



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>	2
1.3 Les composants	3
1.3.1 Le capteur de température et d'humidité	3
1.3.2 Le capteur de luminosité	3
1.3.3 L'écran	3
1.3.4 L'encodeur rotatif	3
2 Configuration de la station	4
2.1 Création et installation de la distribution <i>Raspbian</i>	4
2.2 Première mise en route de la <i>RaspberryPi</i>	5
2.3 Installation, configuration et test du code de la station avec ses capteurs.	7
2.3.1 Installation des capteurs.	7
2.3.2 Configuration de la <i>RaspberryPi</i>	7
2.3.3 Test de la station	7
3 Configuration de <i>Microsoft Azure</i>	8
3.1 Mise en place pour 1 station	8
3.1.1 Explication des fonctionnalités proposé par <i>Microsoft</i>	8
3.1.2 Crédit d'un Hub d'évènements sur <i>Azure</i> avec la <i>CLI</i>	8
3.1.2.1 Installation de l'Interface en Lignes de Commande de <i>Azure</i>	8
3.1.2.2 Crédit d'un groupe de ressources et d'un Hub d'évènement	8
3.1.3 Crédit d'un portail <i>Time Series Insights</i>	9

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 notre 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. Son autre atout et non des moindre, 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 propre qui se branche sur la *RaspberryPi* pour communiquer avec elle. Ainsi, nous pouvons utiliser les fonctions du "*Shield*" en envoyant des instruction de puis 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*.

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 :

- port analogique : Ils sont disponible 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 10bits ($0V = 0$ et $5V = 1023$). Exemple si on envoie 2.5V, alors on aura la valeur 512)
- port digital : ceux ci peuvent être régler 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.

- port 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

Datasheet : [%urlici](#)

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 allumé ou non. Nous détaillerons un protocole à réaliser pour essayer de faire marcher au mieux ce capteur. Ce composant utilise un connecteur analogique

Datasheet : [%urlici](#)

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 caractère sur ce dernier et choisir la couleur de fond qui est affiché. Ce composant utilise un connecteur I2C. Datasheet : [%urlici](#)

1.3.4 L'encodeur rotatif

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

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 nulle autre qu'une carte Micro SD. C'est donc sur ce support que nous allons faire l'installation.

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

- d'une *RaspberryPi*
- d'une carte micro SD de minimum 4Go
- 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.

Nous pouvons desormais commencer l'installation de l'OS sur notre *RaspberryPi*

1. Connecter la micro SD sur votre ordinateur.

2. Lancer le logiciel *Etcher*

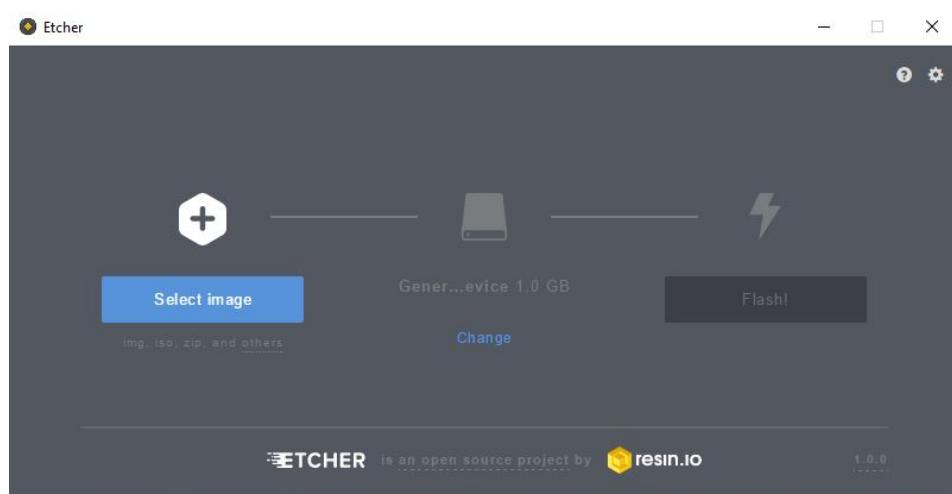


FIGURE 2.1 – Logiciel *Etcher* au lancement

3. Cliquer sur "*Select image*"  gauche puis selectionner le fichier .zip recupr depuis le site *SourceForce* dans les pr-requis.
4. Vrifer que le periphrique qui est renseign au centre est bien la micro SD. Dans le cas contraire cliquer sur "*Change*".
5. Si les deux tapes pr-cdentes sont OK, cliquer sur "*Flash*".

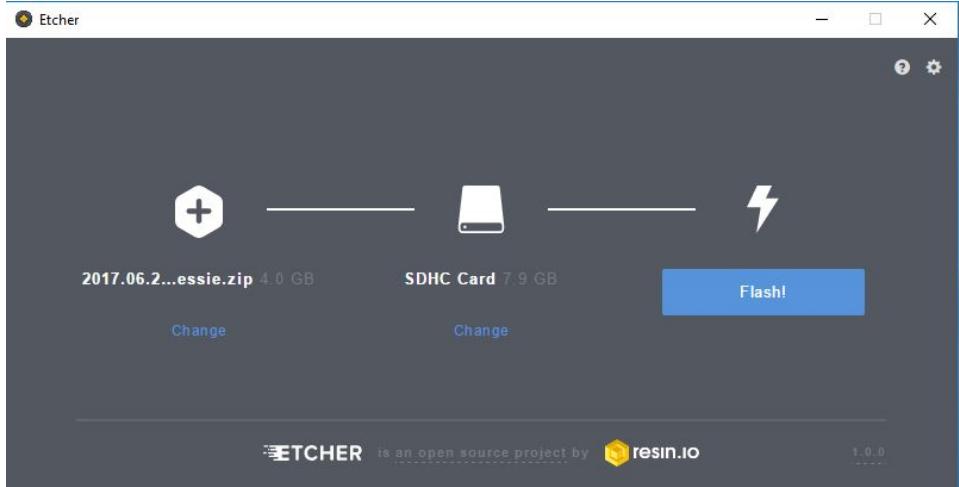


FIGURE 2.2 – *Etcher* prêt à flasher

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*, nous pouvons démarrer notre nouvel ordinateur. Pour cela, il vous suffit d'insérer la microSD au dos de la *RaspberryPi*.



FIGURE 2.3 – La *RaspberryPi* de dos

2. Ensuite, nous allons pouvoir lancer la *RaspberryPi*. 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.
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*.
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"

4. Très bien, désormais que vous avez l'adresse IP à disposition, nous allons pouvoir installer le nécessaire pour utiliser nos capteurs. Nous aurions très bien pu continuer cette installation directement sur la *RaspberryPi* mais si vous n'avez jamais fait ce qui va suivre, cela vous fera un bon entraînement.

- (a) Télécharger le logiciel [Git Bash](#)
- (b) Lancer *Git Bash*
- (c) 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 faites confiance, taper alors "yes" puis sur *Entrée*

- (d) Le mot de passe est "robots1234". Si tout c'est bien passé, vous devriez avoir cet aperçu :
- (e) vous naviguez maintenant dans la *RaspberryPi*. Dans un premier temps, nous allons changer le mot de passe car celui ci est un mot de passe par défaut. Enter la commande :

```
1 sudo raspi-config
```

un écran bleu devrait apparaître.

- (f) Quitter le menu pour revenir au terminal.
- (g) Maintenant, nous allons installer GrovePi+ pour pouvoir utiliser le *Shield*. Entrer alors les deux commandes suivantes :

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

Votre *RaspberryPi* va redémarrer.

- (h) Connecter 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 /home/pi/Desktop
2 sudo git clone https://github.com/DexterInd/GrovePi
3 cd /home/pi/Desktop/GrovePi/Script
4 sudo chmod +x install.sh
5 sudo ./install.sh
```

- (j) Appuyez à nouveau sur *Entrée* une fois arrivé sur cette configuration :
- (k) Au moment où la *RaspberryPi* va redémarrer (vous verrez un "Restart" écrit dans le terminal. Appuyez sur *Ctrl + C* pour empêcher le redémarrage.
- (l) Effectuer alors la commande :

```
1 sudo shutdown now
```

Elle va alors s'arrêter. Vous pouvez alors la débrancher une fois que vous avez un écran noir.

5. Vous pouvez désormais ajouter le *Shield* sur la *RaspberryPi* Comme ci-dessous **ATTENTION AU BROCHES UTILISÉES SUR LA PHOTO !**
6. Brancher à nouveau la *RaspberryPi*. Vous devriez avoir une LED qui s'allume sur votre *Shield*

7. nous allons 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.

Voilà, la première mise en route de la *RaspberryPi* est terminé, nous allons maintenant pouvoir nous 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*. Nous allons maintenant pouvoir installer les différents capteurs. Regardons de plus près les différents connecteurs.

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 télécharger ultérieurement.

Vous allez faire les branchements suivants :

1. Le capteur de température et d'humidité 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 :

2.3.2 Configuration de la *RaspberryPi*.

Nous allons maintenant télécharger le code pour activer le 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 la commande :

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

Tapez la commande :

vous devriez voir apparaître un dossier Si cela n'est pas le cas, réessayer la commande précédente. Nous allons maintenant modifier un fichier dans *Raspbian* pour faire en sorte que le script qui gère les capteurs puisse se lancer au démarrage de la *RaspberryPi*.

Exécuter la commande :

Ajouter alors la ligne :

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 alors la commande suivante pour aller dans le dossier.

Puis taper la commande

Vous devriez voir l'écran 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. Nous allons maintenant vérifier qu'elle démarre bien en même temps que la *RaspberryPi*. Mettons nous dans le scénario où elle a été coupé électriquement. Taper la commande : 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, nous arrivons 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

Nous allons abandonner notre *RaspberryPi* pendant quelques instant pour se focaliser sur la création d'une instance dans notre *Cloud* pour que l'on puisse envoyer les données.

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

3.1.1 Explication des fonctionnalités proposé 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 :

Je vais vous expliciter deux méthodes différentes de gestion de *Azure* pour vous montrer le champs des possible. En effet, nous pouvons 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 Lignes 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. Pour ce faire, lancer un terminal *git bash*.
2. 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 :

3. 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. Nous allons 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 nous l'avons 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, nous pouvons 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 noms 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 envoyées par la station. Nous allons maintenant nous occuper de l'instance qui va traiter les données reçues.

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 stocker ces valeurs pour ensuite les traiter et devoir créer une interface qui puisse afficher les valeurs.

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

Cliquer à gauche sur *Groupes de ressources*. Vous devriez voir apparaître un Hub D'événement qui porte le nom de votre *namespaceName* de la partie précédente. Maintenant si vous cliquez sur ce Hub D'événement, en bas de la vue d'ensemble devrait apparaître un Hub d'événement 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 groupes 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'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)*"