écran ou Ethernet et VNC)

Touche 0	Fermer (ou CTRL+C)
Touche 1	Mode Vidéo
Touche 2	Mode Photo
Touche 3	Shot Photo avec attente de 5s
Touche 4	Stream Vidéo
Touche 5	Test du Flash ou IR selon Camera en cours
Touche 6	Configuration du mode Local (fichier CaptP.cfg)

Fichier de Configuration du Mode LOCAL CaptP.cfg

Dossier Principal	(Dossier)	
Nom Fichiers Photo	(fichierP)	
Durée totale de Prises (Mn)	(dureeP)	
Intervalle entre Prise (S)	(lapseP)	
Heure début Photo (HH:MM)	(heureP)	« 0 » pour lancement immédiat
Nom Fichiers Vidéo	(fichierV)	
Durée Vidéo (Mn)	(dureeV)	
Heure début Video (HH:MM)	(heureV)	« 0 » pour lancement immédiat
Enregistrement Vidéo (O/N)	(enrV)	

Pour chaque Champ, «ENTREE» seulement conserve la valeur

NOTA
Les Vidéos sont automatiquement mises en MP4, car trop de problèmes pour lire les H264

SENSHAT-20 et 21

Module necessaire

sudo ▰ apt-get ▰ install ▰ sense-hat

Utilisée uniquement en mode Local

Bouton 0	Arret
Bouton 1	Affiche Température + Baregraphe
Bouton 2	Affiche Les 3 Mesures T, H, P
Bouton 3	Affiche Température CPU + Capteur AM2302 pour la version 21
Bouton 4	Enregistrement dans fichier MesuSH---- des 3 mesures avec horodatage,

Affiche « W » entre 2 mesures ou « F » si Enregistrement validé (Bouton 4)

Utilise le fichier de configuration /ConfigSH.txt
modifiable en mode Texte sans jamais enlever les lignes #
Comme le capteur de température est enfermé dans le boitier, il faut corriger cette dernière

Exemple :

```
#Pas du Baregraphe   0 pour 0.5°  1 pour 1°
0
#Vitesse de défilement matrice
0.15
#Temps entre 2 cycles de mesures (secondes)
5
#Luminosité matrice 1 pour moitié
1
#chemin du fichier MesuSH
/media/pi/ClePI-16/Echange/
#Correction Température
-2.4
```

UTILISATION DU GPIO

Merci à Denis Bodor Hackable N° 17

LED verte ACT

Par défaut, indique l'activité sur la carte SD
On peut lui faire indiquer l'activité du Raspi, utile pour attendre un ShutDown en sécurité.

Sudo ▰ nano ▰ /boot/config.txt
Ajouter :
dtparam=act_led_trigger=heartbeat

LED d'Activité des 4 coeurs

Ajouter 5 LEDS gérées par le system

1 Créer ce Fichier « /home/pi/cpuleds.dts » avec l'éditeur de texte

```
/dts-v1/;
/plugin/;

/ {
   compatible = "brcm,bcm2708" , "brcm,bcm2709" , "brcm,bcm2710";

  fragment@0 {
    target = <&leds>;
    __overlay__ {
      ma_led0: maled0 {
        label = "maled0";
        gpios = <&gpio 26 0>;
        linux,default-trigger = "cpu0";
      };
      ma_led1: maled1 {
        label = "maled1";
        gpios = <&gpio 19 0>;
        linux,default-trigger = "cpu1";
      };
      ma_led2: maled2 {
        label = "maled2";
        gpios = <&gpio 13 0>;
        linux,default-trigger = "cpu2";
      };
      ma_led3: maled3 {
        label = "maled3";
        gpios = <&gpio 6 0>;
        linux,default-trigger = "cpu3";
      };
      ma_led4: maled4 {
        label = "maled4";
        gpios = <&gpio 21 0>;
        linux,default-trigger = "mmc0";
      };
    };
  };
};
```

2 Compiler cpuleds.dtbo qui sera utilisé dans les overlays

dtc▰@▰-I▰dts▰-O▰dtb▰-o▰/boot/overlays/cpuleds.dtbo▰cpuleds.dts

3 Déclarer cet overlays au boot

Ajouter « dtoverlay=cpuleds » dans le /boot/config.txt

ADRESSE IP STATIQUE SOUS RASBIAN-JESSIE

1 Lire la Configuration actuelle

sudo ifconfig
Noter inet addr 192.168.0.**xx**
Adresse de 010 à 050 pour les FreeBox

2 Lire les informations du Router de la box

sudo route -n
Noter le Gateway (passerelle) 192.168.0.**254** (pour les FreeBox)

3 Configuration

sudo nano /etc/network/interfaces

```
auto lo
iface lo inet loopback
iface eth0 inet static
address 192.168.0.xx
netmask 255.255.255.0
gateway 192.168.0.254
```

4 Re-Booter

sudo reboot

CAPTEUR AM2302

1 Charger la librairie C de Adafruit et la compiler

git clone https://github.com/adafruit/Adafruit_Python_DHT.git
sudo apt-get update
sudo apt-get install build-essential python-dev python-openssl
cd /home/pi/Adafruit_Python_DHT
sudo python setup.py install

2 Essais

les droits du dossier : /home/pi/Adafruit_Python_DHT/examples sont en root !!
Copiez ce dossier ailleurs pour pouvoir utiliser les examples avec Geany
/home/pi/Projets-Python/AM2302

cd /home/pi/Projets-Python/AM2302
Lancer sudo ./AdafruitDHT.py 2302 5 (utilisation du GPIO-5)

ATTENTION

Dans Geany, Commande Exécuter :
mettre python «%f » pour avoir la librairie Adafruit