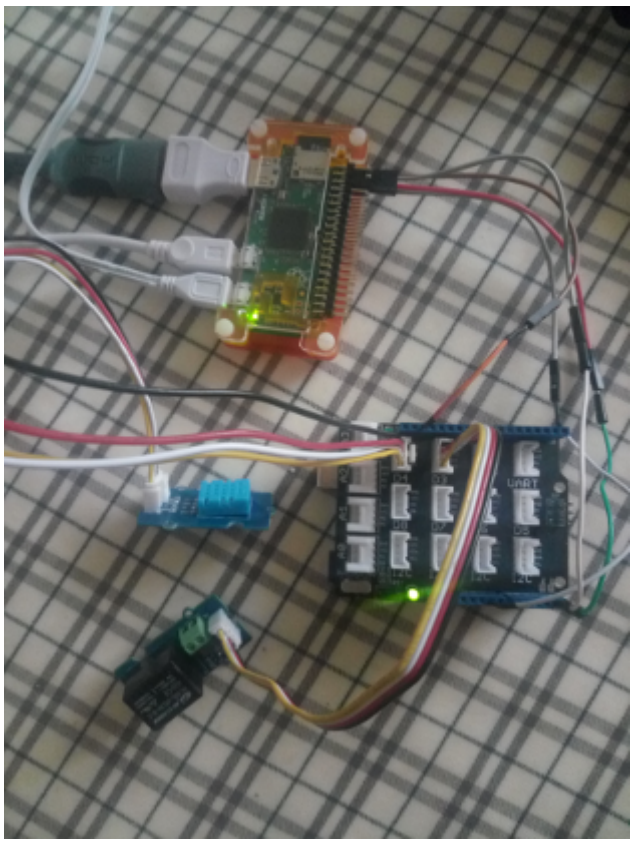
Vous n'avez pas de GrovePi+, dans les entrailles du GrovePi

Voici une solution avec un kit RaspBerry Pi 0 et une carte Arduino munie d'un shield Grove.



Il faut récupérer le fichier C arduino de la grove Pi et le téleverser dans l'arduino.

```
cd /home/pi
git clone https://github.com/DexterInd/GrovePi.git
cd GrovePi/Firmware/Source/v1.2/grove_pi_v1_2_7/
```

Dans ce dossier il faut copier toutes les librairies(*.cpp *.h) dans un dossier Grovepi dans votre espace perso Arduino.

Ensuite ouvrir le fichier grove_pi_v1_2_7.ino et le téleverser dans l'arduino.

Côté câblage:

```
Raspberry  Arduino
GPIO2(SDA) A4 (SDA)
GPIO3(SCL) A5 (SCL)
5V         Vref
GND        GND
```

Pour les inconditionnels des connecteurs Grove, il doit être possible de brancher directement le raspberry sur l'Arduino au travers d'un connecteur Grove sur le port I2C.

Si votre raspberry a été paramétré pour fonctionner avec Grovepi la carte est maintenant reconnue.

```
i2cdetect -y 1
```

Vous affichera l'adresse de la carte Arduino (0x04).

Tout le travail réalisé avec Mqtt/HomeAssistant/Freeboard/NodeRed fonctionnera tout aussi bien avec votre carte UNO.

Le code python suivant [!! montré simplement à des fins de pédagogie !!] vous montre comment fonctionne le protocole avec la carte UNO en communicant au travers du port I2C. Ce programme fait clignoter la sortie 3 toutes les secondes.

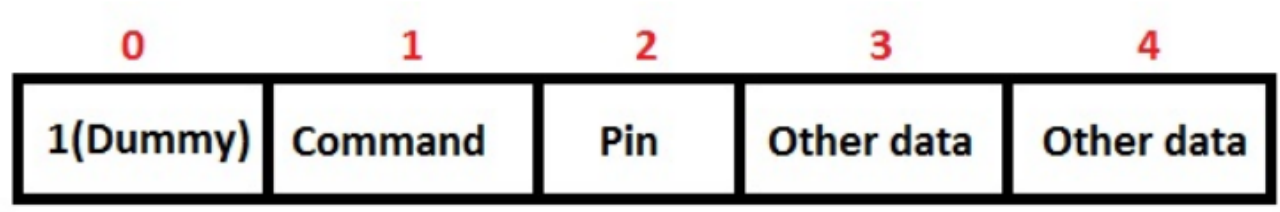
```
import time
import smbus

bus = smbus.SMBus(1)
p_version = 3

def read_i2c_byte(address):
    for i in range(retries):
        try:
            return bus.read_byte(address)
        except IOError:
            print("IOError")
    return -1

def write_i2c_block(address,block):
    for i in range(retries):
        try:
            return bus.write_i2c_block_data(address,1,block)
        except IOError:
            print("IOError")
    return -1
```

```
def read_i2c_block(address):
    for i in range(retries):

        try:
            return bus.read_i2c_block_data(address,1)
        except IOError:
            print("IOError")
    return -1


def digitalWrite(pin, value):
    write_i2c_block(address, dWrite_cmd + [pin, value, unused])
    return 1

def pinMode(pin, mode):
    if mode == "OUTPUT":
        write_i2c_block(address, pMode_cmd + [pin, 1, unused])
```

```
    elif mode == "INPUT":
        write_i2c_block(address, pMode_cmd + [pin, 0, unused])
    return 1

#main
retries = 10
unused = 0

dRead_cmd = [1]
dWrite_cmd = [2]
dht_temp_cmd=[40]
pMode_cmd=[5]

address = 0x04
pin=3
loop=0


pinMode(pin, "OUTPUT")

while True:
    loop = loop + 1
    print (loop, pin, loop % 2)
    digitalWrite(pin, loop % 2)
    time.sleep(1)
```

Le raspberry est la maître et l'arduino est l'esclave (adresse 0x04). Un bloc de 5 bytes est transmis pour paramétrer la carte UNO.

Le premier est fantôme Le second correspond à un ordre précis Le troisième généralement la broche concernée (pin) Les suivants sont les paramètres.



Exemple écriture digitale

1 - dWriteCmd ([2]) + paramètres [pin, value, unused]

Voir le fichier source arduino et le fichier python grovepi.py
https://github.com/DexterInd/GrovePi/blob/master/Software/Python/grovepi.py voir également
https://www.dexterindustries.com/GrovePi/programming/grovepi-protocol-adding-custom-sensors/

Si on veut ajouter des fonctionnalités, il faut donc aller charcuter le fichier C arduino, ajouter les librairies et construire les briques NodeRed pour y accéder ou attendre que quelqu'un le fasse à notre place.

La preuve est donc bien faite qu'une carte GrovePi+ n'est rien de plus qu'une carte arduino munie d'une shield Grove et pilotée par le port I2C. Il doit même être possible de la programmer avec le connecteur ISP (ce que fait la GrovePi).

Vous avez un Arduino Mega ou Mini c'est possible aussi ;)

Voyez on se couchera moins bête ce soir. ;)

Un Esp8266 pourra aussi prendre la main en I2C sur une Arduino, mais il faudra maîtriser les deux côtés!

```
La où beaucoup de monde vous limite et cherche à vous vendre leur service,
le monde du libre
[debian, python, Arduino, Raspberry] vous offre la possibilité de reprendre
la main sur l'ensemble
du système et de la gestion de vos données. Les solutions sont simples: il y
a les esp8266, les
arduinos, les raspberrys et équivalents. Tout le reste n'est qu'une
combinatoire de ces éléments.

Côté programmation on a la méthode scratch-like, la méthode blockly (la
dernière version de scratch
sera en blockly également) ou node-red et la programmation en dur (python, C
Arduino, freepascal, ...), parfois
nécessaire d'y mettre le nez pour charger un firmware.
```