Chorégraphe et Python - Domotique avec Pepper et Raspberry Pi

En cours de rédaction

[Mise à jour le 10/1/2019]

Niveau : avancé

Objectif

Contrôler les entrées, sorties logiques d'un Raspberry Pi “à la voix”.

Vidéo

Lien vers Youtube

Source

Pepper peut faire de la Domotique (Site PwavRobot)

> **Remarques préalables**
> Pour tenir compte du matériel disponible, l'état logique des sorties de la carte Raspberry Pi est simplement visualisé avec des LED. Le flux Chorégraphe de la source du projet a été adapté au matériel listé ci-dessous.

A. Préparation du Raspberry Pi

Liste du matériel

- Une carte Raspberry PI ou Raspberry PI ZERO
- Une alimentation 5V - 2,5A
- Une Card SD pour le système RASPBIAN
- 4 Led
- Raspberry Pi to Arduino Shield (optionnel)
- Sensor shield v2 Tinkerkit ou Module Grove Base Shield (optionnel)

Photo du montage à ajouter

Les connexions

L'utilisation d'un *Raspberry Pi to Arduino Shield* et d'un Sensor shield v2 Tinkerkit simplifie la mise en oeuvre puisqu'il suffit de supperposer les cartes comme sur la photo ci-dessus.

Tinkerkit → Arduino Shield → Raspberry Pi
O0 → 11 → GPIO10
O1 → 10 → GPIO8
O2 → 9 → GPIO22
O3 → 6 → GPIO4
O4 → 5 → GPIO25
O5 → 3 → GPIO23

Installation de Raspbian sur la carte µSD

Suivre la démarche proposé sur le site Raspberrypi.org.

Installation du serveur Python

- **Créer** un fichier **server.py** avec l'éditeur **Geany** et le sauvegarger dans **homme/pi/**.
- **Copier** le code ci-dessous dans le fichier server.py

server.py

```
#!/usr/bin/python
```

```
# coding: utf-8

import socket
import threading
import RPi.GPIO as GPIO
from time import sleep

def clear():
    print(' \n' * 80)

class ClientThread(threading.Thread):
    def __init__(self, ip, port, clientsocket):
        threading.Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.clientsocket = clientsocket
        print("[+] Nouveau thread pour %s %s" % (self.ip, self.port, ))

    def run(self):
        print("Connection de %s %s" % (self.ip, self.port, ))
        r = self.clientsocket.recv(2048)
        print("Instruction recue: " + r + "...")
        port, etat = r.split(",")
        print("GPIO:"+port + " / Etat:"+etat+" ")

# Envoi du signal HIGH sur la pate #4 GPIO pour coller le relai
# Le GPIO est sur ON
        if etat == 'HIGH':
            print "Off"
            GPIO.output(int(port), GPIO.HIGH)
            sleep(1)
            exit()

        if etat == "LOW":
            print "On"
            GPIO.output(int(port), GPIO.LOW)
            sleep(1)
            exit()

        print("Client déconnecté...")

tcpsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpsock.bind(("", 1111))

clear()
print "(c)2016 - Dr CADIC - Serveur de pilotage des GPIO sur Pi ou
PiZero avec le robot Pepper"
# Mise en mode adequat BCM et definition du numéro de port
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

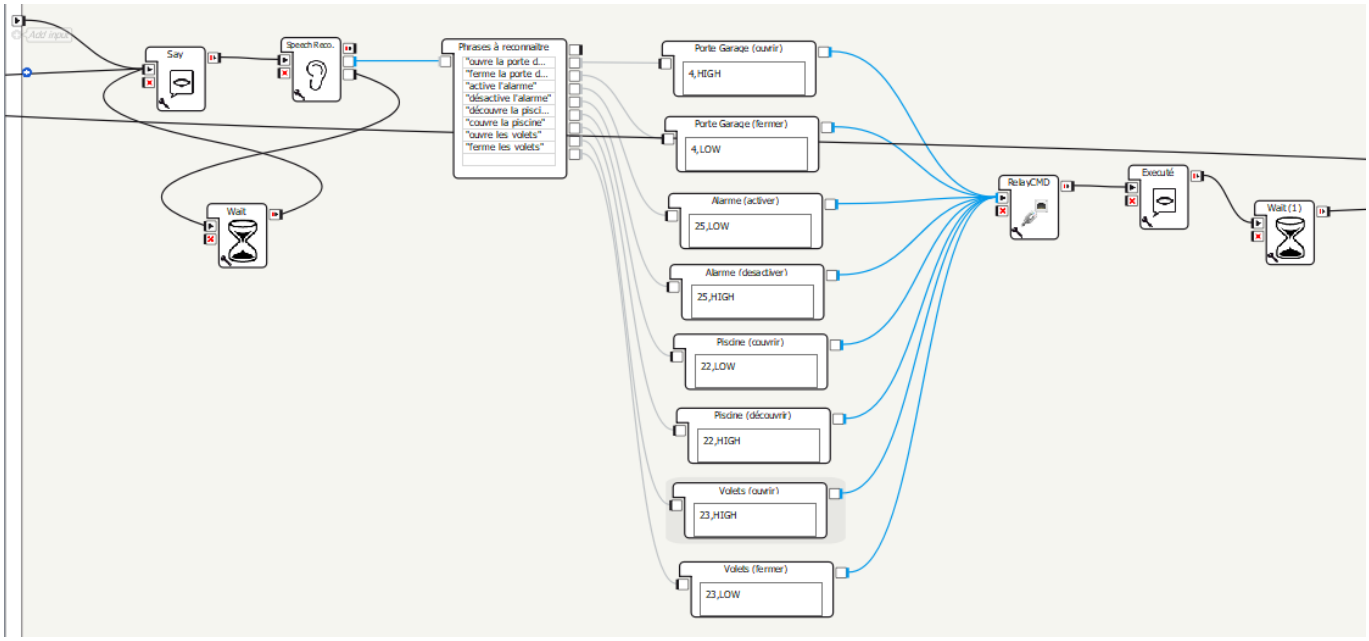
```
# Définit la sortie GPIO à activer
GPIO.setup(4, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)

while True:
    tcpsock.listen(10)
    print("En ecoute...")
    (clientsocket, (ip, port)) = tcpsock.accept()
    newthread = ClientThread(ip, port, clientsocket)
    newthread.start()
```

- **Ouvrir** une console et entrer : pi@Raspi2:~ $ **python relay_server.py**. La console doit afficher le message ci-dessous.

Le serveur se met en écoute des messages envoyés par Pepper.

B. Préparation de Pepper

Le flux Chorégraphe

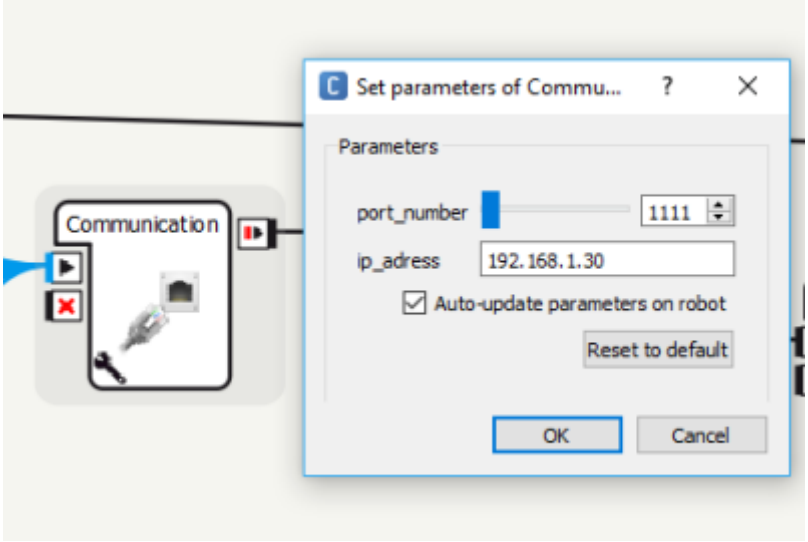


La boîte "Communication"

- **Paramétrage**

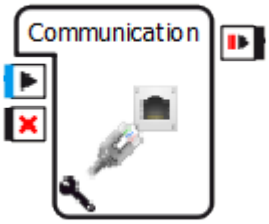
Pour que la communication s'établisse, il faut entrer l'adresse IP du Raspberry Pi (serveur) et le port sur lequel il écoute.

Exemple



- **Code**

La code contenu dans la boîte communication est listé ci-dessous.



client.py

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)


    def onLoad(self):
        #put initialization code here
        pass

    def onUnload(self):
        #put clean-up code here
        pass

    def onInput_onStart(self, p):
        #self.onStopped() #activate the output of the box
        # p est l'instruction à envoyer au serveur
        import socket

        # Recup des parametres de la boite
        ip_address=self.getParameter("ip_adress")
        port_number=self.getParameter("port_number")

        # Connexion au serveur avec IP et port
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((ip_address, port_number))
        s.send(p)
        r = s.recv(9999999)
        self.onStopped()
        pass

    def onInput_onStop(self):
        self.onUnload() #it is recommended to reuse the clean-up as the
box is stopped
        self.onStopped() #activate the output of the box
```

Télécharger

Le code source de l'exemple **Domotique** est téléchargeable A venir.