Senso

détecter la déforestation grâce à l'analyse sonore

Andrei Florian (Irland)

L'exploitation illégale du bois est un problème dans de nombreux pays, auquel le projet Senso apporte une solution possible. Le dispositif, basé sur Arduino MKR Fox, alerte les autorités lorsqu'il détecte des bruits d'abattage.

Lorsque je suis allé en Roumanie pour faire un peu d'escalade, j'ai constaté l'hostilité d'une grande partie de la population à l'abattage d'arbres illégal et j'ai pensé que je pouvais peut-être faire quelque chose pour le combattre. Mon étude du problème m'a conduit à identifier cette activité par le son produit par les outils utilisés. Voici la solution que j'ai trouvée.

Le son de la déforestation

La déforestation est l'un des problèmes les plus importants auxquels notre génération est confrontée. Des forêts sont abattues dans le monde entier pour faire place aux cultures et aux habitations. Des pans entiers de forêt sont incendiés pour créer de l'espace et l'exploitation forestière illégale se pratique à l'échelle internationale. La déforestation est directement liée au changement climatique, car la combustion de matières organiques libère beaucoup de dioxyde de carbone, un gaz à effet de serre.

L'exploitation forestière est un problème majeur tant dans les pays en développement que dans les pays développés. La Roumanie en est un bon exemple. Bien que l'exploitation forestière y soit réglementée, une bonne partie de celle-ci se fait de manière illégale. La Roumanie compte 65 % des forêts vierges d'Europe, mais l'abattage les fait rapidement disparaître.

La plupart des coupes sont effectuées avec des tronçonneuses très bruyantes. C'est là qu'intervient le Senso, un appareil de faible consommation équipé d'un module d'analyse sonore capable de séparer les différentes bandes de fréquences sonores. L'appareil est programmé pour identifier le son des outils d'exploitation forestière tels que les tronçonneuses et alerter les autorités compétentes lorsqu'un tel son est détecté. Des écoutes sont effectuées toutes les quinze minutes. Si un son suspect est détecté, l'appareil envoie un message au centre de traitement où les zones d'activités d'abattage apparaissent sur une carte.

Senso est une arme contre l'abattage illégal, car il donne l'alerte en temps réel, ce qui permet aux autorités d'intervenir très rapidement et de se concentrer sur les zones où l'abattage est le plus important.

Aperçu du dispositif

J'ai utilisé l'Arduino MKR Fox (**figure 1**) pour ce projet en raison de ses excellentes caractéristiques de faible consommation. Je l'ai déjà utilisé dans plusieurs projets qui reposent sur la communication à faible puissance. Il est très facile d'utiliser Sigfox sur cet appareil car il est intégré et les bibliothèques Arduino vous permettent d'être opérationnel en un rien de temps.

Pour l'instant, le dispositif nécessite une alimentation 5 V, les modules périphériques consommant trop d'énergie pour être alimentés directement par la carte. Comme il s'agit d'un prototype, la durée de vie

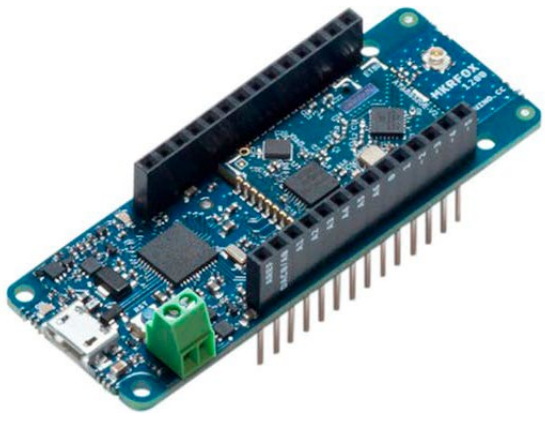


Figure 1. L'Arduino MKR Fox est au cœur de ce projet. (Source : Arduino)

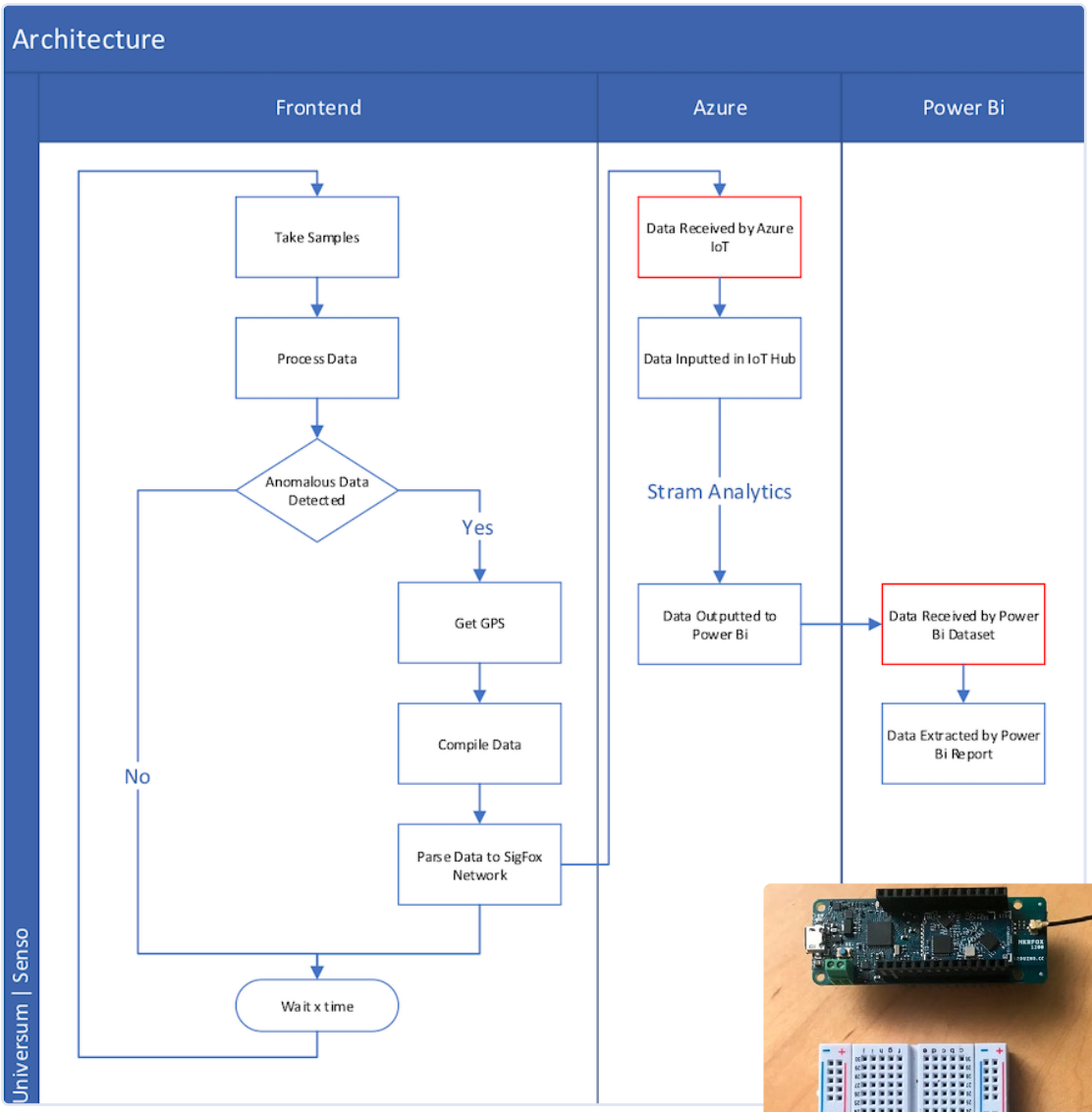


Figure 2. L'architecture du système sous forme d'organigramme.

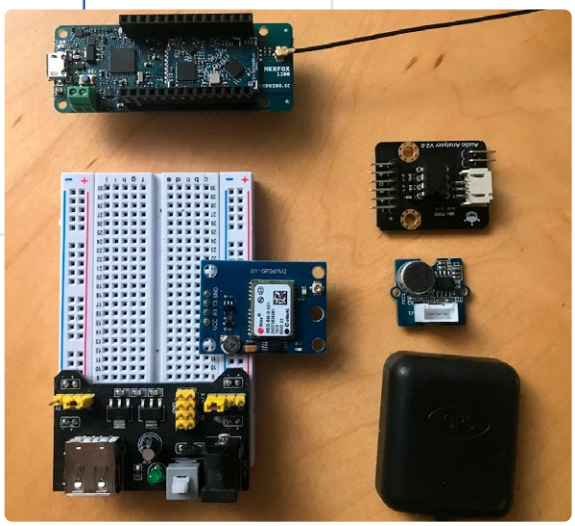


Figure 3. Les modules utilisés dans ce projet.

de la batterie qui l'alimente n'est pas suffisante pour un usage sur le terrain. Je travaille encore à désactiver les capteurs lorsque l'appareil est en mode sommeil afin d'économiser l'énergie.

Architecture du projet

Le projet est divisé en deux parties : un frontal et une base arrière. Le frontal consiste en un dispositif installé en forêt, qui prélève des échantillons de son ; la base arrière utilise Sigfox, Microsoft Azure et Microsoft Power BI pour traiter et afficher les données (**figure 2**).

Frontal : L'appareil se réveille à intervalles réguliers pour écouter l'environnement sonore. Il recherche l'activité dans des bandes de fréquences spécifiques aux tronçonneuses et outils similaires. Si un positif est trouvé, l'appareil envoie sa position et le niveau de sa batterie sur le cloud. Puis il se remet en sommeil jusqu'à l'heure du réveil suivant.

Une fonctionnalité intéressante à ajouter serait le suivi de la position. À l'aide d'accéléromètres, le dispositif pourrait détecter quand l'arbre sur lequel il est fixé est abattu. Dans ce cas, il pourrait envoyer sa position à la base arrière toutes les 10 minutes environ, ce qui permettrait aux autorités de suivre l'arbre abattu pendant son transport.

Base arrière : L'événement détecté par le capteur est reçu par la base arrière Sigfox, qui le transmet à Azure IoT au moyen d'une fonction de rappel. Azure IoT attribue ensuite les données à un hub. Un service de flux continu de données interroge les données et les insère dans un jeu de données Power BI. Enfin, Power BI extrait les valeurs du jeu de données pour créer un rapport qui peut être affiché sur un écran. En outre, une application de messagerie pourrait être utilisée pour envoyer un message aux autorités dès qu'un événement est reçu par la base arrière. Cela permettrait aux intervenants de savoir exactement où se rendre lorsqu'un arbre est abattu.

Réaliser le projet

Assez de discours, au travail ! Ce projet nécessite un certain nombre de modules (**figure 3**) : une carte Arduino MKR Fox, un module GPS et son antenne, une antenne GSM, un module analyseur audio de DFRobot et un microphone. Une carte d'expérimentation est utilisée

pour tout relier (**figure 4**). L'alimentation est fournie par une pile 9 V via un module 5 V/3,3 V. Notez que le microcontrôleur est alimenté par l'alimentation de la carte d'expérimentation.
J'utilise un module GPS bon marché que j'ai trouvé sur internet pour 10 €. Il est facile à utiliser mais nécessite une bonne antenne et met un certain temps à se verrouiller sur les satellites.
Le module analyseur audio fournit le niveau sonore dans sept bandes de fréquence : 63 Hz, 160 Hz, 400 Hz, 1 kHz, 2,5 kHz, 6,25 kHz et 16 kHz. Après des tests approfondis, j'ai constaté que le bruit des tronçonneuses se situe généralement dans la plage de 2,5 kHz à 6,25 kHz. Lors des tests effectués avec des sons de tronçonneuse enregistrés, la bande de 6,25 kHz présentait des pics. J'ai donc pu effectuer un calcul pour éviter que d'autres sources, comme les voitures et les sons naturels, ne soient pris pour des tronçonneuses. Bien qu'encore imparfait, mon algorithme donne de bons résultats.
Le code est divisé en trois sections principales, chacune étant décrite ci-dessous.

Echantillonnage et traitement du son

Cette partie du programme est exécutée à chaque fois que l'appareil se réveille. Le dispositif prélève d'abord des échantillons du capteur sonore, puis traite les données saisies en comparant la valeur de la bande de fréquence cible aux autres bandes. Cette opération est réalisée à l'aide des lignes de code suivantes :

```
long comparison = ((valueMean[0] + valueMean[1] + value-
   Mean[2] + valueMean[3]) / 1.9);
if (valueMean[5] > comparison) ...
```

Les valeurs moyennes de toutes les bandes de fréquences (sauf la bande cible) sont additionnées et divisées par 1,9. Si la valeur de la bande cible est supérieure à `comparison`, la détection est considérée comme positive (**figure 5**).

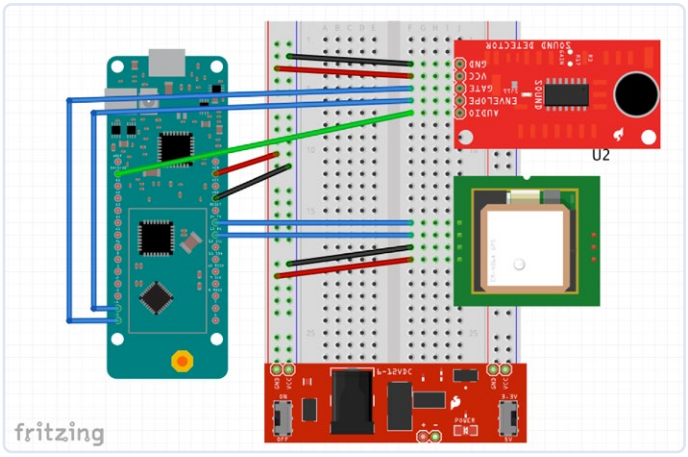


Figure 4. Schéma de câblage du prototype Senso.

Obtenir la position GPS et le niveau de la batterie

La fonction `getGPS` extrait les coordonnées GPS de l'appareil et vérifie qu'elles sont correctes. Ensuite, la tension de la batterie (s'il y a lieu) est mesurée par la fonction `getBatteryVoltage` et ajoutée aux données à envoyer au cloud.
Notez que si la carte Arduino est alimentée par sa broche VIN, la tension de la batterie sera lue égale à zéro. Elle doit être alimentée par ses broches d'alimentation sur la carte d'expérimentation pour une lecture correcte.

Envoyer des données à Sigfox

Les fonctions `encodeData` et `sendToSigfox` codent les données de position et le niveau de la batterie au format octet et les envoient à la base arrière Sigfox.

Configuration du programme

Il y a deux paramètres de configuration qui doivent être renseignés : `proDebug` – mis à *true* (vrai) pour le débogage. Dans ce cas, le dispositif doit être connecté à un PC via une connexion série et la fenêtre

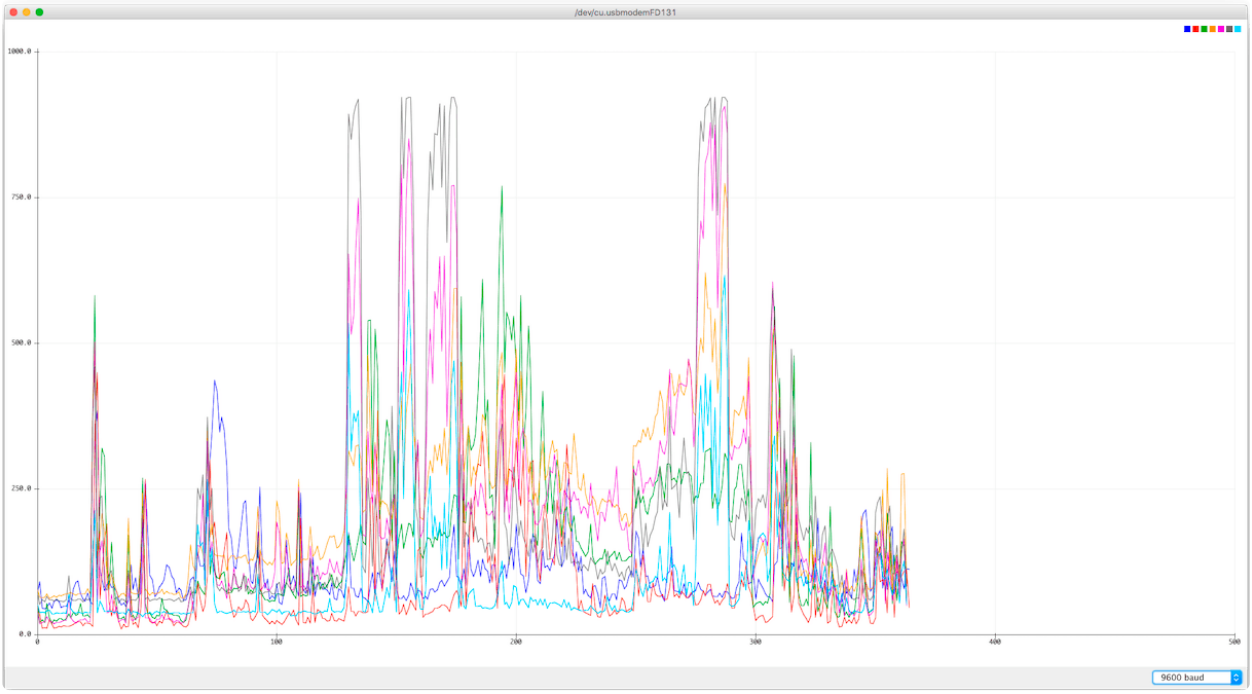


Figure 5. Test sonore. Les pointes montrent l'activité d'une tronçonneuse.

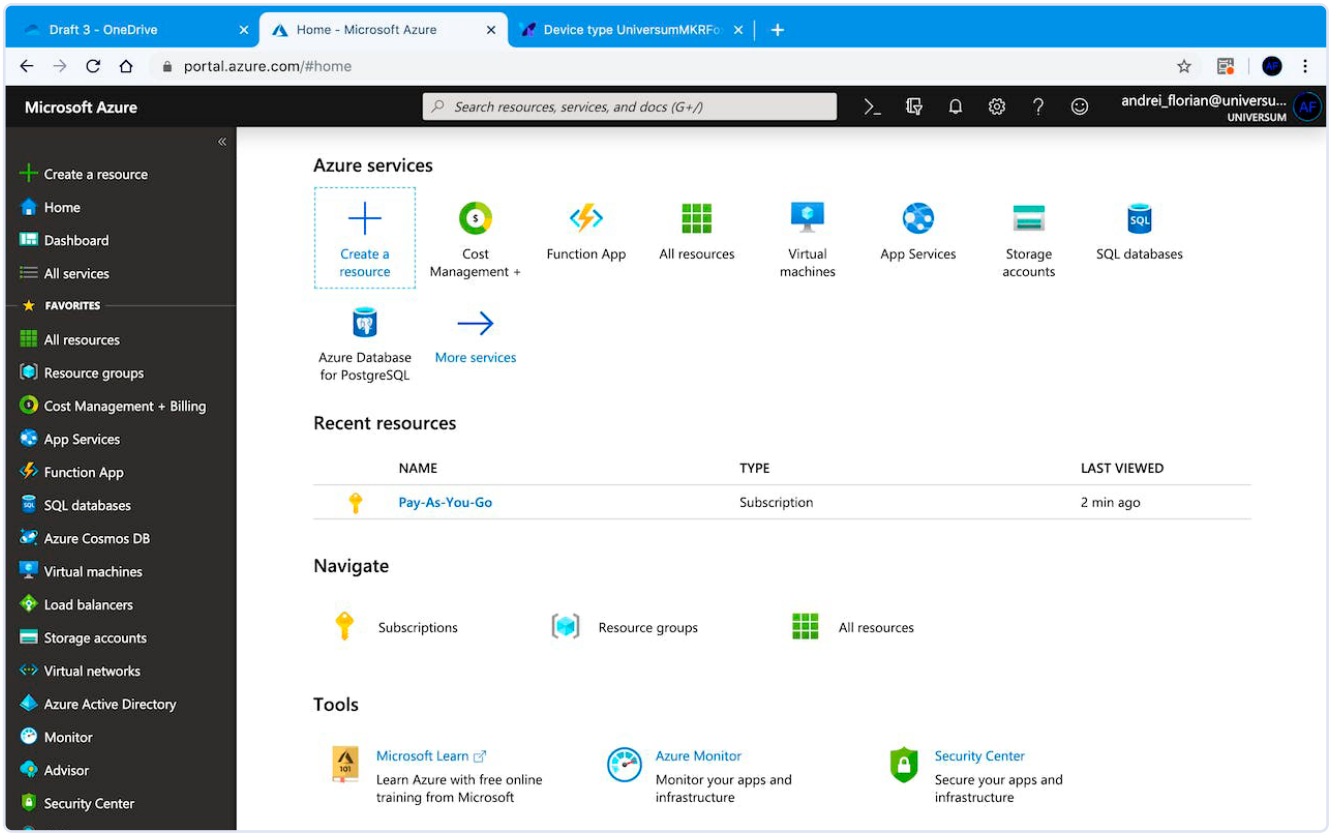


Figure 6. La première étape de la création d'un compte Microsoft Azure. Voir dix-neuf autres captures d'écran sur [1].

Terminal de l'EDI Arduino doit être ouverte pour que le code soit exécuté. Mis à *false* sinon.
`nrSamples` – nombre d'échantillons à prélever. Chaque échantillon est composé de 100 lectures.

Préparer Microsoft Azure

Ce projet utilise Microsoft Azure comme base arrière. Il y a cependant quelques conditions préalables :

- Compte Azure
- Abonnement Azure
- Connaissance de base de l'application

Puisqu'une image vaut mille mots (**figure 6**). Comme vingt captures d'écran prendraient trop de place ici, veuillez vous reporter à [1] pour obtenir des détails sur la configuration d'un compte Azure et du hub IdO qui stockera les données reçues de l'appareil.

Préparer Sigfox

Nous devons également préparer le callback de Sigfox. Encore une fois, il y a quelques conditions préalables :

- Un compte base arrière Sigfox
- Enregistrez le dispositif dans la base arrière

Pour les mêmes raisons que pour la configuration du compte Azure, veuillez vous reporter à [1] pour les détails de la configuration d'un compte Sigfox (**figure 7**).
Pour informer la base arrière Sigfox du format des données que nous envoyons, trois valeurs à virgule flottante (latitude, longitude et niveau de batterie), entrez la ligne suivante dans le champ *custom-data-config* :

```
geoLat::float:32:little-endian geoLng::float:32:little-endian battery::float:32:little-endian
```

Ensuite, remplissez le corps JSON du message avec les données suivantes :

```
{
    "device" : "{device}",
    "data" : "{data}",
    "latitude" : {customData#geoLat},
    "longitude" : {customData#geoLng},
    "battery" : {customData#battery},
```

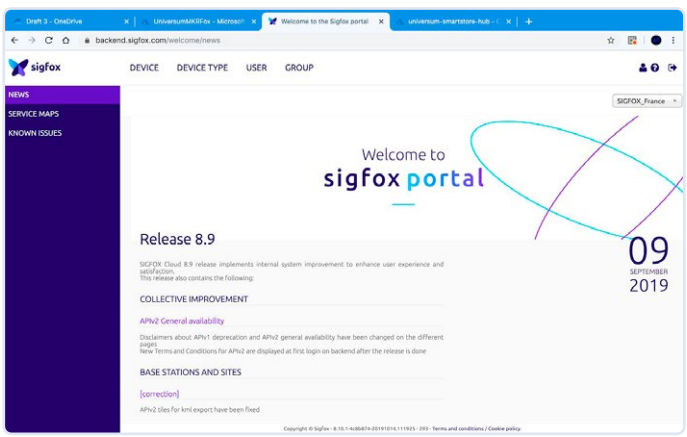


Figure 7. L'ouverture du portail Sigfox est l'étape 1. Voir neuf autres étapes sur [1].

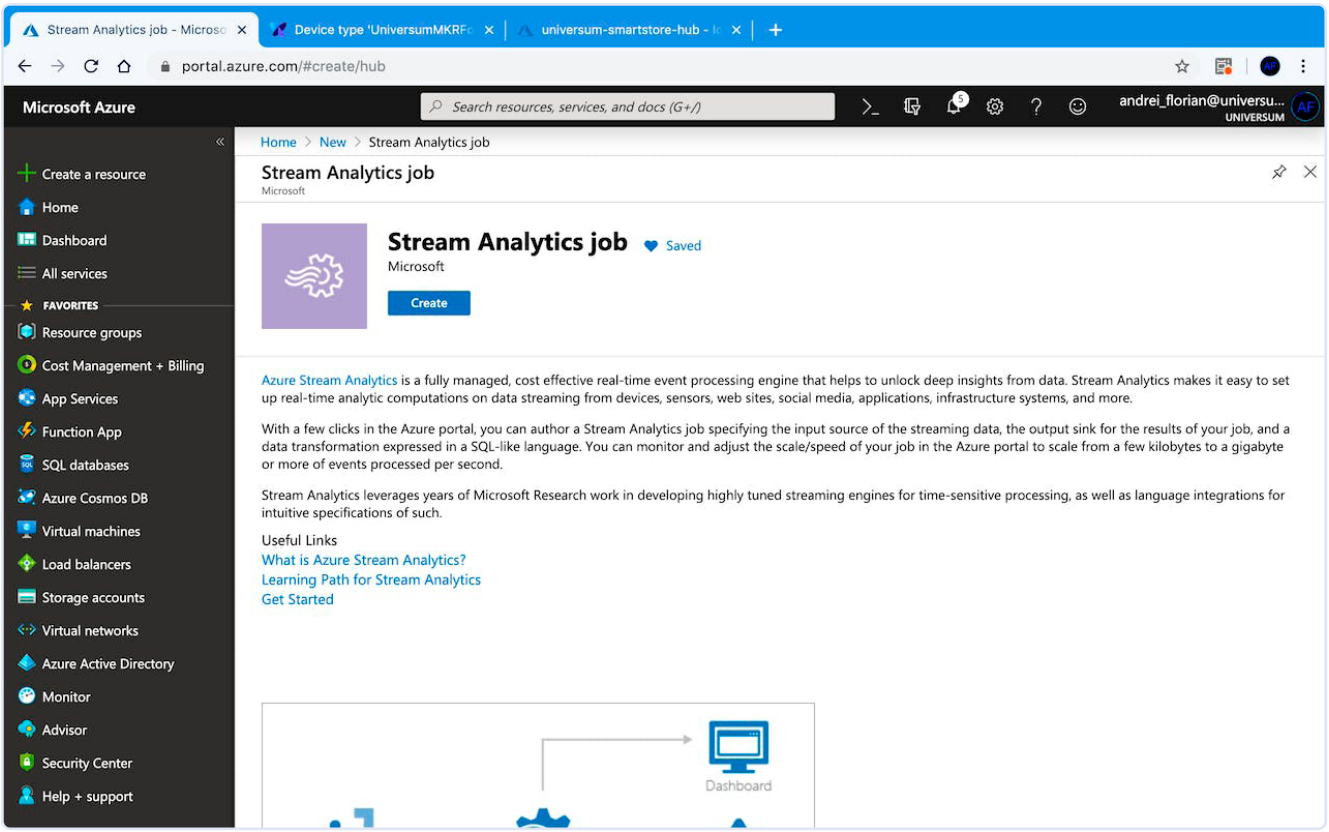


Figure 8. Configurez la tâche Stream Analytics dans Microsoft Azure.

```
    "time" : {time},
    "duplicate" : {duplicate},
    "snr" : {snr},
    "station" : "{station}",
    "avgSignal" : {avgSnr},
    "lat" : {lat},
    "lng" : {lng},
    "rssi" : {rssi},
    "seqNumber" : {seqNumber}
}
```

Ceci définit les valeurs que nous voulons transmettre à Azure.

Configuration de la gestion des flux

L'étape suivante consiste à configurer la tâche *Stream Analytics* (**figure 8**) qui interrogera les données du hub IdO Azure et les enverra dans un ensemble de données Power Bi. Pour que cela fonctionne, vous devez d'abord effectuer toutes les étapes ci-dessus, puis vous reporter à [1] pour plus de détails sur la configuration de cette partie. À l'étape 15, vous devez saisir la requête suivante :

```
SELECT
latitude as latitude,
longitude as longitude,
battery as battery,
System.Timestamp AS Timestamp
INTO
[OutputToPowerBI]
FROM
[InputFromIOTHub]
```

Télécharger le code

Nous devons maintenant télécharger le code pour tester la connexion. Assurez-vous que la tâche d'analyse de flux est en cours d'exécution, puis téléchargez le code sur l'appareil. Simulez un bruit de tronçonneuse sur votre téléphone et faites en sorte que l'appareil envoie un événement vers le cloud. Vous devriez voir les graphiques sur votre tableau de bord se déplacer après avoir envoyé les données. Cela indique que les données sont reçues, et vous pouvez passer à autre chose. Si l'événement n'apparaît sur aucun des graphiques, vous devez commencer à déboguer avec la base arrière Sigfox, puis vous diriger vers Azure.

Configuration de Microsoft Power BI

Nous allons maintenant configurer Microsoft Power BI (**figure 9**) pour nous aider à visualiser nos données. Notez qu'un compte professionnel est nécessaire pour attribuer un abonnement à Power BI dans Microsoft. Nous espérons que vous avez accès à un tel compte. Sinon, il existe de nombreuses alternatives que vous pouvez utiliser. Les conditions préalables sont les suivantes :

- Compte Power BI
- Toutes les étapes précédentes effectuées

Pour les mêmes raisons que pour la configuration du compte Azure, veuillez vous réporter à [1] pour la configuration d'un compte Power BI en quinze étapes détaillées.

Boîtier

Enfin, après avoir terminé la configuration de l'application, nous devons créer un boîtier pour le projet (**figure 10**). J'ai décidé d'attacher le dispositif à un arbre et de le camoufler pour prendre les bûcherons par surprise. Veillez à placer le microphone à l'extérieur du boîtier pour qu'il puisse capter les sons.

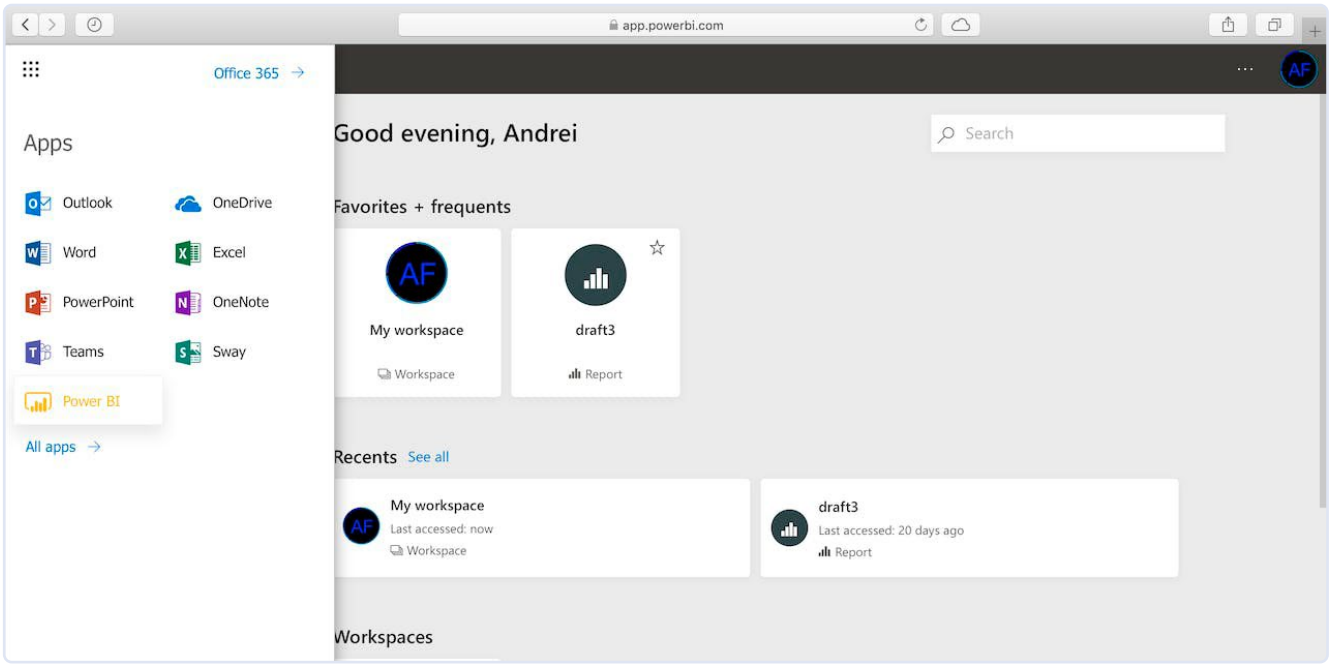


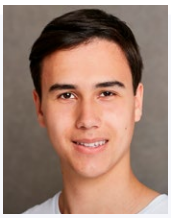
Figure 9. Accédez à Power BI de Microsoft dans l'application Office Online.

Une solution IdO

Un ensemble d'appareils Senso tels que décrits ci-dessus, cachés dans les bois, peut guetter les tronçonneuses et alerter les autorités lorsque des arbres sont abattus. J'espère que les pays seront en mesure de mettre fin à la déforestation illégale grâce aux solutions IdO. Senso est une première approche dans la recherche de solutions bon marché pour lutter contre le changement climatique pas à pas.

220423-04 — VF : Helmut Müller

Figure 10. Le boîtier que j'ai conçu pour mon prototype.



À propos de l'auteur

Basé en Irlande, Andrei Florian est un étudiant de deuxième cycle qui se passionne pour la technologie comme moyen d'inspirer des changements positifs et de gérer le développement global. Il a travaillé sur plusieurs projets portant sur divers objectifs de développement durable spécialisés dans l'Internet des objets et la cryptographie.

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (andrei@andreiflorian.com) ou contactez Elektor (redaction@elektor.fr).

Produits

Vous recherchez les principaux éléments mentionnés dans cet article ? Arduino et Elektor s'occupent de vous !

- **Arduino MKR FOX 1200 (SKU 19096)**
  www.elektor.fr/19096
- **OPEN-SMART GPS – module série GPS pour Arduino (SKU 18733)**
  www.elektor.fr/18733

LIEN

[1] Senso sur le Arduino Project Hub : https://create.arduino.cc/projecthub/andreiflorian/senso-c00153