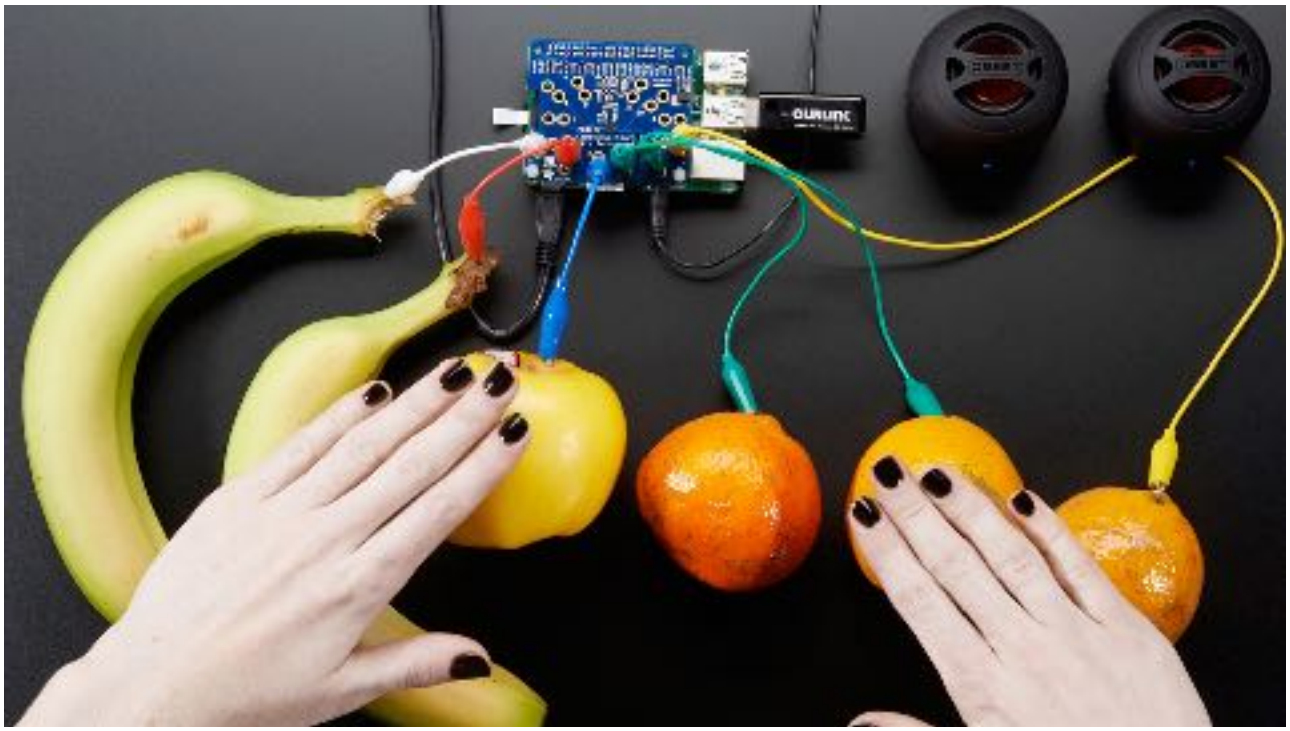
Rapport M2M

Thomas Lerchundi
M2 GI

Matériel Utilisé

-> Raspberry PI model A+ B+ ou PI2
->Clé wifi compatible avec Raspberry
->Capacitive Touch HAT for Raspberry Pi - Mini Kit - MPR121



TUTORIEL POUR UTILISER LES I/O AVEC LE CAPTEUR :

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-gpio

Docker :

Voici les différentes images que j'ai utilisé localement sur Docker

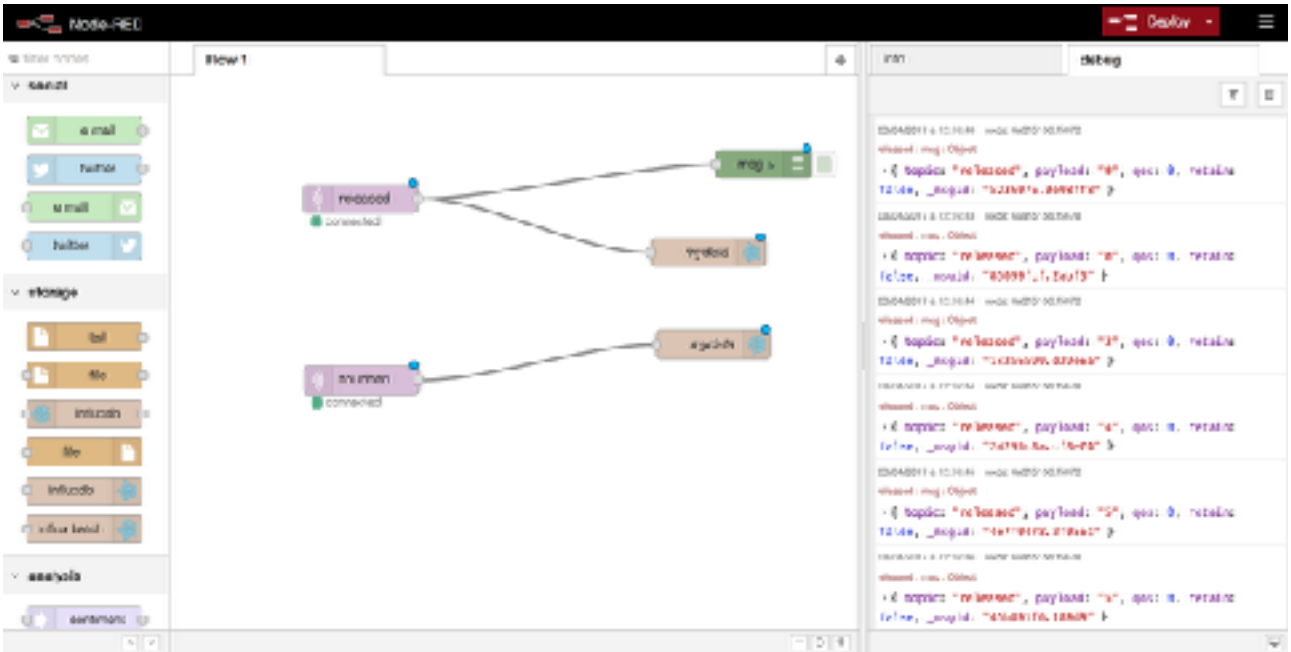
```
CONTAINER ID        IMAGE                           COMMAND
CREATED             STATUS              PORTS
NAMES
cfc1cc55275c        nodered/node-red-docker         "npm start -- --us..."
7 days ago          Up 7 days           0.0.0.0:1880->1880/tcp
mynodered
b0ddbec404a9        villlem/mqtt-server:latest      "/sbin/my_init"
7 days ago          Up 7 days           0.0.0.0:1883->1883/tcp
mqtt-broker
dbc4eaaa1e87        grafana/grafana                 "/run.sh"
13 days ago         Up 7 days           0.0.0.0:3000->3000/tcp
infallible_morse
6aceb426a623        tutum/influxdb                  "/run.sh"
13 days ago         Up 7 days           0.0.0.0:8083->8083/tcp, 8090/
tcp, 0.0.0.0:8086->8086/tcp, 8099/tcp   zealous_mayer
```

```
Deux topic pour lequel le serveur mqtt est abonné :

-> touched
-> released

Les données sont envoyées depuis le PI via mqtt (voir partie CODE)
```

Node RED

Pour obtenir les noeuds influxDB sur node Red:

npm install node-red-contrib-influxdb



Importation du noeud NodeRed déjà configuré:

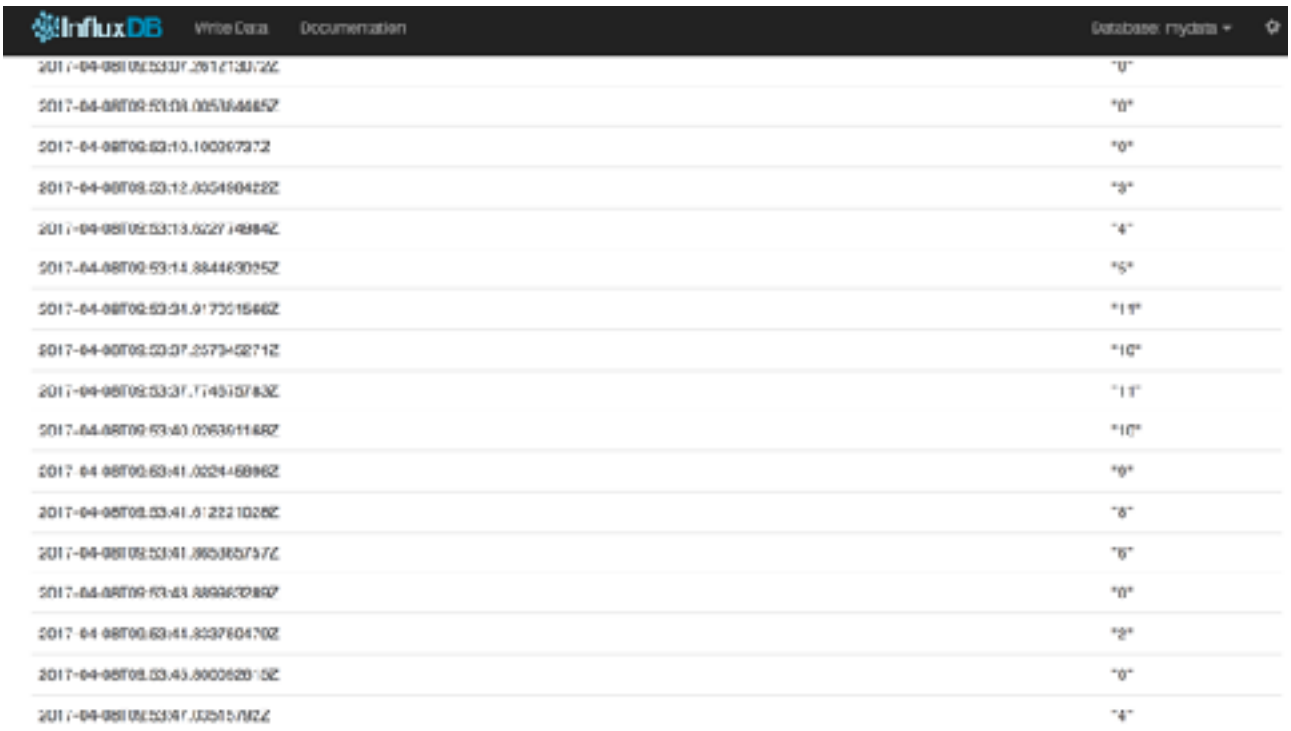
[{"id":"6a815156.f5678","type":"debug","z":"99b6249e.db5ac8","name":"","active":true,"console":"true","complete":"true","x":619.5,"y":93,"wires":[]},{"id":"de8f0bc7.a38928","type":"mqtt in","z":"99b6249e.db5ac8","name":"","topic":"released","qos":"1","broker":"5ba753c2.8ca25c","x":198.5,"y":129,"wires":[["bd6945e4.a44e28","6a815156.f5678"]]},{"id":"d1d44ae0.5291a8","type":"mqtt in","z":"99b6249e.db5ac8","name":"","topic":"touched","qos":"1","broker":"b118a0ed.23abd","x":199.5,"y":318,"wires":[["99fa9e77.9ca43"]]},{"id":"bd6945e4.a44e28","type":"influxdb out","z":"99b6249e.db5ac8","influxdb":"57ed5313.7962bc","name":"mydata","measurement":"mydata_test","x":561.5,"y":185,"wires":[]},{"id":"99fa9e77.9ca43","type":"influxdb out »,"z":"99b6249e.db5ac8","influxdb":"57ed5313.7962bc","name":"mydata","measurement":"mydata_test2","x":567,"y":281,"wires":[]},{"id":"5ba753c2.8ca25c","type":"mqtt-broker","z":"","broker":"mqtt: 127.0.0.1","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":true,"willTopic":"","willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""},{"id":"b118a0ed.23abd","type":"mqtt-broker","z":"","broker":"mqtt: 127.0.0.1","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":true,"willTopic":"","willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""},{"id":"57ed5313.7962bc","type":"influxdb","z":"","hostname":"ibd: 127.0.0.1","port":"8086","protocol":"http","database":"mydata","name":"","usetls":false,"tls":""}]

InfluxDB

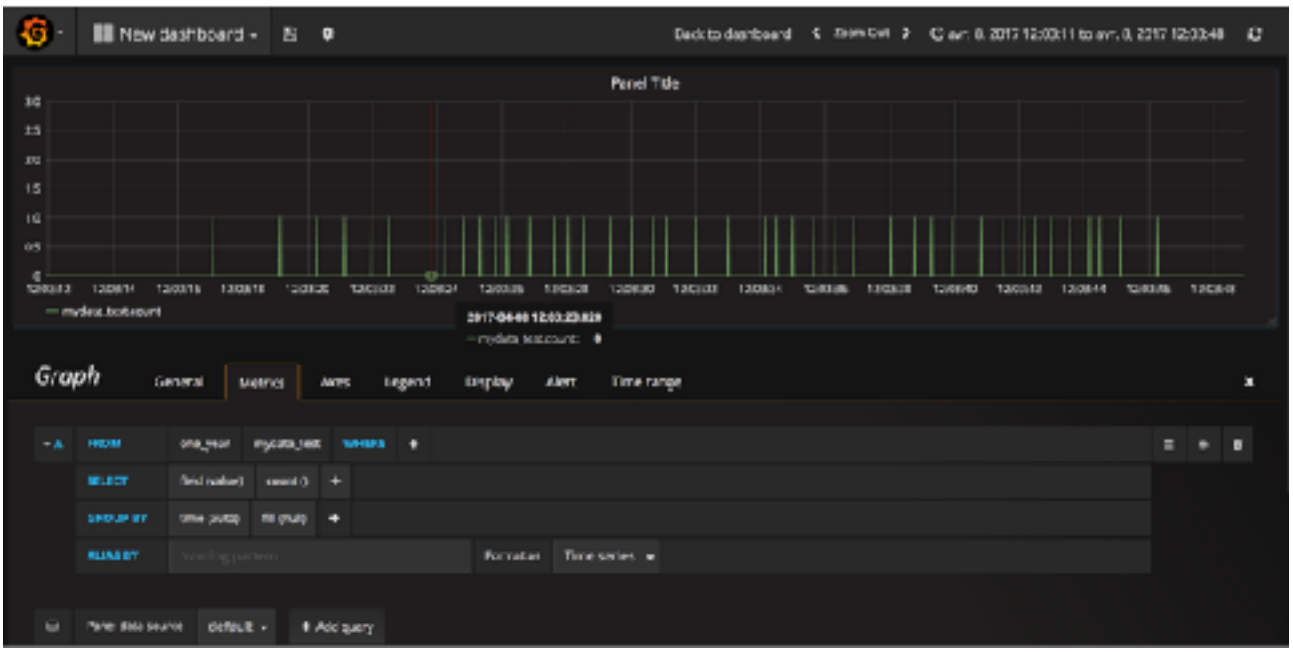
J'ai crée une base de donnée appelée mydata.

2 tables de crées

-mydata_test (correspond à l'action touched)
-mydata_test2 (correspond à l'action released)



Visualisation sous Grafana



Code

SCRIPT PYTHON POUR RASPBERRY :

```python
import Adafruit_MPR121.MPR121 as MPR121
import paho.mqtt.publish as publish
import os


print('Adafruit MPR121 Capacitive Touch Sensor Test')

# Create MPR121 instance.
cap = MPR121.MPR121()

# Initialize communication with MPR121 using default I2C bus of device, and
# default I2C address (0x5A).  On BeagleBone Black will default to I2C bus 0.
if not cap.begin():
    print('Error initializing MPR121.  Check your wiring!')
    sys.exit(1)

# Alternatively, specify a custom I2C address such as 0x5B (ADDR tied to 3.3V),
# 0x5C (ADDR tied to SDA), or 0x5D (ADDR tied to SCL).
#cap.begin(address=0x5B)

# Also you can specify an optional I2C bus with the bus keyword parameter.
#cap.begin(busnum=1)

# Main loop to print a message every time a pin is touched.
print('Press Ctrl-C to quit.')
last_touched = cap.touched()
while True:
    current_touched = cap.touched()
    # Check each pin's last and current state to see if it was pressed or released.
    for i in range(12):
        # Each pin is represented by a bit in the touched value.  A value of 1
        # means the pin is being touched, and 0 means it is not being touched.
        pin_bit = 1 << i
        # First check if transitioned from not touched to touched.
        if current_touched & pin_bit and not last_touched & pin_bit:
            print('{0} touched!'.format(i))
            publish.single("touched",i,hostname = "mac")
            if i % 5 == 0:
                os.system('mpg123 -q /home/pi/Desktop/SOL.mp3 &')
            if i % 5 == 1:
                os.system('mpg123 -q /home/pi/Desktop/SIB.mp3 &')
            if i % 5 == 2:
                os.system('mpg123 -q /home/pi/Desktop/DO.mp3 &')
            if i % 5 == 3:
                os.system('mpg123 -q /home/pi/Desktop/FA.mp3 &')
            if i % 5 == 4:
                os.system('mpg123 -q /home/pi/Desktop/FAD.mp3 &')
        # Next check if transitioned from touched to not touched.
        if not current_touched & pin_bit and last_touched & pin_bit:
            print('{0} released!'.format(i))
            publish.single("released",i,hostname = "mac")
    # Update last state and wait a short period before repeating.
    last_touched = current_touched
    time.sleep(0.1)
```