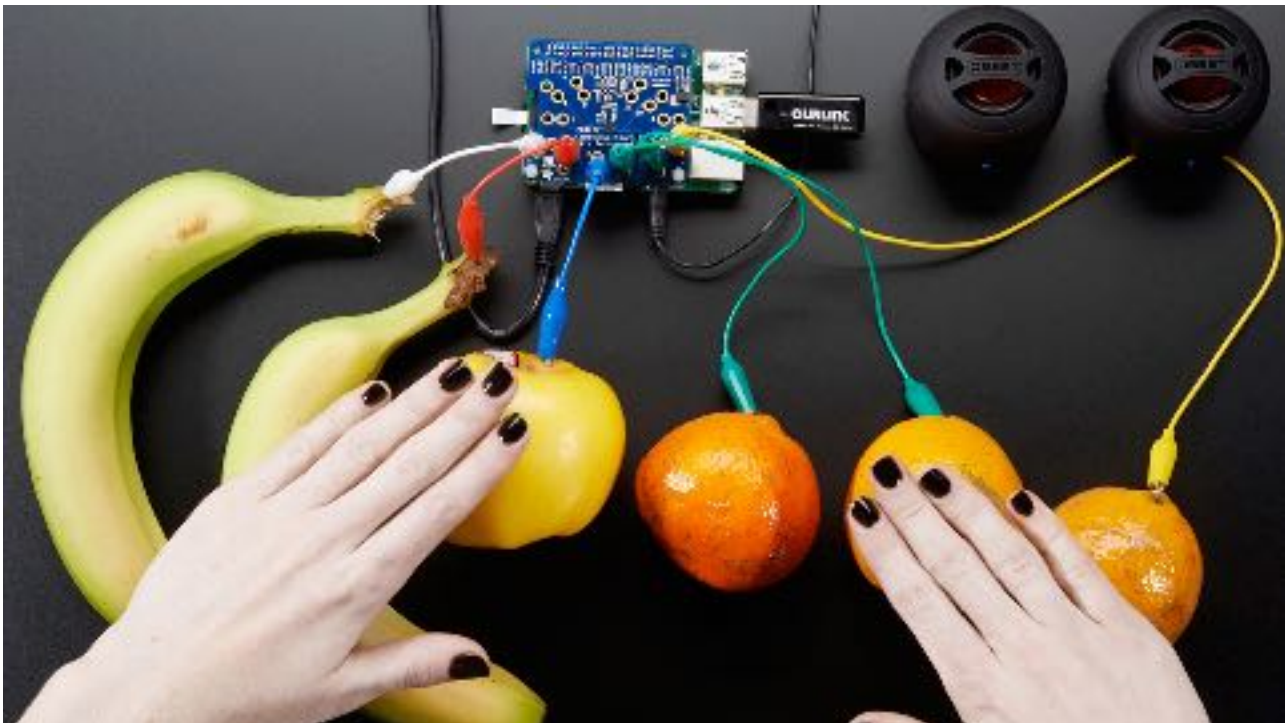


Rapport M2M

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 CREATED NAMES	IMAGE STATUS	PORTS	COMMAND
cfc1cc55275c 7 days ago mynodered	nodered/node-red-docker Up 7 days	0.0.0.0:1880->1880/tcp	"npm start -- --us..."
b0ddb404a9 7 days ago mqtt-broker	villem/mqtt-server:latest Up 7 days	0.0.0.0:1883->1883/tcp	"/sbin/my_init"
dbc4eaaa1e87 13 days ago infallible_morse	grafana/grafana Up 7 days	0.0.0.0:3000->3000/tcp	"/run.sh"
6aceb426a623 13 days ago tcp, 0.0.0.0:8086->8086/tcp, 8099/tcp	tutum/influxdb Up 7 days	0.0.0.0:8083->8083/tcp, 8090/ zealous_mayer	"/run.sh"

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
```

The screenshot shows the Node-RED web interface. On the left, there are panels for 'storage' and 'analysis'. The main workspace contains a flow with two MQTT nodes: 'mqtt in' and 'mqtt out'. The 'mqtt in' node is connected to the 'mqtt out' node. On the right, the 'debug' console displays a series of MQTT messages with their respective topics and payloads.

```
ED048014 10:16:41  INFO  MQTT: MESSAGE  
Received : msg : Object  
- { topic: "released", payload: "4", qos: 0, retain: false, _msgid: "c23e099-8226e8" }  
ED048014 10:16:42  INFO  MQTT: MESSAGE  
Received : msg : Object  
- { topic: "released", payload: "4", qos: 0, retain: false, _msgid: "c23e099-8226e8" }  
ED048014 10:16:43  INFO  MQTT: MESSAGE  
Received : msg : Object  
- { topic: "released", payload: "4", qos: 0, retain: false, _msgid: "c23e099-8226e8" }  
ED048014 10:16:44  INFO  MQTT: MESSAGE  
Received : msg : Object  
- { topic: "released", payload: "4", qos: 0, retain: false, _msgid: "c23e099-8226e8" }  
ED048014 10:16:45  INFO  MQTT: MESSAGE  
Received : msg : Object  
- { topic: "released", payload: "4", qos: 0, retain: false, _msgid: "c23e099-8226e8" }
```

Importation du noeud NodeRed déjà configuré:

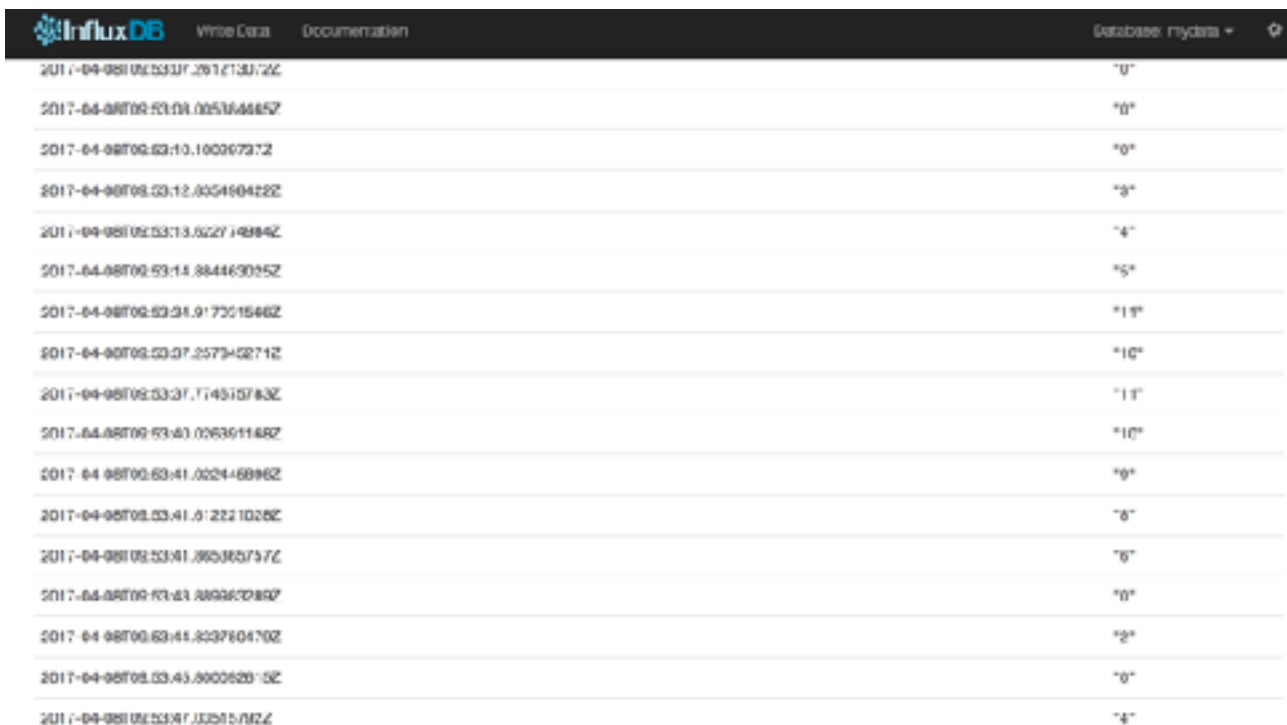
```
{"id":"6a815156.f5678","type":"debug","z":"99b6249e.db5ac8","name":"","active":true,"console":true,"complete":true,"x":619.5,"y":93,"wires":[]},{ "id":"de8f0bc7.a38928","type":"mqtt in","z":"99b6249e.db5ac8","name":"","topic":"released","qos":1,"broker":"5ba753c2.8ca25c","x":198.5,"y":129,"wires": [{"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":"ibid: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 créés

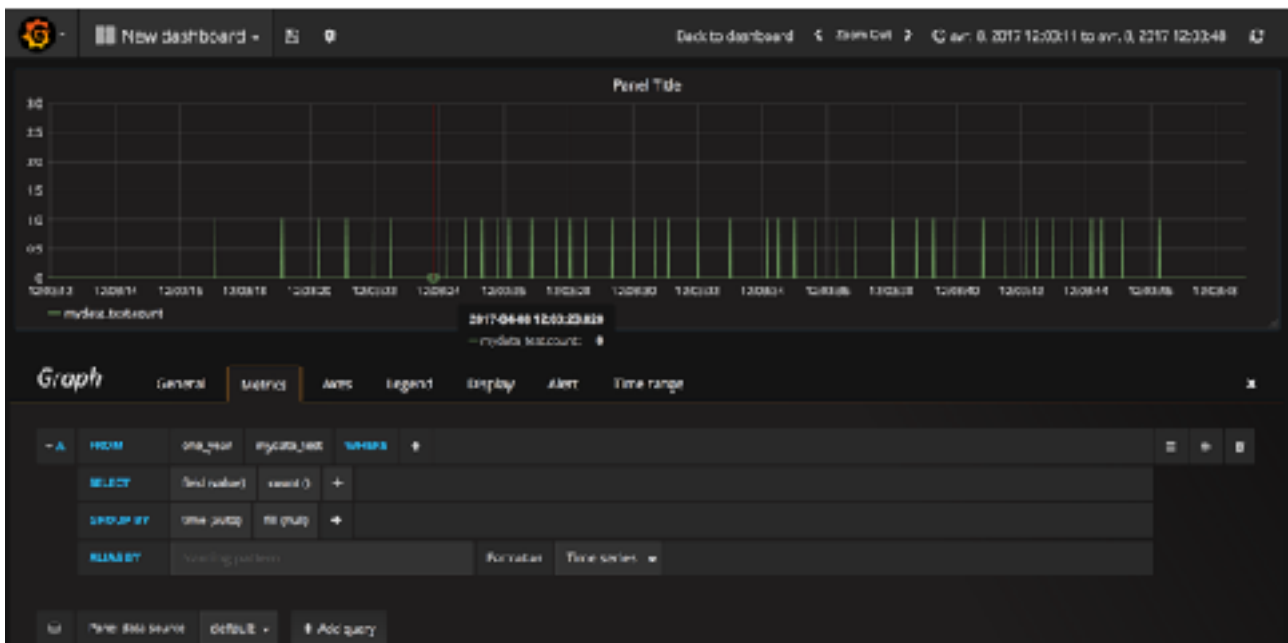
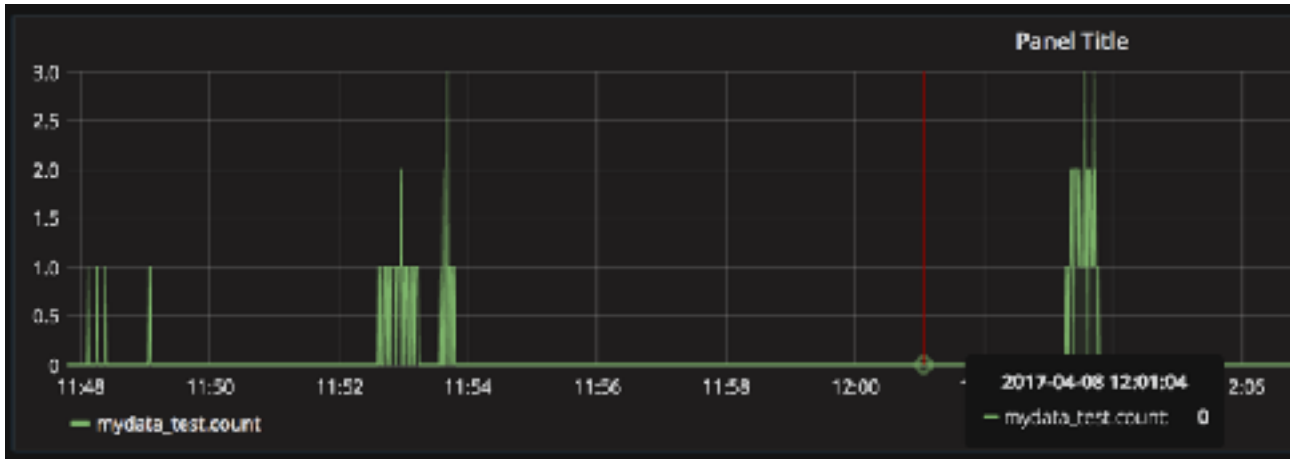
- mydata_test (correspond à l'action touched)
- mydata_test2 (correspond à l'action released)



The screenshot shows the InfluxDB web interface with a table of data points. The table has two columns: a timestamp and a value. The values are integers ranging from 0 to 14. The interface includes the InfluxDB logo, navigation links for 'Write Data' and 'Documentation', and a dropdown menu for the database 'mydata'.

Timestamp	Value
2017-04-08T09:53:37.261213012Z	0
2017-04-08T09:53:38.00516465Z	0
2017-04-08T09:53:10.10036791Z	0
2017-04-08T09:53:12.005160428Z	3
2017-04-08T09:53:13.522714984Z	4
2017-04-08T09:53:14.88446305Z	5
2017-04-08T09:53:31.917301546Z	11
2017-04-08T09:53:37.257345271Z	10
2017-04-08T09:53:37.174513783Z	11
2017-04-08T09:53:40.006301148Z	10
2017-04-08T09:53:41.00244896Z	9
2017-04-08T09:53:41.91221026Z	8
2017-04-08T09:53:41.86530575Z	6
2017-04-08T09:53:43.88936298Z	0
2017-04-08T09:53:44.803760470Z	8
2017-04-08T09:53:45.800052010Z	0
2017-04-08T09:53:47.00201592Z	4

Visualisation sous Grafana



Code

SCRIPT PYTHON POUR RASPBERRY :

```
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)
```