



MONITORING : MISE EN PLACE D'UNE SOLUTION IoT AVEC
AZURE

Dossier d'installation

Auteur :
Florian GRANTE

Superviseur :
David LAROSE

Table des matières

Introduction	1
1 Présentation du matériel	2
1.1 La <i>RaspberryPi</i>	2
1.2 Le shield <i>GrovePi+</i>	3
1.3 Les composants	4
1.3.1 Le capteur de température et d'humidité	4
1.3.2 Le capteur de luminosité	4
1.3.3 L'écran	5
1.3.4 L'encodeur rotatif	5
2 Configuration de la station	6
2.1 Création et installation de la distribution <i>Raspbian</i>	6
2.2 Première mise en route de la <i>RaspberryPi</i>	7
2.3 Installation, configuration et test du code de la station avec ses capteurs.	12
2.3.1 Installation des capteurs.	12
2.3.2 Configuration de la <i>RaspberryPi</i>	13
2.3.3 Test de la station	14
3 Configuration de <i>Microsoft Azure</i>	15
3.1 Mise en place pour 1 station	15
3.1.1 Explication des fonctionnalités proposées par <i>Microsoft</i>	15
3.1.2 Crédit d'un Hub d'évènements sur <i>Azure</i> avec la <i>CLI</i>	15
3.1.2.1 Installation de l'Interface en Ligne de Commande de <i>Azure</i>	15
3.1.2.2 Crédit d'un groupe de ressources et d'un Hub d'évènement	15
3.1.3 Crédit d'un portail <i>Time Series Insights</i>	16
3.1.4 Ajout d'une source d'évènements dans le portail <i>Time Series Insights</i>	19
3.1.5 Configuration logiciel de la station pour <i>Azure</i>	21

Introduction

L'objectif de ce dossier est de vous expliquer de A à Z la mise en place d'un système avec capteurs permettant l'affichage en temps réel de différentes données comme la température, l'humidité ou encore la luminosité d'une pièce, d'un couloir, etc....

En premier lieu, nous allons nous attarder sur la partie matériel : *La Raspberry Pi* (je vous laisserais le choix du genre, c'est une question qui divise la France comme bien d'autre)



FIGURE 1 – *Une RaspberryPi à nue*

Nous aborderons l'installation de capteur sur cette carte puis nous verrons l'installation logiciel.

Dans un second temps, nous verrons ensemble comment envoyer ces données dans le *Cloud* mis en place par *Microsoft : Azure*

A la fin de ce dossier, vous serez en mesure de regarder n'importe où dans le monde la température ambiante de votre installation mais cessons de tergiverser et mettons nous au travail !

Chapitre 1

Présentation du matériel

1.1 La *RaspberryPi*

Si vous avez été intrigué par la bête que je vous ai mis en photo dans l'introduction, pas de panique, elle va devenir votre nouvelle meilleure amie. Il s'agit ni plus ni moins que d'un ordinateur en modèle réduit. Cette puce est capable de faire tourner une distribution de *Linux* connue sous le nom de *Raspbian*. La *RaspberryPi* est un véritable bijoux de miniaturisation pour les *Makers* qui désire embarquer des ordinateurs dans leur projets les plus fou. Vous l'aurez compris, cela va être la base de notre station.

Pourquoi ce choix ?

La *RaspberryPi* peut être repoussante au début car vendue tel que vous avez pu la voir plus tôt mais une fois prise en main, elle est simple d'utilisation. Elle permet d'avoir un accès internet de façon simple avec son port ethernet, on peut y brancher un écran ainsi que des périphériques si nécessaire pour réaliser de la maintenance (même si il existe d'autre moyen d'y accéder mais nous verrons ça plus tard). Une raison non négligeable pour notre projet est son prix. Compté une quarantaine d'euros pour le dernier modèle en date. Elle possède également des broches pour réaliser de l'électronique embarquée comme nous allons le faire. Ainsi notre ordinateur est modulaire et on peut y ajouter des composants par ces broches.

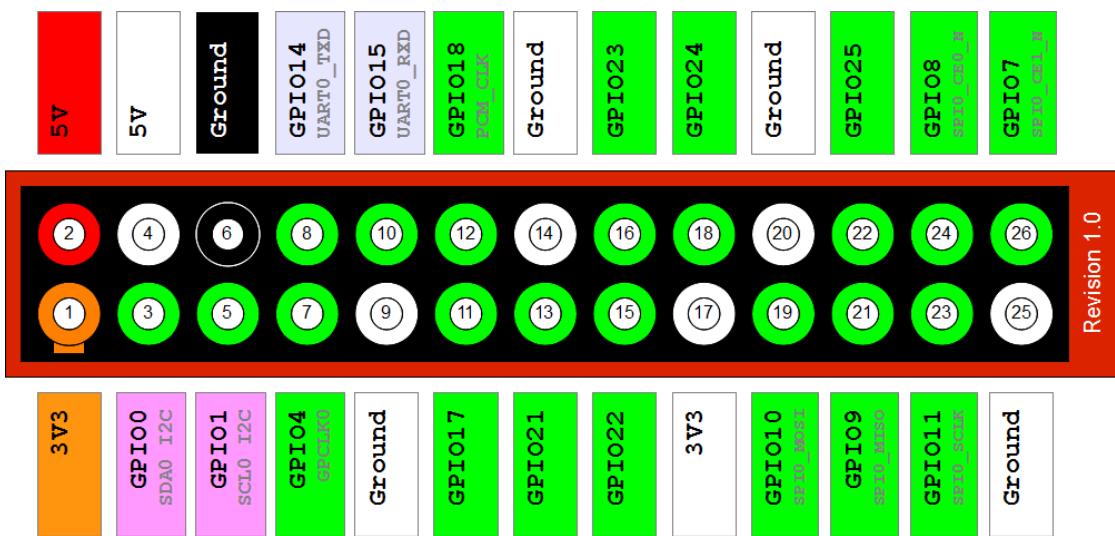


FIGURE 1.1 – Pin Mapping de la *RaspberryPi* 3

Son autre atout et non des moindres, c'est sa communauté qui s'est étendue de façon exponentielle ces dernières années car accessible à tout le monde et permet alors une aide conséquente via les forums.

Vous voilà plus familier avec l'engin, si vous voulez plus d'information, je vous laisse visiter [le site officiel](#) où vous pouvez notamment télécharger la version de base de son OS, *Raspbian* mais qui possède nombre de version modifiée (un OS pour réaliser une station de jeux rétro est un exemple)

1.2 Le shield *GrovePi+*

Je vous parlait juste avant que la *RaspberryPi* possédait des broches. Ces dernières vont nous servir à y installer les capteurs. Mais nous n'allons pas brancher les capteurs directement sur la *RaspberryPi*. En effet nous allons ajouter ce que l'on appelle un "*Shield*". Il s'agit ni plus ni moins que d'une autre carte électronique avec ses fonctions propres qui se branche sur la *RaspberryPi* pour communiquer avec elle. Ainsi, nous pouvons utiliser les fonctions du "*Shield*" en envoyant des指令 de la *RaspberryPi*.

Celui que nous utilisons porte le doux nom *GrovePi+*. Sans m'attarder sur ses détails techniques, sachez (pour les connaisseurs) qu'elle embarque un microcontrôleur ATMega pour gérer les capteurs. Il s'agit ni plus ni moins d'une version "*User Friendly*" d'un *Arduino* que l'on branche sur notre *RaspberryPi*.

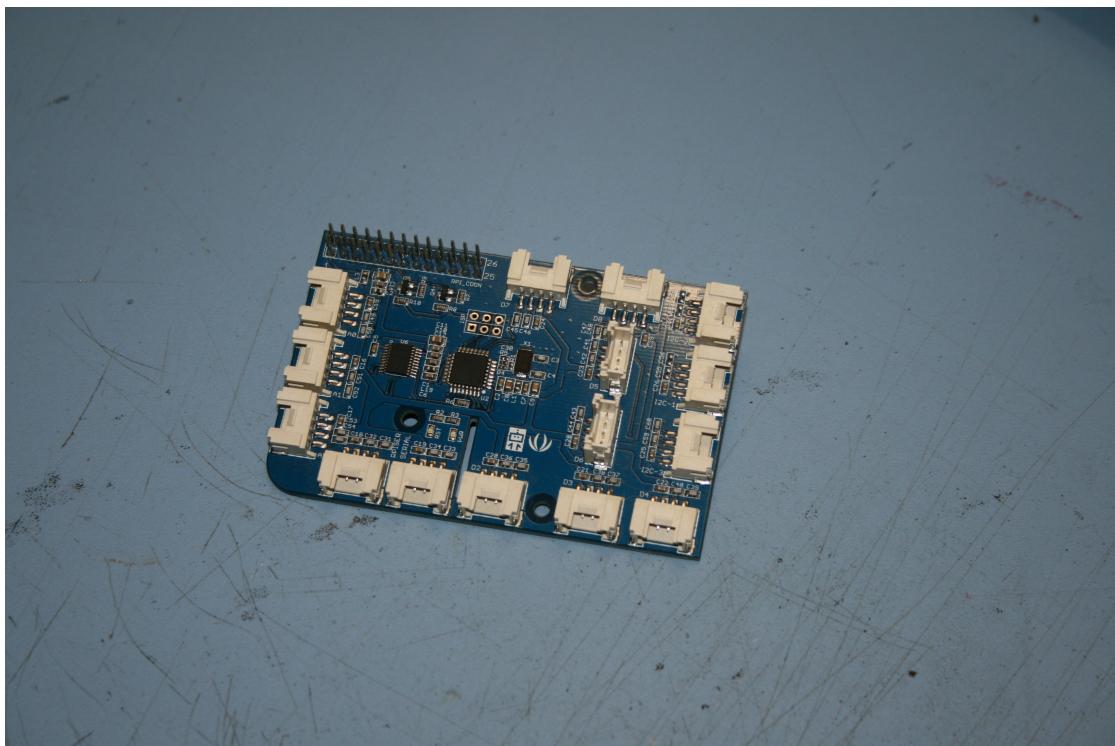


FIGURE 1.2 – le Shield *GrovePi+*

Comme vous voyez sur la photo, nous avons des connecteurs *4-pins*, en blanc. C'est là dessus que l'on va brancher nos capteurs.

ATTENTION : Ces connecteurs sont reliés à des broches particulière du microcontrôleur. Ils sont donc chacun d'entre eux prévu pour un type de composant particulier.

On distingue trois types de connecteurs parmi ceux-là :

- ports analogiques : Ils sont disponibles uniquement en *INPUT*, c'est à dire qu'il s'agit d'une entrée pour une tension comprise entre 0 et 5V. Le *Shield* possède un convertisseur analogique numérique qui échantillonne cette valeur sur 10 bits (0V = 0 et 5V = 1023). Exemple si on envoie 2.5V, alors on aura la valeur 512)
- ports digitaux : ceux-ci peuvent être réglés en *INPUT* ou en *OUTPUT*. Comme ce sont des entrée/sortie numérique, elle répond à la loi du tout ou rien. Soit nous avons 0V, soit nous avons 5V (0 ou 1). Typiquement, pour une sortie, cela peut être utilisé comme interrupteur électronique ou pour allumer une LED.
- ports I2C : Il s'agit d'un protocole de communication développé par Philips pour minimiser le nombre de fil nécessaire pour faire communiquer deux périphériques entre eux. Il y a en effet uniquement 3 fils : un signal de données (SDA), un signal d'horloge (SCL) et un signal de référence électrique (masse).

Nous allons maintenant parler rapidement des différents capteurs et composants que nous allons brancher sur notre *Shield*

1.3 Les composants

1.3.1 Le capteur de température et d'humidité

Les capteurs d'humidité répondent à une loi qui dépend de la température. C'est pour cette raison que ces capteurs fournissent bien souvent les deux à la fois. De notre côté, il s'agit du capteur *DHT11*. Ce composant utilise un connecteur digital

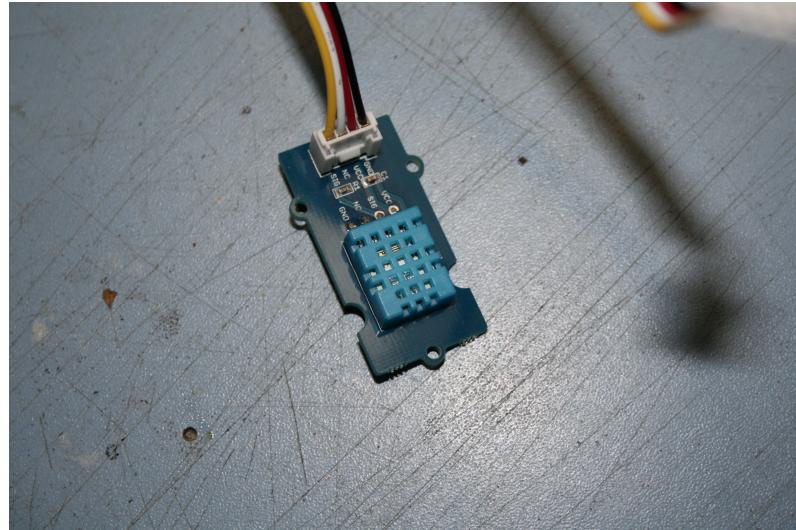


FIGURE 1.3 – Capteur d'humidité et de température

[Datasheet du DHT11](#)

1.3.2 Le capteur de luminosité

Le capteur de luminosité mesure la lumière ambiante. Exprimé en Luxmètre, les capteurs de luminosité "basique" ne permettent pas d'obtenir une valeur précise de la luminosité. Néanmoins, cette valeur est très abstraite pour l'utilisateur. C'est pour cette raison que nous allons l'utiliser pour déterminer si la lumière de la pièce où il est placé est allumée ou non. Nous détaillerons un protocole à réaliser pour essayer de faire marcher au mieux ce capteur. Ce composant utilise un connecteur analogique.

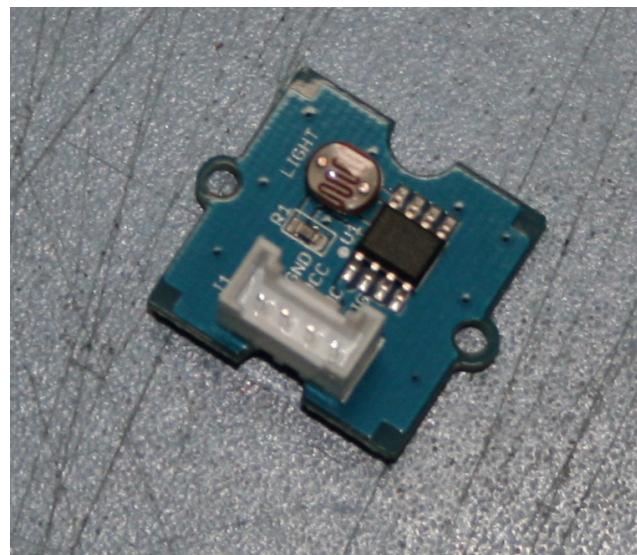


FIGURE 1.4 – Le capteur de luminosité

[Datasheet du capteur de luminosité](#)

1.3.3 L'écran

Pour pouvoir accéder localement aux valeurs des capteurs, nous allons placer un écran. Nous pouvons afficher 2 lignes de 16 caractères sur ce dernier et choisir la couleur de fond qui est affichée. Ce composant utilise un connecteur I2C.

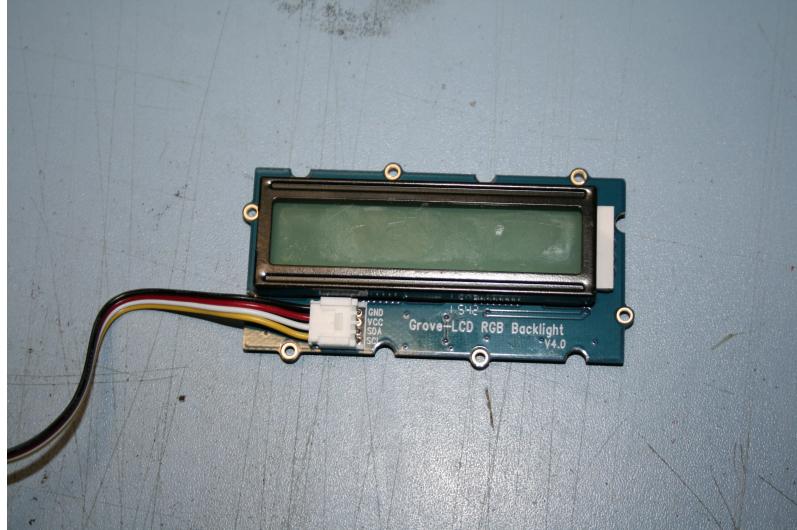


FIGURE 1.5 – L'écran LCD

1.3.4 L'encodeur rotatif

Nous n'allons pas l'utiliser comme un capteur à proprement parler ici mais comme un bouton tournant pour naviguer dans différents affichages sur l'écran. On pourra ainsi passer de l'affichage de la température à l'humidité,etc...



FIGURE 1.6 – Encodeur rotatif pour bouton tournant

Maintenant que nous avons connaissance du matériel que nous avons à disposition pour notre station, nous allons pouvoir passer à son installation.

Chapitre 2

Configuration de la station

2.1 Creation et installation de la distribution *Raspbian*.

Pour une *RaspberryPi*, son disque dur est nul autre qu'une carte Micro SD. C'est donc sur ce support que vous allez faire l'installation.

Pour realiser cette tape, vous aurez besoin de :

- d'une *RaspberryPi*
- d'une carte micro SD de minimum 8Go
- du logiciel *Etcher*
- de *Raspbian*. En realite il s'agit d'une version modifie par la socit Dexter Industries qui est spcialis dans l'utilisation de *RaspberryPi* pour la robotique.
- d'un adaptateur pour relier la micro SD  votre ordinateur.

Vous pouvez dsormais commencer l'installation de l'OS sur notre *RaspberryPi*

1. Connecter la micro SD sur votre ordinateur.

2. Lancer le logiciel *Etcher*



FIGURE 2.1 – Logiciel *Etcher* au lancement

3. Cliquer sur "Select image"  gauche puis selectionner le fichier .zip rcupr depuis le site *SourceForce* dans les pr-requis.
4. Vrifer que le priphrique qui est renseign au centre est bien la micro SD. Dans le cas contraire cliquer sur "Change".

5. Si les deux étapes précédentes sont OK, cliquer sur "Flash".



FIGURE 2.2 – *Etcher* prêt à flasher

Et voilà, l'opération peut prendre une dizaine de minutes. C'était simple non ?

2.2 Première mise en route de la *RaspberryPi*.

1. Maintenant que vous avez *Raspbian*, vous allez pouvoir démarrer votre nouvel ordinateur. Pour cela, il vous suffit d'insérer la microSD au dos de la *RaspberryPi*.



FIGURE 2.3 – La *RaspberryPi* de dos

2. Avant de l'alimenter, nous allons distinguer deux cas.. Si vous avez la possibilité, de façon extérieur, d'accéder à l'adresse IP de votre *RaspberryPi* alors vous pouvez sauter l'étape N°3. Si cela n'est pas possible, brancher un écran et un clavier souris.

Vous pouvez maintenant l'alimenter en utilisant le port micro USB qui est à côté du port HDMI.

3. Si vous suivez cette étape, vous devriez voir apparaître des lignes de commandes qui défilent. Quelques instants plus tard, vous arrivez sur l'environnement de bureau de votre *RaspberryPi*. Sur ce bureau, ouvrez un terminal en cliquant sur l'écran en haut à gauche :



FIGURE 2.4 – Barre de tâche de Raspbian

Une fois le terminal ouvert, entrer la commande suivante :

```
1 sudo ifconfig
```

un mot de passe devrait vous être demandé, par défaut, le mot de passe est "robots1234". Vous devriez avoir un résultat de ce type :

```
pi@dex: ~
File Edit Tabs Help
access control disabled, clients can connect from any host
pi@dex: ~ $ sudo ifconfig
eth0      Link encap:Ethernet HWaddr b8:27:eb:d8:e5:50
          inet addr:192.168.1.39 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::561a:4b99:6d77/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:400 errors:0 dropped:0 overruns:0 frame:0
          TX packets:131 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30959 (30.2 KiB) TX bytes:17625 (17.2 KiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:220 errors:0 dropped:0 overruns:0 frame:0
          TX packets:220 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:17774 (17.3 KiB) TX bytes:17774 (17.3 KiB)

wlan0    Link encap:Ethernet HWaddr b8:27:eb:8d:b0:05
          inet6 addr: fe80::b80b:2243:a88f:6bab/64 Scope:Link
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:122 errors:0 dropped:122 overruns:0 frame:0
```

FIGURE 2.5 – Résultat de la commande sudo ifconfig

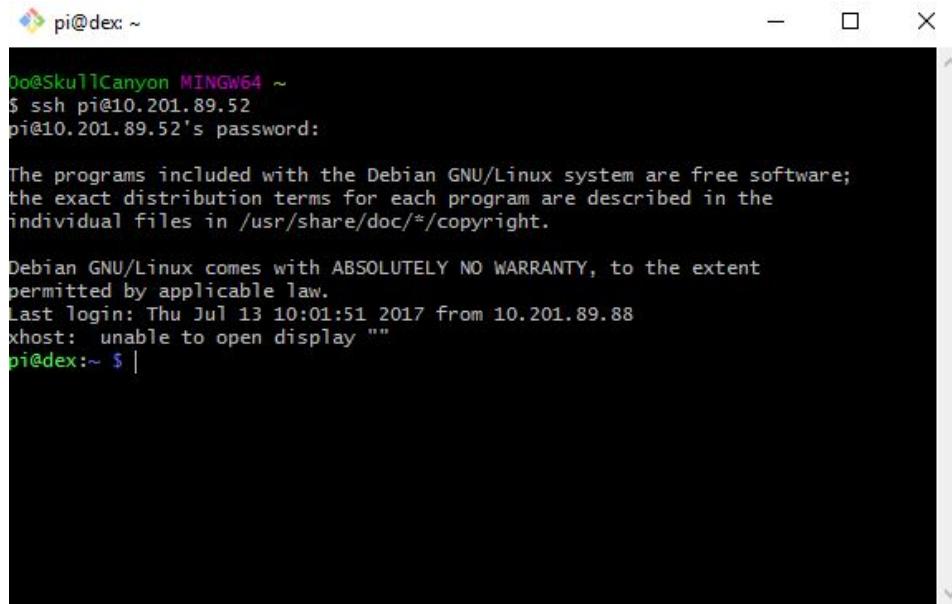
Vous pouvez ainsi récupérer l'adresse IP et adresse MAC si nécessaire pour gérer votre réseau. Pour l'adresse IP, selon l'architecture de votre réseau, il s'agit d'une adresse privé ou public. Dans le premier cas, vous serez obligé d'être dans le même réseau pour y accéder, dans le second pas de problème quelque soit l'endroit où vous vous trouver. Maintenant que vous avez récupérer l'adresse IP, vous pouvez débrancher tous les périphériques de votre *RaspberryPi* et laisser uniquement l'alimentation et le câble ethernet.

4. Très bien, désormais que vous avez l'adresse IP à disposition, vous pouvez installer le nécessaire pour utiliser nos capteurs. Vous pourriez très bien continuer cette installation directement sur la *RaspberryPi* mais si vous n'avez jamais fait ce qui va suivre, cela vous fera un bon entraînement.
- Télécharger et installer le logiciel **Git Bash**. Ce logiciel permet l'utilisation d'un terminal plus avancé et compatible avec le langage système *Bash*.
 - Lancer *Git Bash*
 - taper la commande en remplaçant "xxx.xxx.xxx.xxx" par l'adresse IP de la *RaspberryPi*

```
1 ssh pi@xxx.xxx.xxx.xxx
```

la première fois, il vous sera demander si vous voulez réellement vous connecter sur le périphérique, taper alors "yes" puis sur la touche *Entrée*

- (d) Le mot de passe est "robots1234". Si tout c'est bien passé, vous devriez avoir cet aperçu :



```
pi@dex: ~
Do@SkullCanyon MINGW64 ~
$ ssh pi@10.201.89.52
pi@10.201.89.52's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jul 13 10:01:51 2017 from 10.201.89.88
xhost: unable to open display ""
pi@dex:~ $ |
```

FIGURE 2.6 – connexion à la *RaspberryPi* en *SSH*

- (e) vous naviguez maintenant dans la *RaspberryPi*. Dans un premier temps, il va falloir changer le mot de passe car celui ci est un mot de passe par défaut. Enter la commande :

```
1 sudo raspi-config
```

un menu sur fond bleu devrait apparaître :

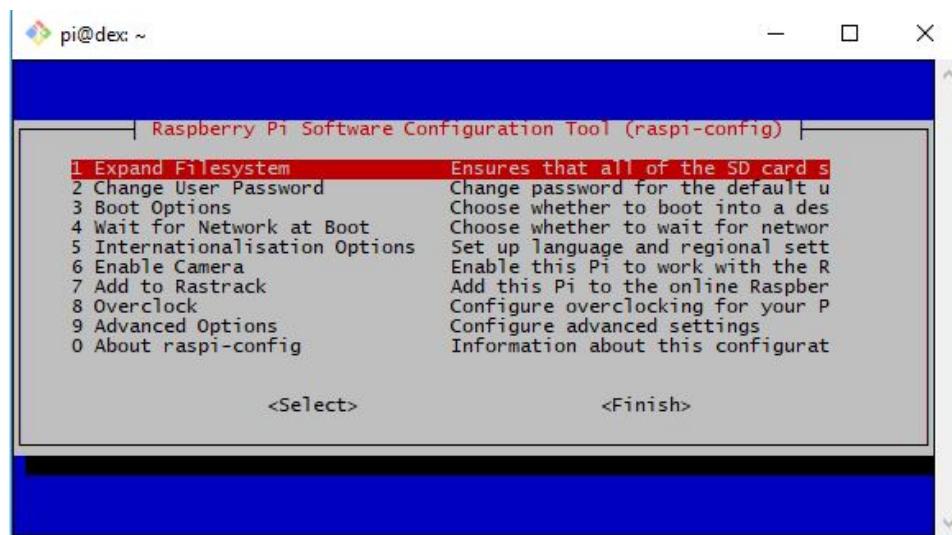


FIGURE 2.7 – Menu *Raspi-config*

Premièrement, choisissez la première option *Expand Filesystem*. Cela va faire en sorte que *Raspbian* occupe toute la micro SD. Une fois terminé, choisissez la seconde option. Nous allons changer le mot de passe par défaut "robots1234". Il va vous être demandé de saisir 2 fois un nouveau mot de passe. Pas de panique, lorsque l'on tape un mot de passe sous linux, aucun caractère apparait !

- (f) Quitter le menu en choisissant l'option "*Finish*". Il vous sera demandé de redémarrer la *RaspberryPi*.
Après quelques temps, refaite l'étape *c d* mais avec le nouveau mot de passe qui a été renseigné.
- (g) Maintenant, vous pouvez installer GrovePi pour pouvoir utiliser le *Shield*. Entrer alors les deux commandes suivantes :

```

1      sudo curl https://raw.githubusercontent.com/DexterInd
2          /Raspbian_Forum_Robots/master/upd_script/fetch_grovepi.sh | bash
3
4      sudo reboot

```

Votre *RaspberryPi* va redémarrer.

- (h) Connectez-vous à nouveau en "ssh" comme pour l'étape 4 mais cette fois ci avec votre nouveau mot de passe.
(i) Réaliser alors cette suite de commande une à une. Appuyer sur la touche *Entrée* lorsque l'on vous demande de continuer :

```

1      cd
2      sudo git clone https://github.com/DexterInd/GrovePi
3      cd GrovePi/Script
4      sudo chmod +x install.sh
5      sudo ./install.sh

```

- (j) Vous devriez à la fin obtenir un écran similaire à la photo précédente.

Effectuer alors la commande :

```
1      sudo shutdown now
```

Elle va alors s'arrêter. Vous pouvez alors la débrancher après quelques instant.

5. Vous pouvez désormais ajouter le *Shield* sur la *RaspberryPi*. Comme ci-dessous **ATTENTION AU BROCHES UTILISÉES, BRANCHER LE SHIELD DANS LA MÊME CONFIGURATION QUE SUR LA PHOTO !**

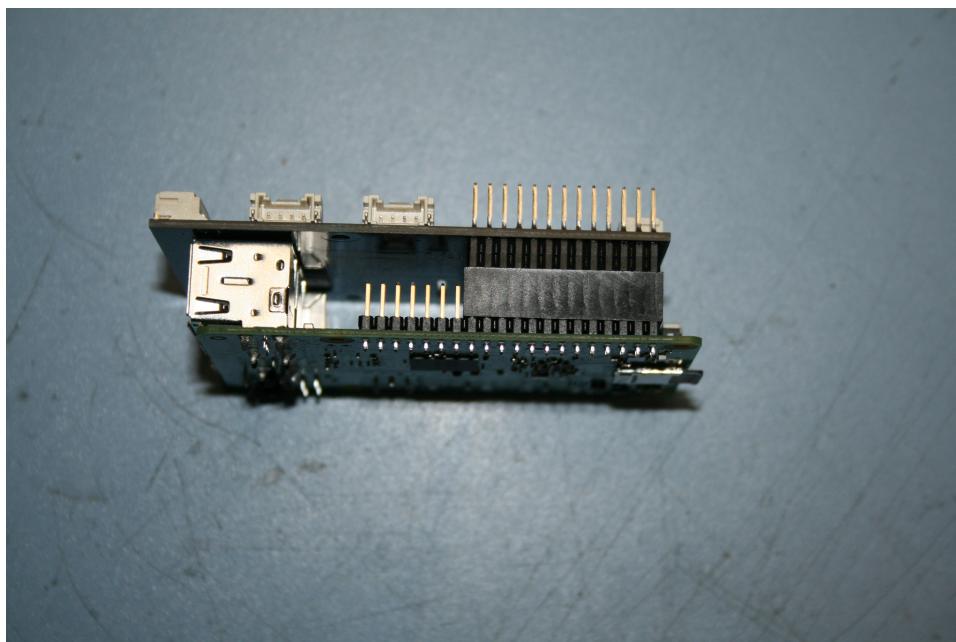


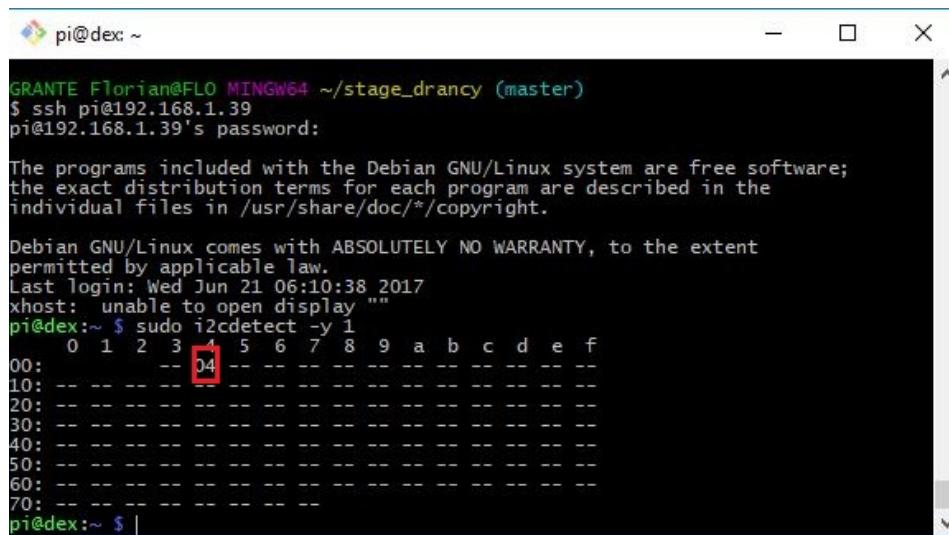
FIGURE 2.8 – La *RaspberryPi* avec le *Shield* branché dessus

6. Brancher à nouveau la *RaspberryPi* au secteur. Vous devriez avoir une LED verte qui s'allume sur votre *Shield*

7. vous allez tester s'il a bien été reconnue, pour cela, connecter vous en "ssh" sur votre *RaspberryPi* (vous devriez savoir le faire maintenant !)
8. lancer la commande :

```
1  sudo i2cdetect -y 1
```

vous devriez obtenir ce résultat, avec le 04 en première ligne.



```
pi@dex: ~
GRANTE Florian@FLO MINGW64 ~/stage_drancy (master)
$ ssh pi@192.168.1.39
pi@192.168.1.39's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jun 21 06:10:38 2017
xhost: unable to open display ""
pi@dex:~ $ sudo i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- 04 -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- --
70: -- -- -- -- -- --
pi@dex:~ $ |
```

FIGURE 2.9 – Test de détection du Shield

Voilà, la première mise en route de la *RaspberryPi* est terminé. Maintenant il faut s'occuper du code pour la station final.

2.3 Installation, configuration et test du code de la station avec ses capteurs.

2.3.1 Installation des capteurs.

Votre *RaspberryPi* est configurée, ainsi que son *Shield*. Il faut installer les différents capteurs. Regardons de plus près les différents connecteurs.

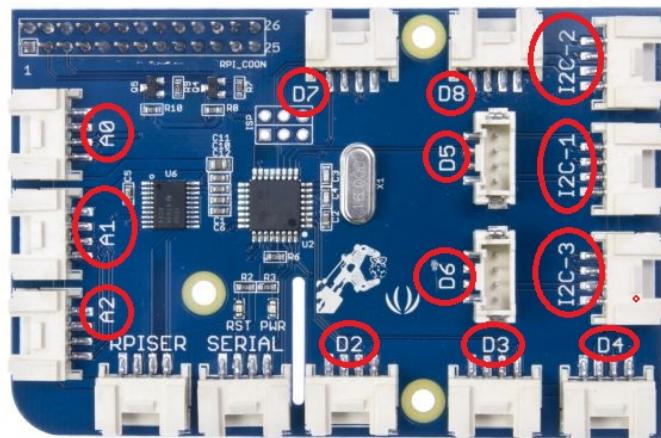


FIGURE 2.10 – vu de dessus du shield

Comme vous pouvez le voir, il y a écrit un numéro d'identification du connecteur sur le *Shield*. Vous allez donc brancher sur des ports particulier en adéquation avec le code qui sera téléchargé ultérieurement.

Procéder aux branchements suivants :

1. Le capteur de température et d'humidité (DHT11) sur le port D7.
2. Le capteur de luminosité sur le port A1.
3. L'encodeur rotatif sur le port A2.
4. L'écran LCD sur un des ports I2C.

Vous devriez alors obtenir un résultat similaire à celui là :

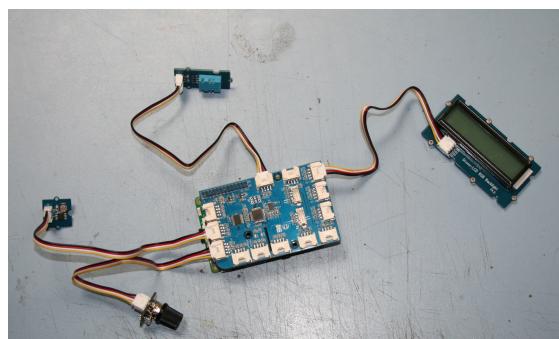


FIGURE 2.11 – Branchement des composant sur le shield

2.3.2 Configuration de la *RaspberryPi*.

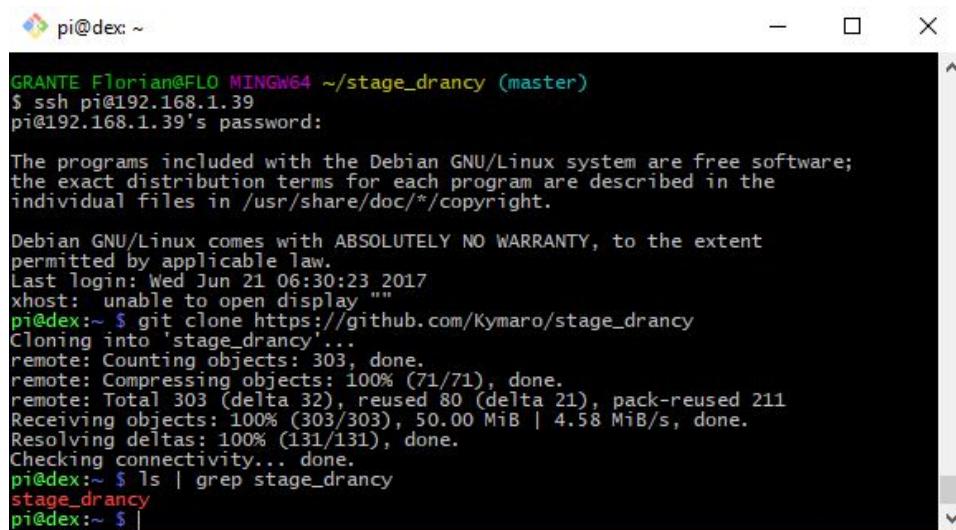
Vous allez maintenant télécharger le code pour activer la station. Pour cela, brancher votre *RaspberryPi* au secteur et au réseau si jamais vous aviez enlevé le câble ethernet. Connectez vous en *ssh* via *git bash* et exécutez les deux commandes :

```
1 cd  
2 git clone https://github.com/Kymaro/stage_drancy
```

Cette commande va télécharger le code nécessaire au fonctionnement de la station. Vous modifierez ce code dans le troisième chapitre pour envoyer les données sur le *Cloud*.

Tapez la commande :

```
1 ls | grep stage_drancy
```



```
pi@dex: ~  
GRANTE Florian@FLO MINGW64 ~/stage_drancy (master)  
$ ssh pi@192.168.1.39  
pi@192.168.1.39's password:  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed Jun 21 06:30:23 2017  
xhost: unable to open display ""  
pi@dex:~ $ git clone https://github.com/Kymaro/stage_drancy  
Cloning into 'stage_drancy'...  
remote: Counting objects: 303, done.  
remote: Compressing objects: 100% (71/71), done.  
remote: Total 303 (delta 32), reused 80 (delta 21), pack-reused 211  
Receiving objects: 100% (303/303), 50.00 MiB | 4.58 MiB/s, done.  
Resolving deltas: 100% (131/131), done.  
Checking connectivity... done.  
pi@dex:~ $ ls | grep stage_drancy  
stage_drancy  
pi@dex:~ $ |
```

FIGURE 2.12 – Résultat de la commande *ls* avec *grep*

Si vous avez bien ce résultat, cela signifie que le code à bien été téléchargé et qu'il est présent dans le dossier *stage_drancy*. Si cela n'est pas le cas, ressayez la commande précédente.

Puis il faut faire en sorte que le script soit lancé au démarrage de la *RaspberryPi*. Pour cela, il faut modifier un fichier qui gère les commandes système qui sont lancé lorsque l'*OS* démarre.

Exécuter la commande :

```
1 sudo nano /etc/rc.local
```

Un fichier va alors s'ouvrir dans votre terminal, vous ne pouvez pas utiliser la souris dans cet éditeur de texte. Descendez alors avec les flèches directionnelles jusqu'à l'avant dernière ligne, la dernière ligne étant normalement *exit 0*. Allez à la fin de la ligne et appuyez alors sur Entrée pour ajouter une nouvelle ligne. Tapez alors sur cette nouvelle ligne :

```
1 sudo python /home/pi/stage_drancy/rpi/test_complet.py &
```

L'esperluette permet d'exécuter le script en tâche de fond, il est important de ne pas l'oublier sinon votre *RaspberryPi* ne sera pas capable de faire autre chose que de lancer votre script.

Pour quitter et enregistrer, taper sur la combinaison de touche une à une :

```
1   Ctrl + X  
2   Y  
3   Entrée
```

Vous êtes désormais prêt à tester la station.

2.3.3 Test de la station

Vous allez tout d'abord essayer la station en lançant le script manuellement pour vérifier que cela fonctionne, puis nous redémarrerons la RaspberryPi pour voir si la modification de l'étape précédente est fonctionnelle.

Tapez la commande suivante pour lancer le script python.

NOTE : Lorsque vous indiquez le chemin d'un fichier en Bash, vous avez votre meilleur ami qui est là pour vous aider et qui s'appelle l'auto-complétion. En effet, lorsque vous commencer à écrire le nom d'un fichier, par exemple station.cloud, il vous suffit d'appuyer sur la touche Tabulation pour voir les suites complter. Si cela ne se complte pas, alors il faut appuyer sur la touche Espace.

```
1   cd  
2   sudo python stage_drancy/rpi/test_complet.py &
```

Cela ne vous rappelle rien ? Il s'agit ni plus ni moins de la ligne qui a été ajoutée dans le fichier /etc/rc.local pour démarrer automatiquement le script. (il manque /home/pi/ puisque nous sommes dans ce dossier par défaut).

Vous devriez voir l'écran LCD de la station qui s'allume avec le message "Bienvenue dans l'IoT Hub" pendant quelques secondes puis un des menus qui affiche les données des capteurs. Si cela reste sur le premier message, tourner l'encodeur rotatif pour afficher les menus.

Si vous voyez bien tous les menus avec une valeur, on y est, la station fonctionne. Vérifiez maintenant qu'elle démarre bien en même temps que la RaspberryPi. Mettez-vous dans le scénario où elle a été coupée électriquement. Taper la commande :

```
1   sudo shutdown now
```

Patienter quelques instants, débrancher puis rebrancher là. Vous devriez alors voir l'écran s'allumer comme pour le test précédent au bout d'un certain temps.

Et voilà, vous êtes au terme de ce chapitre, vous avez désormais une station fonctionnelle localement. Je vous invite alors à suivre le chapitre trois pour configurer Microsoft Azure d'une part puis pour modifier le code de tel sorte qu'il envoie les messages sur le Cloud Azure.

Chapitre 3

Configuration de *Microsoft Azure*

3.1 Mise en place pour 1 station

Vous allez abandonner notre RaspberryPi pendant quelques instant pour se focaliser sur la création d'une instance dans le Cloud pour que vous puissiez envoyer les données.

Pour réaliser cette partie, munissez vous de vos identifiant Microsoft Azure.

3.1.1 Explication des fonctionnalités proposées par *Microsoft*

Avant de commencer, nous allons détailler un peu plus précisément le cheminement que vont avoir nos données. Comme souvent une image vaut mieux qu'un long discours, je vous laisse observer :

Nous allons expliciter deux méthodes différentes de gestion de Azure pour vous montrer le champs des possible. En effet, vous pouvez utiliser le portail qui nous est proposé par Microsoft ou bien utiliser ce qui s'appelle L'Azure CLI pour Azure Command Line Interface.

3.1.2 Crédation d'un Hub d'évènements sur *Azure* avec la *CLI*

3.1.2.1 Installation de l'Interface en Ligne de Commande de *Azure*

Pour cette partie il n'est donc pas nécessaire de se rendre sur le portail. Mais avant toute chose, vous devez installer l'interface en ligne de commandes d'Azure.

1. Installer [Node.js](#) en choisissant l'extension .msi pour x64 ou x32 selon si votre système est en 64bit ou 32bit.
2. Lancer un terminal git bash.
3. Tapez les deux commandes suivantes pour vérifier que Node.js et NPM (Node Package Manager) a bien été installé. NPM est l'utilitaire qui va nous permettre d'installer Azure

```
1 node -v  
2 npm -v
```

Vous devriez avoir ce résultat :

4. Ensuite entrer la commande :

```
1 npm install -g azure-cli
```

Et voilà, Azure est installé, vous devriez pouvoir le lancer en tapant azure dans votre terminal git bash. Voici ce que vous devriez obtenir :

3.1.2.2 Crédation d'un groupe de ressources et d'un Hub d'évènement

Maintenant, il va falloir créer nos instance qui vont récupérer les valeurs envoyée par notre station.

1. Connecter vous avec votre compte Azure :

```
1 azure login
```

Aller alors sur la page : <https://aka.ms/devicelog> pour entrer le code qui vous sera spécifié sur votre terminal. Entrer ensuite vos identifiants. Vous devriez alors être connecté.

2. Taper la commande :

```
1    azure group list
```

Vous verrez alors la liste des groupes de ressources déjà existant. Vous allez alors créer un groupe de ressources par la commande : **NOTE : POUR LE DOSSIER, LE GROUPE A POUR NOM IoT. VOUS POUVEZ RENSEIGNER LE NOM QUE VOUS SOUHAITEZ**

```
1    azure group create IoT westeurope
```

westeurope détermine le serveur où le groupe sera créé.

Si vous refaites la commande de l'étape N°2, vous devriez voir apparaître votre nouveau groupe fraîchement créé sur la liste.

3. Maintenant, vous allez devoir télécharger le code de la station, comme vous l'avez fait pour la RaspberryPi dans le chapitre 2. En effet, il y a un fichier qui sert de patron pour les paramètres à appliquer pour la commande qui suivra. Ainsi, entrer les commandes :

```
1    cd
2    git clone XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

4. Maintenant, vous pouvez faire la commande :

```
1    azure group deployment create -g IoT -n deployment1 -f ~/stage_drancy/IoT.json
```

Vous allez être invité à entrer différentes informations. Les informations que vous allez rentrer seront indispensable car devront être ajouté au code de la station pour pouvoir envoyer les données. Les valeurs sont des chaînes de caractère, celle mise en photo sont arbitraires pour le dossier, veuillez entrer une valeur différente.

- (a) *namespaceName* : Il est conseillé de rajouter NS à la fin pour vous y retrouver ensuite.
- (b) *eventHubName* : Il s'agit de l'instance à proprement parlé qui va recevoir les données. vous pouvez par exemple entrer une valeur du type "Station".
- (c) *consumerGroupName* : Pour celle valeur, je vous conseille d'entrer la même que pour *namespaceName* mais de remplacer NS par GN.

Le déploiement peut prendre un peu de temps. Si cela ne fonctionne pas, retenter en entrant des nom différents. Une fois le déploiement terminé. Plusieurs valeurs sont à noter et sauvegarder précieusement.

Notez d'une part la valeur des trois champs que vous avez indiquer !. De plus, il vous faut noter la valeur de SharedAccessKeyName et SharedAccessKey

Nous en avons terminé avec la création de l'instance qui va récupérer les données envoyé par la station. Il faut maintenant vous occuper de l'instance qui va traiter les données reçus.

3.1.3 Crédit d'un portail Time Series Insights

Microsoft à pensé à nous puisque il a créé un module qui s'occupe seul de traiter les données d'une source pour ensuite créer des courbes et ainsi voir le suivi dans le temps. Sans cela, le processus aurait été nettement plus long puisqu'il aurait fallut d'une part stocké ces valeurs pour ensuite les traiter et devoir créer une interface qui puisse afficher les valeurs.

Vous allez pour cette partie nous rendre sur le [Portail Azure](#). Nous allons dans un premier temps jeter un coup d'œil sur le groupe que vous avez créé dans la partie d'avant.

Cliquer à gauche sur Groupes de ressources. Vous devriez voir apparaître un Hub D'évènement qui porte le nom de votre *namespaceName* de la partie précédente. Maintenant si vous cliquez sur ce Hub d'évènement, en bas de la vue d'ensemble devrait apparaître un Hub d'évènement dont le nom est celui que vous avez entré pour le champs *eventHubName* précédemment. Une fois n'est pas coutume, continuons notre jeu des poupées russes et cliquer dessus. Une fois encore, vous devriez voir apparaître en bas deux groupe de consommateur. Le premier, Default est.... celui utilisé par défaut si jamais vous ne voulez pas créer de groupe de consommateur. Mais c'est toujours mieux d'en avoir un dont on connaît le nom par soucis de visibilité. Le second est celui que

vous avez créé tout à l'heure.

Maintenant que vous êtes rassuré sur les lignes de commande de la partie précédente nous allons pouvoir créer notre instance de traitement des données.

1. Cliquer sur le + en haut à gauche de votre portail.
2. Dans la barre de recherche du menu qui vient de s'ouvrir, entrer "Time Series"
3. Choisissez alors "Time Series Insights (aperçu)

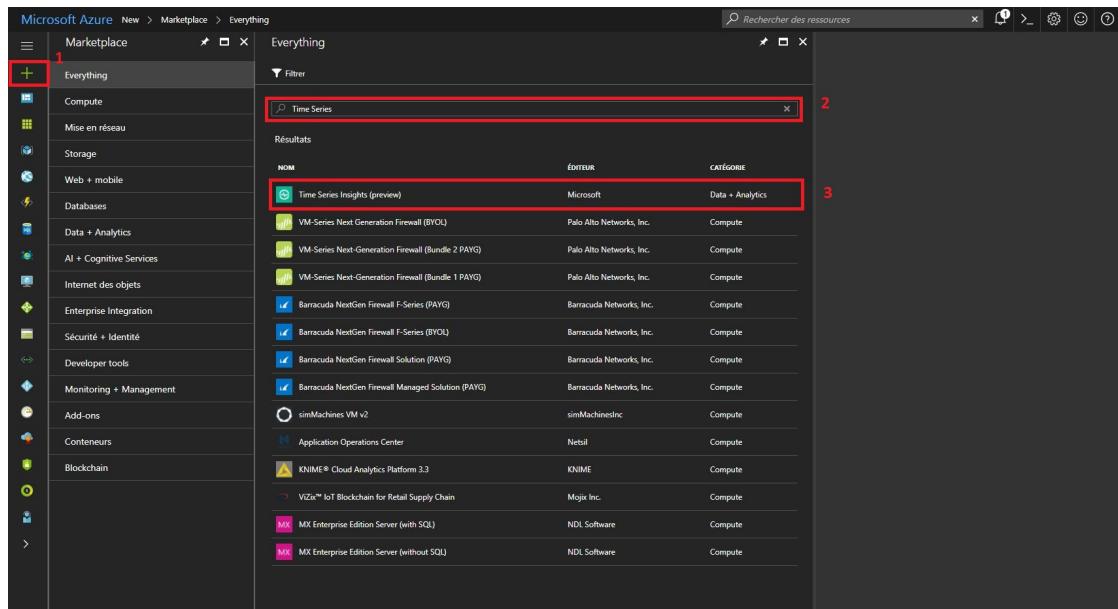


FIGURE 3.1 – Menu de recherche de ressources sur Azure

4. Cliquer alors sur Créer.
Vous arrivez alors sur la fenêtre suivante :

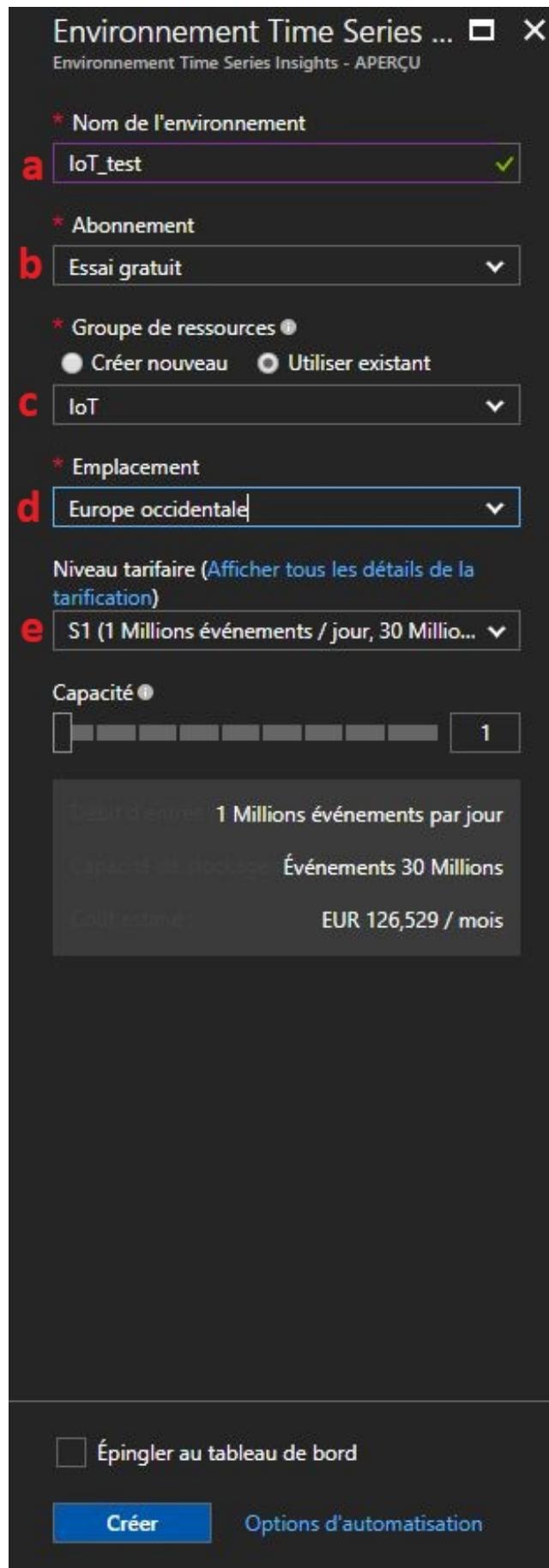


FIGURE 3.2 – Menu de recherche de ressources sur Azure

- (a) Il s'agit du nom que vous voulez donner à votre Portail. Ce nom apparaîtra sur le Portail, donc préférez un nom qui vous indique les données qui vont y être envoyées.
- (b) Choisissez votre abonnement Azure.
- (c) Choisissez le groupe de ressources que nous avons créé tout à l'heure. Pour le dossier, il s'agissait du groupe "IoT".
- (d) Sélectionnez "Europe Occidentale". C'est le serveur sur lequel va être implanté le portail.
- (e) Pour ce choix, cela dépendra de votre besoin. Sachez que ce choix détermine le tarif du portail. Pour information, la station envoie ses données environ toutes les deux minutes. Il n'était pas possible de mettre un délai plus long auquel cas il y aurait des problèmes de déconnexion de la station. Partez du principe qu'une station envoie 1000 événements par jour, cela vous donne une indication quant aux nombres de stations que vous pouvez implémenter sur le portail.
- (f) Cliquer alors sur Créer.

Voilà, votre portail est en cours de déploiement, nous allons maintenant devoir faire en sorte que cette instance écoute les valeurs que l'on envoie sur le Hub d'événement.

3.1.4 Ajout d'une source d'événements dans le portail *Time Series Insights*

Maintenant que vous avez créé votre instance Time Series Insights, cliquer dessus pour voir sa page principale affichée :

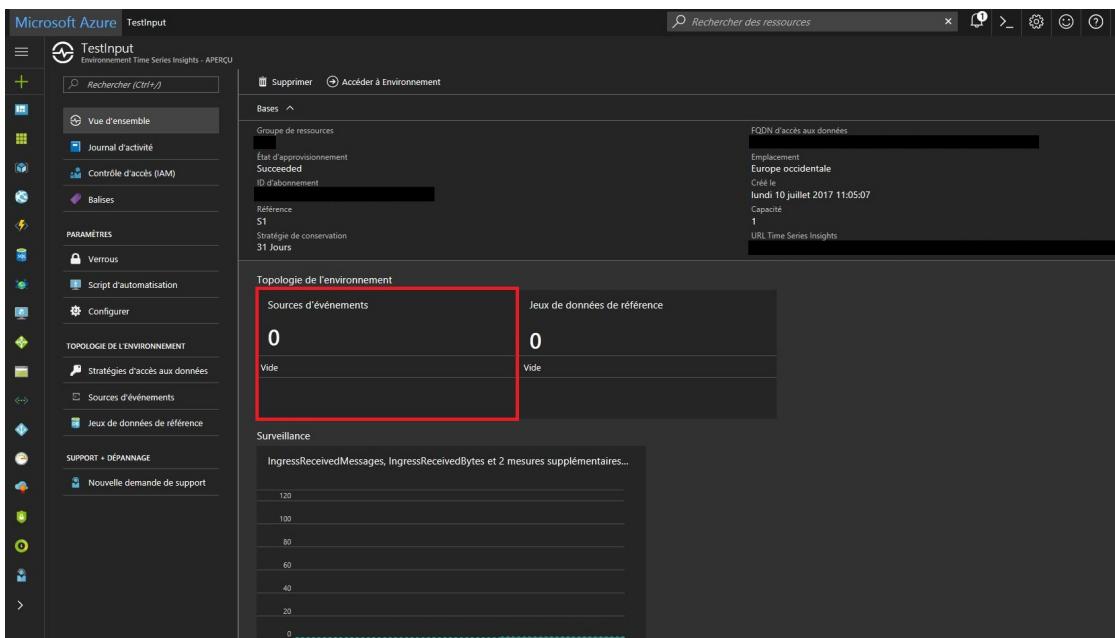


FIGURE 3.3 – Menu principal du *Time Series Insights*

1. Cliquer sur la zone Sources d'événements puis sur Ajouter en haut dans la fenêtre qui s'ouvrira.
2. vous devriez alors voir apparaître cette fenêtre :

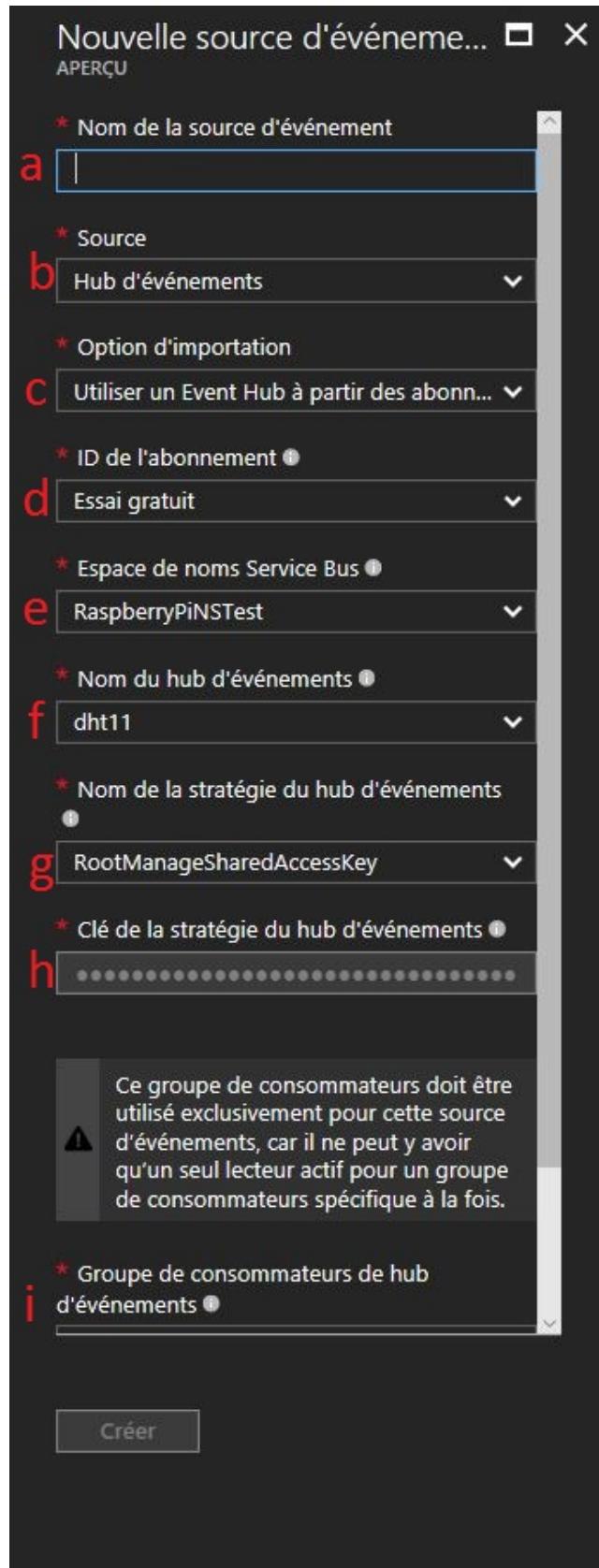


FIGURE 3.4 – Ajout d'une source d'évènements

- (a) C'est le nom que vous voulez donner à votre source. Le choix est arbitraire.
- (b) Choisir Hub d'évènements.
- (c) Choisi Utiliser un Event Hub à partir des abonn..
- (d) Ici, le champs dépend de votre abonnement.

- (e) Pour les étapes qui vont suivre, il sera nécessaire de reprendre les champs indiqué lors de la section 4 page 16. Pour le premier, entrer le namespaceName.
- (f) Ici doit être indiqué l'evenHubName.
- (g) Le choix RootManageSharedAccessKey devrait être proposé. Il s'agit probablement de votre SharedAccessKeyName. Sinon, renseigner votre SharedAccessKeyName
- (h) Renseigner votre SharedAccessKey
- (i) Ici doit être entré votre consumerGroupName.
- (j) laisser ce dernier champs vide et cliquer sur Créer.

Vous avez donc ajouté le hub d'évènement comme source pour l'instance de Time Series Insights. Il reste alors à éditer le code de la station pour indiquer le hub sur lequel envoyer les données.

3.1.5 Configuration logiciel de la station pour Azure

Avant de vous montrer les modifications à réaliser dans le code, quelques informations concernant celui ci. Le code est écrit dans le langage **Python**. En plus d'avoir le mérite d'être considéré comme un des langages les plus haut niveau (ie. proche du langage de l'Homme et donc intuitif), il est particulièrement adapté pour l'utilisation de script automatisé comme vous allez le faire pour la station.

ATTENTION : Avant de vous faire découvrir le code et de vous expliquer les lignes à modifier pour pouvoir envoyer vos données, une petite explication sur l'architecture d'un code Python. Il est impératif de respecter les indentations (espacement) du code. Si jamais vous rajouter ne serait-ce qu'un espace sur une ligne, cela peut avoir comme effet de faire planter l'intégralité de la station. Si cela ce produit, pour simplifier le problème, reportez vous à l'annexe N°1.

Il reste cependant une étape avant de modifier le code. En effet, il est nécessaire d'installer la bibliothèque azure sur la RaspberryPi. Pour cela, connectez vous en ssh dessus puis entrer la commande :

```
1 sudo pip install azure-servicebus
```

Nous pouvons désormais regarder le code :

```
1 ##### IMPORTATION DES BIBLIOTHEQUES#####
2 from grovepi import *
3 from grove_rgb_lcd import *
4 import time,datetime,json
5 from math import *
6 #from azure.servicebus import ServiceBusService
7 #from azure.servicebus import Message
8 ##### DECLARATION DES CONSTANTES GLOBALES#####
9 dht_sensor_port = 7 # capteur humidite et temperature (DHT11 ou DHT22) sur D7
10 dht_sensor_type = 0 # mettre 0 si bleu (DHT11) ou 1 si blanc (DHT22)
11 t_actuator = 799 # nombre de loop entre deux envoie de donnees
12 t_wait = 25
13 lum_seuil = 10 # seuil au dessus du quel on determine la lumiere comme allume
14 lum_sensor = 1 # capteur de lumiere sur A1
15 potentiometer = 2 # bouton de menu sur A2
16 ##### ACTIVATION DES PORTS DU SHIELD GROVEPI#####
17 pinMode(potentiometer,"INPUT")
18 pinMode(lum_sensor,"INPUT")
19 time.sleep(1)
20 ##### VARIABLES INTERNES#####
21 t_refresh = 800 # nombre de loop entre deux envoie a l instant t
22 compteur_echec_envoie = 0 # compteur d echec d envoie de donnee sur azure (se remet a 0 si reussite)
23 mode_value = 0 # les deux valeurs permette d eviter une actualisation inutile de l ecran qui le faisait clignoter
24 mode_value_old = 0
25 ##### VARIABLES DE DONNEES ENVOYEEES#####
26 temp_dht = 0
27 hum = 0
28 lum = 0
29 identifiant = "Rpi Test" # a modifier
30 ##### FONCTIONS#####
31 def DHT() : #temperature et humidite numerique
32     global temp_dht,hum
33     [temp_dht,hum] = dht(dht_sensor_port,dht_sensor_type)
34     temp_dht = round(temp_dht,1)
35
36 def Luminosite() : #luminosite qui envoie True ou False avec le seuil
37     lum_value = analogRead(lum_sensor)
38
39 def screen_administrator() : # permet de gerer lecran sans quil refresh a chaque iteration
    global mode_value,temp_dht,hum
```

```

41     encoder_value = analogRead(potentiometer)
42     mode_value_old = mode_value
43     if (encoder_value <=341 and encoder_value >= 0) and mode_value != 1 : #MODE 1
44         setText("Temperature : \n"+str(temp_dht))
45         setRGB(0,128,255)
46         mode_value = 1
47     elif (encoder_value <= 682 and encoder_value > 341) and mode_value != 2 : #MODE 2
48         setText("Humidite : \n"+str(hum))
49         setRGB(255,0,128)
50         mode_value = 2
51     elif (encoder_value > 682 and encoder_value <=1023) and mode_value != 3 : # MODE 3
52         setText("Luminosite \n")
53         setRGB(255,128,0)
54         mode_value = 3
55     if not ((mode_value - mode_value_old) != 0) : #s il y a pas eu un changement de mode sur l ecran
56         time.sleep(140.0/1000.0) #on attend 140 ms pour etre sur du temps a chaque loop
57 """
58 def createSBS() : #permet de creer le canal de communication avec Azure
59
60     service_namespace = 'RaspberryPiNSTest' #a modifier
61     key_name = 'RootManageSharedAccessKey' # a modifier
62     key_value = 'yEcs0927kFYs1U8J7x4VCwZAaf3ck38hqSGY9YjiMAo=' # a modifier
63
64     sbs = ServiceBusService(service_namespace, shared_access_key_name=key_name, shared_access_key_value=key_value)
65
66     return sbs
67 """
68 #####SETUP#####
69 #sbs = createSBS()
70 setText("Bienvenue\n dans l 'IoT Hub")
71 setRGB(128,255,0)
72 time.sleep(2)
73 #####BOUCLE INFINI#####
74 while True :
75     if ( t_refresh >= t_actuator ) :
76         DHT()
77         while (isnan(temp_dht) or temp_dht == 0) : # on essaie tant que le capteur n a pas de valeur valide
78             DHT()
79             dt = str(datetime.datetime.now())
80             d = {'DeviceID' : identifiant, 'Temperature' : temp_dht, 'Humidity' : hum,'Time' : dt }
81             msg = json.dumps(d) #cree le message a envoyer
82             try :
83                 #sbs.send_event('dht11',msg) #a modifier
84                 compteur_echec_envoie = 0
85             except :
86                 compteur_echec_envoie += 1
87                 if compteur_echec_envoie == 720 : #echec d envoie de message depuis 24h
88                     setRGB(255,0,0)
89                     setText("Probleme envoie\nmessage azure")
90                     break # sort de la boucle, reboot necessaire du programme ou de la RPI
91                 if (t_refresh >= t_wait) : # on attend un peu avant de refresh l ecran
92                     screen_administrator()
93                 t_refresh += 1

```
