PROGRAMMEZ!

Le magazine des développeurs

07/08/09 2020 N°1 Spécial

SPÉCIAL ÉTÉ 2020

/ **Créer des jeux 2D**
avec GDevelop

/ **Raspberry PI**
Créer un GPS
Comment utiliser Lora ?
Hacker les Sonoff

/ **Makers**
Les modules M5

/ **Hack**
Je hacke une carte bleue

/ **Lapin**
La nouvelle vie
du Nabaztag

/ **Matériel**
Diskio Pi :
une tablette française

© François Tonic

En partenariat avec
Framboise 314

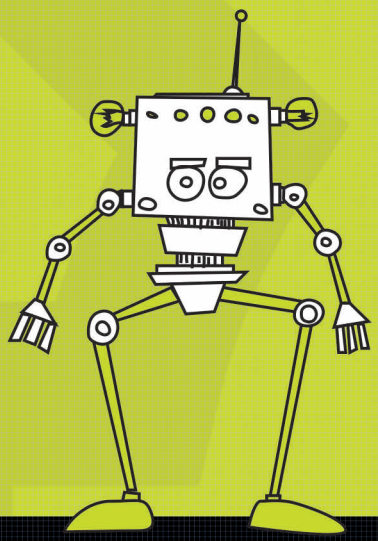
M 01642 - 1H - F: 6,50 € - RD

Le seul magazine écrit par et pour les développeurs

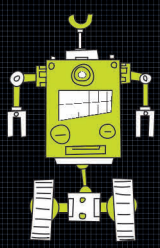
Printed in EU - Imprimé en UE - BELGIQUE 7 € - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 € - TOM 1020 XPF - MAROC 55 DH

Collection

LA FABRIQUE



L'éditeur des **MAKERS**



eni

www.editions-eni.fr

Spécial été 2020

Après les mois difficiles que nous avons vécus, il était temps que l'été arrive… Avant de penser à la rentrée...

Et nous vous proposons notre premier trimestriel. Comme nous vous l'avons précisé dans le 241, Programmez! fait sa révolution : 6 numéros standards et 4 numéros spéciaux. Ces numéros seront thématiques.

Ce numéro thématique spécial été reprend la philosophie que nous appliquons depuis longtemps au magazine : profitons de l'été pour nous amuser un peu ! Nous allons parler matériel, maker, DIY, hacking, IoT, jeux et algorithme !

La Raspberry Pi sera une des vedettes de ce numéro. Framboise 314 vous a réservé 3 excellents articles : Lora, Sonoff et GPS ! Les modules M5 n'auront plus de secrets non plus. Une bonne excuse pour découvrir cette curieuse plateforme.

Avec GDevelop, vous pourrez développer, en quelques semaines, vos propres jeux 2D. Nous parlerons aussi de la PybStick série Programmez! Philippe nous montre comment utiliser un TFT 2,4'' tout en repoussant les limites de l'animation sur cette petite carte, qui possède une puissance insoupçonnée.

Nous parlerons du petit lapin Nabaztag et de la 3e génération grâce à la nouvelle carte, la TagTagTag et à une pile logicielle open source. Le lapin reprend vie et, cette fois-ci, il est réellement autonome.

Nous aurons aussi du hacking de carte bleue et les techniques possibles. Attention : dossier à manipuler avec prudence.

Bon été et rendez-vous le 4 septembre pour le numéro 242 !

François Tonic - ftonic@programmez.com

SOMMAIRE

Prochain numéro

PROGRAMMEZ! 242
DISPONIBLE LE 4 SEPTEMBRE

JFTL 2020 / Montrouge

La journée des tests logiciels revient pour la 12e année. Le but est de montrer, expliquer les tests, et pourquoi ils sont si importants. 1 000 personnes sont attendues. La journée du 6 est réservée aux ateliers et sessions avancées. Site : http://www.cftl.fr/JFTL/accueil/

SEPTEMBRE

1	2	3	4	5	6	7
JFTL 2020 / Montrouge						
8	**9**	**10**	**11**	**12**	**13**	**14**
Meetup Programmez!/ Paris Détail sur programmez.com			JUG Summer Camp / La Rochelle			BIG DATA PARIS
15	**16**	**17**	**18**	**19**	**20**	**21**
BIG DATA PARIS						
22	**23**	**24**	**25**	**26**	**27**	**28**
Solutions RH/Paris	Cloud Expo & IoT World / Paris Solutions RH/Paris	Cloud Expo & IoT World / Paris Solutions RH/Paris				
29	**30**					

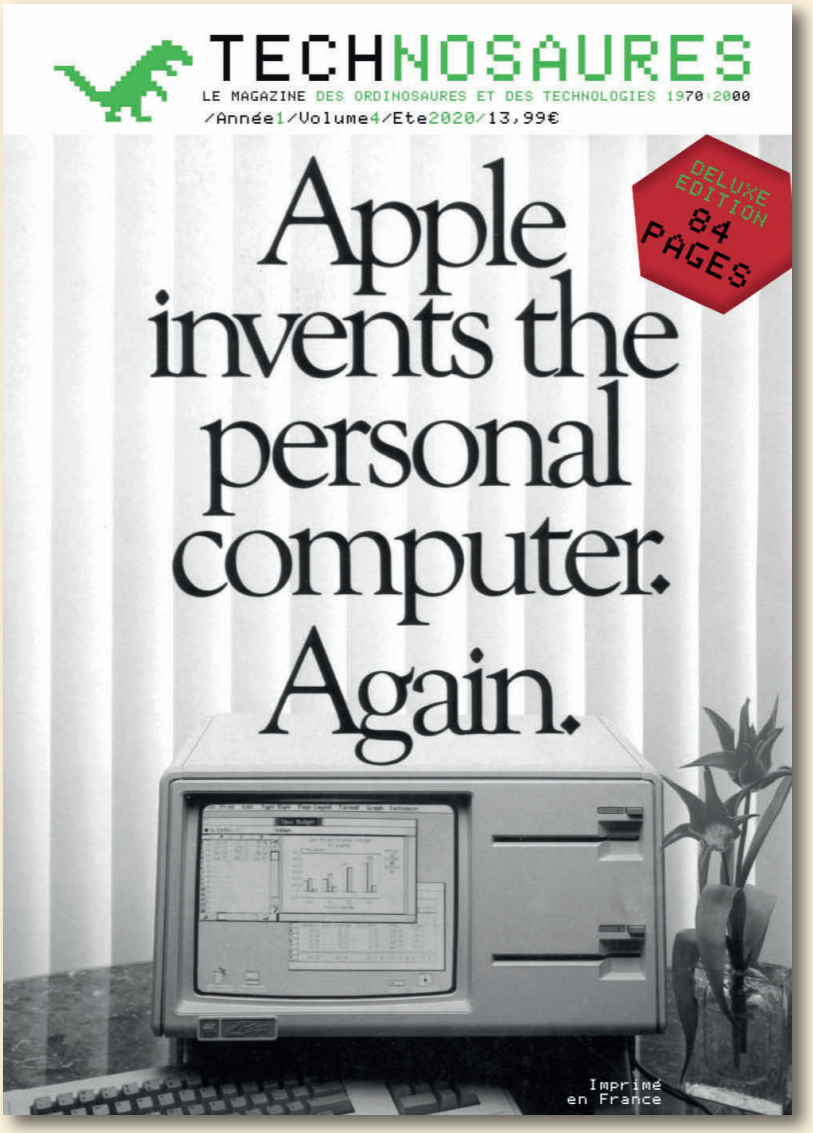
OCTOBRE

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	**16**	17	18	19	20	21
DevOps Days / Paris DevFest Nantes Volcamp / Clermond Ferrand	DevFest Nantes Volcamp / Clermond Ferrand					
22	23	24	25	26	27	28
29	**30**	31				
	BDX I/O / Bordeaux					

Merci à Aurélie Vache pour la liste 2020, consultable sur son GitHub :
https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

Ne ratez pas **Technosaures #4**

Spécial Apple Lisa



2 versions : **52** ou **84 pages** !

SOUTENEZ LE PROJET :

https://fr.ulule.com/technosaures-numero-special-apple-lisa/

Campagne participative de financement.

Jusqu'au 7 août

Disponibilité : **septembre**

Open Source : Microsoft continue son éternel mea culpa

Lors d'une conférence au MIT, le président et directeur juridique de la firme de Redmond, Brad Smith, a reconnu que les positions passées de Microsoft à l'égard de l'open source l'avaient placé « du mauvais côté de l'histoire » et que tout ceci était aujourd'hui de l'histoire ancienne. Microsoft ayant acquis GitHub et fait de son cloud Azure son cheval de bataille, qualifier l'open source de « cancer » comme à la bonne époque serait en effet plutôt malvenu.

iOS : la course au jailbreak continue

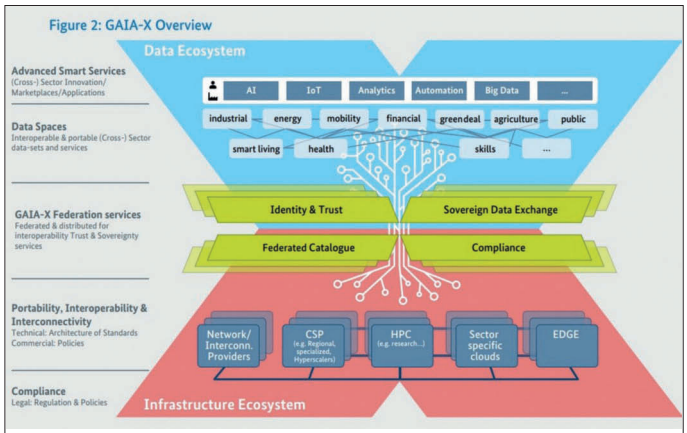
Les développeurs de l'application de jailbreak unc0ver ont eu droit à leur moment de gloire : grâce à une faille de sécurité laissée par Apple dans sa version 13.5 d'iOS, l'application de jailbreak a été en mesure de rooter les iPhone ayant installé la dernière version d'iOS. Un petit exploit qui n'était pas arrivé depuis 2014, mais Apple a rapidement sifflé la fin de la récré et a corrigé la faille en question une semaine après la sortie du jailbreak, en publiant une version 13.5.1 d'iOS.

StopCovid : l'application de contact tracing est arrivée

L'application de contact tracing StopCovid est arrivée sur les app store le 2 juin. Disponible pour iOS et Android, cette application vise à utiliser le Bluetooth pour identifier les contacts avec les autres utilisateurs et éventuellement alerter les utilisateurs en cas de contact avec une personne infectée par la Covid19. Fin de l'histoire ? Pas tout à fait : le nombre d'utilisateurs ne décolle pas vraiment et le coût de l'hébergement jugé excessif fait débat. À tel point que l'association Anticor a révélé avoir signalé la situation au parquet de Paris, dénonçant un soupçon de favoritisme. Ne rangez donc pas le popcorn tout de suite.

L'internet satellitaire de SpaceX bientôt une réalité ?

Non content d'avoir envoyé deux humains dans la station spatiale internationale au début du mois de juin, SpaceX a continué à bombarder ses minisatellites constituant sa flotte Starlink. Avec 450 satellites en orbite, la société d'Elon Musk se rapproche peu à peu de son objectif de 800 satellites et espère pouvoir proposer une version bêta de son service d'accès à internet à partir du mois d'août en Amérique du nord. La société a déjà reçu en mars l'aval de l'autorité américaine des télécoms. Et avec la récente faillite de son principal concurrent, OneWeb, ce serait dommage de laisser passer l'occasion.



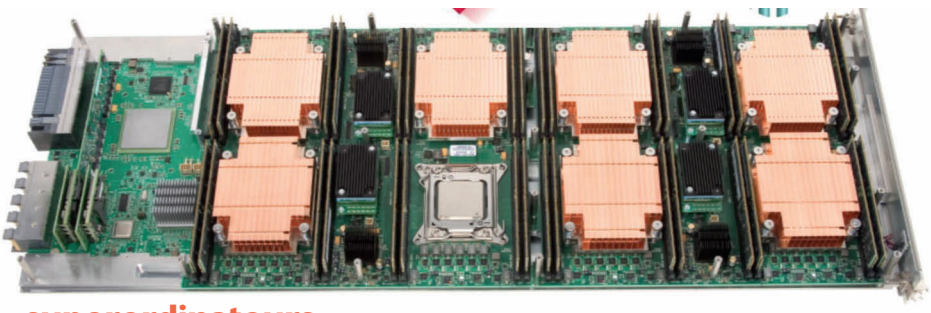
Gaia-X : le cloud souverain (mais européen cette fois) fait son retour

La France et l'Allemagne ont présenté début juin le projet Gaia-X, projet visant à proposer une infrastructure cloud européenne. Le projet prendra la forme d'une association à but non lucratif basée à Bruxelles et regroupera plusieurs entreprises, dont Atos, OvhCloud, 3 DS Outscale, Orange ou encore Scaleway. Le but affiché est de proposer une alternative aux géants américains du cloud, en développant un écosystème de fournisseurs cloud mettant l'accent sur l'interopérabilité. Reste à faire oublier les précédents échecs en matière de cloud souverain et à rattraper l'avance prise par les gros acteurs américains. Plus facile à dire qu'à faire.

Facebook s'offre Giphy

Facebook a annoncé vendredi 15 mai le rachat de Giphy, le réseau social dédié au partage d'image animée. Le montant exact de la transaction n'a pas été révélé, mais l'acquisition est estimée à environ 400 millions de dollars pour Facebook. Facebook explique que la moitié du trafic de Giphy provenait déjà de Facebook, Instagram et d'autres applications appartenant déjà au réseau social. Dans ce contexte, autant l'intégrer directement au sein des produits.

MàJ : mi-juin, le rachat est sous enquête de l'autorité de la concurrence anglaise. Cette procédure ne signifie pas le blocage de l'opération. Facebook se veut rassurant : Giphy restera autonome.



Des superordinateurs piratés à la chaîne

Pendant le mois de mai, plusieurs superordinateurs académiques ont dû mettre en pause leurs opérations après avoir détecté une intrusion malveillante au sein de leurs systèmes. La première intrusion a été détectée par les opérateurs du systèmes ARCHER, à l'université d'Edimbourg, puis plusieurs superordinateurs basés en Allemagne et en Suisse ont signalé des intrusions similaires. La méthode utilisée pour accéder aux systèmes était chaque fois la même : le vol d'identifiants SSH utilisés par les universitaires. Et l'objectif chaque fois le même également : miner des cryptomonnaies, en l'occurrence du Monero, pour le compte des attaquants.

PROGRAMMEZ!

Le magazine des développeurs

A PARTIR DU 1ER SEPTEMBRE 2020

LES CONTACTS POUR LES ABONNEMENTS ET RÉABONNEMENTS CHANGENT.

Une unique adresse :

Programmez
service abonnement
57 rue de Gisors
95300 Pontoise

Une seule adresse mail :

abonnement@programmez.com

Sans oublier notre boutique en ligne : www.programmez.com

François Tonic
(texte & photos)

Sauvez votre lapin avec la tagtagtag !

Ah le petit lapin Nabaztag ! Cet objet connecté français eut finalement une courte vie et un succès relatif malgré un intérêt certain. Mais à sa sortie, en 2005-06, son usage était limité. Et sa vie fut mouvementée : rachat en 2009 puis tombé entre les mains d'Aldebaran. Les serveurs furent coupés en 2015 laissant 150 000 lapins sans vie. Pour redonner un peu d'espoir, le projet TagTagTag fut initié par Enero. Amusons-nous un peu !

Enero est la nouvelle société du co-créateur du lapin, Olivier Mével. TagTagTag (version Ulule) fonctionne avec les 2 versions du lapin.

Attention : ce kit n'est pas compatible avec le Karotz.

Le matériel

- 1 Nabaztag ou Nabaztag:tag
- 1 carte tagtagtag
- 1 Pi Zero version sans fil
- 1 carte SD 16 Go
- 1 des tournevis
- 1 PC ou Mac pour flasher la SD et configurer

En option :

- Souffleur d'air chaud
- Pince
- Colle ou pâte à fixer

On change la carte !

Le lapin étant un objet connecté, il n'est pas autonome et nécessite un serveur distant pour pouvoir fonctionner. TagTagTag remplace purement et simplement la carte électronique originelle. Surtout, pour pouvoir lui offrir une connectivité plus ouverte et une capacité de traitement local. La carte fonctionne avec l'ajout d'une Raspberry Pi Zero WH pour la connectivité et l'OS. Il s'agit donc d'un hat (un peu plus grosse que la normale).

Elle possède : 5 NeoPixels, 2 microphones, des capacités audio, un accéléromètre et un capteur de température + humidité.
Mais TagTagTag prend son sens avec la partie logicielle : Pynab. PyNab est l'association du Nabaztag et de Python. Cette pile vient avec une distribution Raspbian préconfigurée. Ainsi, il propose des services de base comme la météo, l'heure, la reconnaissance vocale, les humeurs, etc. La configuration de base se fait par une interface web, simple, mais efficace.

En théorie, la configuration se fera d'elle-même au premier démarrage du lapin, qui nécessite plusieurs minutes. On pourra ensuite passer l'interface web. Il faut tout d'abord configurer l'audio, les oreilles et le lecteur RFID. Cela reste un peu rudimentaire.

Une configuration / connexion parfois laborieuse

Il faut avouer que la partie configuration n'est pas le point fort de la tagtagtag, même chose pour la connexion à l'interface web et à la CLI.

Voilà, vous avez téléchargé la dernière version de la distribution et flashé la carte avec un outil de type Etcher. Mais avant de mettre la SD dans le lapin, deux choses à faire :

- Créer un fichier wpa_supplicant.conf
- Créer un fichier ssh

Le premier sert à configurer (en clair) le réseau wifi :

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1


network={
  ssid="ssid"
  psk="mot de passe"
  id_str="wifi access"
}
```

Mettez le nom du réseau et son mot de passe. Le second fichier permet tout simplement d'autoriser une connexion ssh alors que l'accès est par défaut bloqué sur la Raspbian.

Attention : supprimez bien l'extension txt à chaque fois. Le fichier ssh n'a aucune extension.

Avant de mettre la coque et terminer entièrement le montage, nous vous conseillons de tester le bon fonctionnement : branchez le lapin, après quelques secondes, les LED et les oreilles s'activent. Vous pouvez maintenant vous connecter en ssh et à l'interface web.

Si la connexion normale est nabaztag.local, il arrive régulièrement qu'il faille passer par l'IP. Pour trouver l'IP du lapin, utilisez un IP scan. Dans le navigateur, tapez l'IP et l'interface web doit se charger. Parfois, il faut plusieurs minutes pour que la distribution et la pile Nabaztag soient entièrement montées. Sur un Terminal, la commande ssh (avec l'IP) suffit. Le mot de passe par défaut est raspberry.

.Net, programmation par bloc, Python

Pour le support .Net / Mono : voir article dédié.

Pour utiliser simplement les fonctionnalités du lapin, on peut utiliser nabblocky, un environnement de programmez par bloc. C'est simple et ludique. Dommage que ce service soit si mal mis en avant ! On l'utilise directement depuis l'interface web.

Le nouveau lapin est codé en Python et son nouveau core code est Pynab…

Les +

- Redonner vie au Nabaztag
- Code sur GitHub
- Ludique
- Montage
- Usage de la Pi Zero

Les –

- Fonctionnalités standard limitées
- Configurations parfois fastidieuses
- Montage des câbles
- Documentation

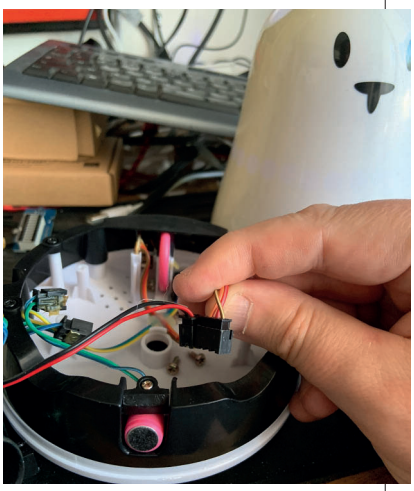
ON DÉMONTE !

Le démontage du lapin ne pose pas de réel problème.

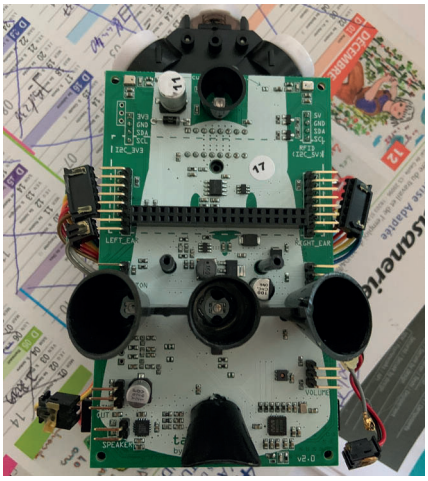
1. Retirez les oreilles puis enlever les vis du dessus pour retirer la coque. L'intérieur apparaît.
2. Dévissez l'ensemble électronique du socle et retirez avec précaution l'ensemble
3. Déconnectez l'ensemble des câbles. Attention ne tirez pas dessus utilisez une pince ou un tournevis plat. Si la colle résiste, chauffez-la. Mais avec le temps, elle peut sécher et se casser facilement.
4. Déconnectez et retirez la carte RFID
5. Dévissez la carte centrale et retirez avec délicatesse les éléments plastiques des différentes LED
6. Installez la tagtagtag et reconnectez les câbles. Attention : suivez attentivement le tutoriel officiel, car tous les câbles ne se montent pas dans le même sens. N'oubliez pas de remettre les éléments en plastique.
7. Insérez le Pi Zero (avec la carte SD) puis remettez en place la carte RFID. Vissez-la.
8. Réinstallez le bloc électronique sur le socle. Le micro du lapin n'est pas utilisé. Revissez.
9. Remettez en place la coque. Si quelque chose gêne, ne forcez pas ! Revissez les vis situées en dessous. Remettez les oreilles.
10. Connectez le fil d'alimentation.
11. Si tout est OK, le Nabaztag s'allume et les oreilles s'agitent !

Le lecteur RFID est sensible, il faut positionner la carte suffisamment près de la coque, sans forcer sur les vis. Calez avec du carton.

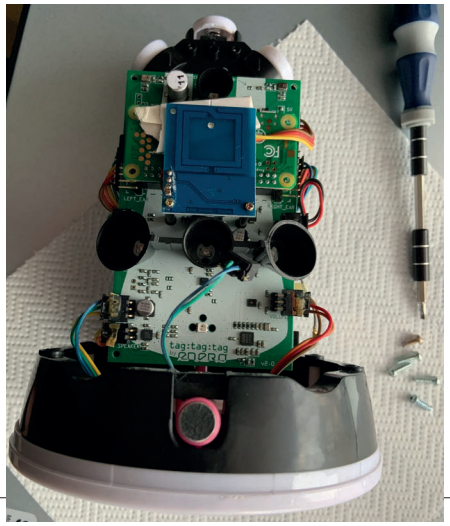
On retire le capot et on découvre l'électronique interne



On retire l'ensemble des câbles et le bloc central



La nouvelle carte en cours d'installation



Le montage est presque terminée. On cale bien le lecteur RFID

Le lapin revit !

Laurent Ellerbach
Principal Engineer Manager
Microsoft
laurelle@microsoft.com
https://github.com/Ellerbach

Faire tourner son Nabaztag en .NET

Avec la nouvelle carte disponible pour mettre à jour nos lapins préférés, il devient possible grâce au Raspberry Pi Zero qu'il embarque de faire tourner du code .NET dessus. Alors, c'est parti pour une exploration des possibilités !
En premier lieu, il faut activer l'accès en ssh de façon à pouvoir s'y connecter. Un simple fichier nommé « ssh » dans la racine de la partition de boot est suffisant !

.NET Core : bah non… mais mono oui !

La famille .NET est assez large et il n'est pas forcément facile de s'y retrouver. Il y a d'un côté .NET Core (version 3.1 actuellement) qui est multi-plateforme (Windows, macOS, Linux) et multi-architecture (x32, x64, arm). .NET Core est totalement open source, maintenu par Microsoft et l'ensemble des sources sur GitHub. .NET Core s'installe sur n'importe quelle plateforme et il est possible de compiler sur n'importe quelle plateforme également. Une application se déploie facilement, il suffit de récupérer un simple répertoire qui contient toutes les dépendances. L'outil de développement favori est Visual Studio ou VS Code. VS Code fonctionne sur toutes les plateformes.

De l'autre, on trouve .NET Framework (actuellement version 4.8) qui est la version la plus traditionnelle et tourne sous Windows avec des architectures x32 et x64. On trouve aussi mono qui est une implémentation open source et permet de faire tourner du code .NET Framework sur diverses plateformes et architectures. .NET Framework se compile donc sous Windows.

Au milieu, vous avec .NET Standard. C'est la meilleure façon de pouvoir partager du code entre les 2 familles précédentes de .NET. La version actuelle est 2.1 mais seule la 2.0 est compatible avec .NET Framework 4.8.

Et pour finir, les 2 frameworks .NET Core et .NET Framework vont être réunis avec la version .NET 5.0 actuellement en preview et prévue pour la seconde moitié de l'année calendaire. .NET 5.0 sera comme .NET Core multi-plateforme et multi-architecture.

Quel rapport avec mon lapin me demanderez-vous : mon premier réflexe a donc été de faire une application en .NET Core et de le déployer sur le lapin. Et là, bam, une erreur : « Segmentation fault ».

Mais que se passe-t-il ? Le temps de reprendre mes esprits, je me souviens d'une chose : l'architecture des Raspberry Pi Zero n'est pas la même que celle des Raspberry Pi 2, 3 et suivantes. Le processeur est de la famille des arm v6 alors que les suivants sont arm v7. Et donc après quelques minutes de recherche sur bing, je trouve qu'effectivement le support de .NET Core ne commence que pour les architectures arm v7. A noter qu'un PR est en cours pour le support de arm v6.

Et c'est là que mono rentre dans le jeu. Il supporte ce type d'architectures mais ne supporte pas directement les binaires générés par .NET Core. Il faut donc changer de type de projet et passer en .NET Framework 4.8. Je développe sous Windows avec Visual Studio. J'ai le SDK .NET Framework 4.8 installé. Je peux donc compiler sous Windows et copier le répertoire aves les fichiers générés sur mon Raspberry Pi Zero. Côté performances, évidemment, c'est loin de celles de .NET Core mais au moins cela fonctionne.

Pour installer mono sur le pi avec ssh :

```
sudo apt-get install mono-runtime
```

Cela va installer le runtime qui permettra de faire tourner des applications en mode console. Si vous voulez également faire des applications de type ASP.NET, il faudra installer « mono-complete » à la place de « mono-runtime ».

Implémentation du protocole pynab

La librairie qui vient avec le nouveau Nabaztag est en python, *pynab*, elle propose un protocole de communication entre le cœur principal, le service *nabd*, et les services auxiliaires. Je me suis dit que j'allais créer un composant qui me permette de faire du Text to Speech (TTS), donc me permettre de lire quelque chose sur le lapin à partir d'un texte. L'idée étant de pouvoir l'utiliser pour des notifications. Me voilà donc parti à écrire en .NET un connecteur qui permette de faire cela.

J'ai commencé les développements avant d'avoir la carte ! J'ai donc également écrit un simulateur et en fin de compte j'ai fini par faire une classe qui permette de supporter l'ensemble du protocole. Une fois la carte arrivée et le tout installé, j'ai pu raffiner le code. Et commencer à écrire le TTS.

Ce qui a été compliqué c'est le retro engineering sur les chorégraphies, j'ai donc écrit une classe qui permet de dumper de façon lisible les fichiers et aussi d'en construire facilement. On trouve cet utilitaire dans le répertoire Utils du GitHub. Et l'autre chose qui est assez compliquée aussi c'est que le protocole se base sur des sockets qui sont connectées en permanences. Ce qui n'est pas si simple à gérer, notamment pour les reconnections et détection d'interruption.

Une application exemple est disponible :
https://github.com/Ellerbach/Nabaztag.Net/tree/master/Nabaztag.Net.Sample
Elle permet de montrer les différentes possibilités offertes par la classe Nabaztag.Net notamment s'abonner à des événements comme l'appui du bouton, la reconnaissance vocale ou le mouvement des oreilles.

```
var _nabaztag = new Nabaztag("localhost", 10543);
_nabaztag.StateEvent += Nabaztag_StateEvent;
_nabaztag.ButtonEvent += Nabaztag_ButtonEvent;
_nabaztag.EarsEvent += Nabaztag_EarsEvent;
_nabaztag.AsrEvent += Nabaztag_AsrEvent
var resp = _nabaztag.EventMode(ModeType.Idle, new EventType[] { EventType.Button,
EventType.Ears, EventType.Asr }, false, 30);
if (resp.Status == Status.Ok)
  Console.WriteLine("Your Nabaztag is in Idle mode and will receive all events");
else
  Console.WriteLine($"Something wrong happened: {resp.ErrorClass}, {resp.ErrorMessage}");
Console.ReadKey();
```

Les callbacks de réception des évènements ne font qu'une mise en forme pour les afficher, ici la fonction pour la position des oreilles :

```
private static void Nabaztag_EarsEvent(object sender, EarsEvent ears)
{
  Console.WriteLine($"New ear event. Left: {ears.Left} Right: {ears.Right} Ear: {ears.Ear},
Time: {ears.Time}");
}
```

L'initialisation est donc très simple et l'ensemble des fonctions disponibles également. Et vous recevez donc les événements :

```
Your Nabaztag is in Idle mode and will receive all events
{"type":"response","status":"ok","state":0,"uptime":388702,"connections":5}
{"right":249,"time":"2020-06-08T09:37:36.285392+02:00"}
{"right":10,"time":"2020-06-08T09:37:38.442023+02:00"}
{"type":"button_event","event":0,"time":"2020-06-08T09:37:40.955232+02:00"}
New button event: Down, Time: 2020-06-08T09:37:40.955232+02:00
{"type":"button_event","event":1,"time":"2020-06-08T09:37:41.680561+02:00"}
New button event: Up, Time: 2020-06-08T09:37:41.680561+02:00
{"type":"button_event","event":2,"time":"2020-06-08T09:37:41.860852+02:00"}
New button event: Click, Time: 2020-06-08T09:37:41.860852+02:00
```

Autre exemple, faire jouer au lapin une chorégraphie et une musique :

```
Sequence[] seq = new Sequence[] { new Sequence(), new Sequence() };
seq[0].ChoreographyList = CreateChoreography().SerializeChoreography();
seq[1].AudioList = new string[] { "nabsurprised/respirations/Respiration01.mp3" };
var resp = _nabaztag.Command(seq, true, 30);
if (resp.Status == Status.Ok)
  Console.WriteLine("Your Nabaztag is doing a respiration and a small choreography");
else
  Console.WriteLine($"Something wrong happened: {resp.ErrorClass}, {resp.ErrorMessage}");
```

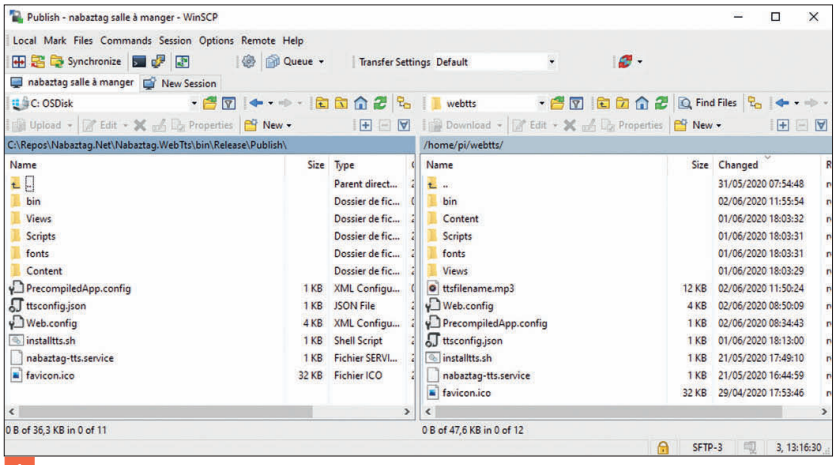
Nabaztag.Net contient également la possibilité de créer des chorégraphies en les décrivant. Un utilitaire dans le projet permet de lire les existantes et d'en écrire de nouvelles. Voici un exemple de création de chorégraphie à la volée :

```
private static Choreography CreateChoreography()
{
  Choreography choreography = new Choreography();
  // delay: 0, OpCode: FrameDuration frameDuration: 16
  choreography.SetFrameDuration(16);
  // delay: 0, OpCode: SetLedPalette Led: Bottom, palette: 0
  choreography.SetLedPalette(Led.Bottom, 0);
  // delay: 0, OpCode: SetLedPalette Led: Left, palette: 1
  choreography.SetLedPalette(Led.Left, 1);
  // delay: 0, OpCode: SetLedPalette Led: Right, palette: 1
  choreography.SetLedPalette(Led.Right, 1);
  // delay: 0, OpCode: SetLedPalette Led: Nose, palette: 2
  choreography.SetLedPalette(Led.Nose, 2);
  // delay: 1, OpCode: SetLedoff Led: Left
  // Etc, Etc, Etc
  // delay: 1, OpCode: SetLedoff Led: Nose
  choreography.SetLedOff(Led.Nose, 1);

  return choreography;
}
```

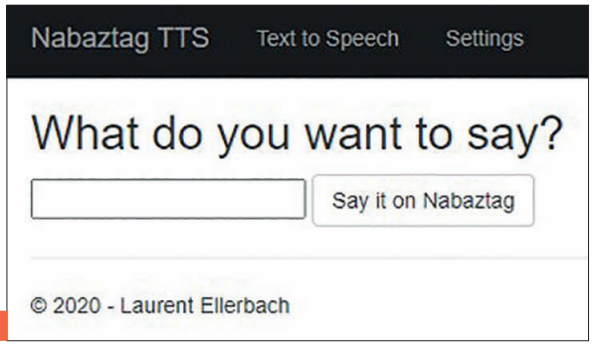
Vous pouvez également produire vos propres sons et musiques. Cela ouvre donc les portes à une interaction et l'écriture de nouveaux services. Dans mon cas, c'est surtout la connexion avec Azure qui m'intéressait.

L'exemple de TTS sur le Nabaztag

Cet exemple se trouve sur https://github.com/Ellerbach/Nabaztag.Net/tree/master/Nabaztag.Tts.
C'est donc une application en .NET 4.8 qui va utiliser les services cognitifs de Azure pour la synthèse vocale. Le texte va y être envoyé et un fichier audio mp3 mono va être récupéré. L'application va ensuite demander au service nabd de le jouer.
Le code en tant que tel est assez simple une fois tout le protocole implémenté dans la classe Nabaztag.Net. Vous aurez besoin d'un compte Azure gratuit et d'activer les services cognitifs. Le niveau gratuit F0 sera largement suffisant pour une utilisation ponctuelle du service.
Plus d'information sur les services disponibles ici : https://docs.microsoft.com/fr-fr/azure/cognitive-services/speech-service/index-text-to-speech. Une fois le compte Azure créé, ajoutez un « Text to Speech » et sélectionnez F0 pour le niveau, la zone France ou n'importe quelle autre zone.
Il faudra ensuite mettre dans le fichier de configuration la clé qui sera générée. Une documentation plus complète est disponible sur le même GitHub dans la section WebTts.
Vous pourrez ensuite compiler la solution et la copier sur le Nabaztag, en utilisant par exemple WinSCP (https://winscp.net/), ou SCP en ligne de commande. Il faut juste la mettre dans /home/pi/tts par exemple. Il est important que le répertoire dans lequel vous le mettez soit au même niveau que le répertoire pynab. WinSCP permet de copier facilement des répertoires et des fichiers depuis Windows sur des machines distantes de type Linux qui supportent des protocoles



1



2

de type FTP, SFTP, SCP et d'autres. C'est le cas du Raspberry Pi une fois SSH installé. La copie s'effectue de façon sécurisée. Il suffit de se connecter avec le login et le mot de passe.
Voici un exemple avec l'application Nabaztag.WebTts : 1
Il ne vous reste plus ensuite qu'à aller dans le répertoire et lancer :

```
mono Nabaztag.Tts.exe
```

Vous obtiendrez en résultat :

```
pi@nabaztag:~/tts $ mono Nabaztag.Tts.exe
Hello Text to Speech!
Type the text for Nabaztag to say: C'est super cool d'avoir un exemple de text to speech
 qui fonctionne top!
TTS played properly
```

Si vous souhaitez ajouter cette fonctionnalité de Text to Speech en plus, j'ai créé un projet Nabaztag.WebTts qui permet d'offrir cela. Le tout est packagé avec un fichier d'installation script pour installer les composants de mono et tout paramétrer correctement. Attention, lors du premier lancement et du premier accès, le site prend un peu de temps à se lancer. 2

La page settings permet de rentrer les clés et de sélectionner la voix et la langue. Une REST API est également disponible. Soit en GET soit en POST : GET /api/Speak
Par exemple : http://nabaztagipaddress:8888/api/Speak/dis%20quelque%20chose
Cela permet déjà d'apporter ce service qui est quand même très utile. Côté performance, mis à part le premier lancement et la première utilisation dans lesquels il y a un petit délai, c'est ensuite très rapide. A peine vous envoyez le texte que le lapin le joue déjà ! Cela m'avait d'ailleurs impressionné, je pensais qu'il y aurait plus de latence. La réalité est que le fichier mp3 généré est vraiment petit et se charge donc rapidement. Au final, l'animation avec les oreilles et le son d'introduction sont ce qui prend le plus de temps.

Pour aller plus loin: remplacer nabd par Nabaztag.Server

Si vous voulez aller plus loin, vous pouvez essayer de replacer nabd par Nabaztag.Server que je suis en train de finaliser. Quand j'ai installé la première fois le lapin, d'une part, j'ai été très frustré qu'il mette vraiment très longtemps à booter à cause des services de reconnaissance vocaux installés localement et d'autre part que la reconnaissance vocale et la transformation en intention ne fonctionne juste pas. Même pour des choses très simples, le service est quasiment inutilisable.
Je voulais également pouvoir étendre facilement les fonctionnalités du lapin en ajoutant d'autres composants qui puissent facilement se greffer dessus. Mais j'avais vraiment besoin d'une bonne reconnaissance vocale, d'intention et synthèse vocale qui fonctionne vraiment bien.
L'idée est donc de garder le reste des composants qui fonctionnent et sont écrits en Python, y compris le site web de configuration, et de remplacer le service principal nabd. Cela permet d'avoir directement intégré nativement un service de Speech to Text, donc de reconnaissance vocale native qui s'appuie sur les service cognitifs d'Azure, l'identification (intents en anglais) avec LUIS (Language Understanding Intelligent Service) qui est aussi un composant d'Azure, et bien sûr le support du Voice to Text intégré directement dans cette partie serveur.
Techniquement, je m'appuie sur l'utilisation de .NET Core IoT pour le support du bouton. J'ai récupéré et étendu le support de media pour supporter la lecture et l'enregistrement à longueur non préalablement définie de cette même librairie. Pour ce dernier composant, comme il n'est pas écrit en .NET Standard 2.0, il m'a fallu faire quelques ajustements.
Pour le mouvement des oreilles, je m'appuie sur le driver natif en pinvoke. Le driver est simple, il suffit de lire et écrire dans un simple fichier /dev/ear0 et 1 suivant l'oreille.
Et enfin, pour les lumières, ce sont des neo led. Leur pilotage est fait par PWM avec DMA et je m'appuie sur une librairie existante. L'implémentation de .NET Core IoT ne supporte pas ce mode, seul SPI est implémenté.
Côté performances, on peut noter que la mémoire utilisée par l'application avec mono est d'environ 150M vs plus de 500M avec Python pour plus de fonctionnalités en .NET ! Par contre, l'utilisation du processeur est plus élevée, à cause d'un plus grand nombre de threads qui tournent notamment pour la gestion des évènements du bouton et des oreilles.
Au moment où j'écris ces lignes, le code est sur la branch NabaztagServer. L'idée étant de proposer prochainement une version avec installation.

Jérôme Bezet-Torres
Professeur d'informatique Académie de Lyon BTS SIO, formateur et consultant en informatique, mes domaines d'expertises : Active Directory, DNS, GPO, MDT, Automatisation, PowerShell, Virtualisation, WPF. Je suis Microsoft Certified Tranier depuis 10 ans
Blog : https://JM2K69.github.io

TrueNas : API et Powershell le couple gagnant

Qui est à l'origine de TrueNas ?

iXsystems, Inc. est une société américaine de technologie informatique privée basée à San Jose, en Californie, qui développe, vend et prend en charge des produits et services informatiques et de stockage. Ses principaux produits sont des distributions FreeBSD (1) open sources personnalisées. Les serveurs de fichiers et les systèmes de stockage en réseau **FreeNAS** et **TrueNAS**. Elle commercialise également des plateformes matérielles pour ces produits et développe des architectures de stockage à l'échelle de l'entreprise et des infrastructures convergentes.

La solution de Stockage Open source

Récemment iXsystems a annoncé un grand changement concernant les deux produits **FreeNas** qui étaient une version gratuite, et **TrueNas** qui était la version payante destinée aux entreprises. Ils ont décidé de les unifier et de faire une seule version de leur serveur de stockage TrueNas Core. Quels sont les avantages de ce rassemblement des équipes de développement :

- Un développement rapide,
- Amélioration de la qualité des versions,
- Documentation unifiée,
- Flexibilité,
- Un projet open source,
- Intégration de OpenZFS 2.0.

De cette union des équipes, il existera toujours les deux versions. La version Entreprise sera déverrouillée lors de l'achat d'une licence.

TrueNAS CORE — Community supported, Open source, Rapid development, Early availability — TrueNAS ENTERPRISE

TrueNAS fournit le partage de fichiers de type de blocs (iSCSI) à tous les principaux systèmes d'exploitation clients et plateformes de virtualisation. Les protocoles basés sur les fichiers pris en charge incluent Windows SMB, Apple AFP, Time Machine et Unix NFS, ainsi que FTP et WebDAV.

Le partage de blocs TrueNAS iSCSI prend en charge **VMware VAAI**, **Microsoft ODX** et Microsoft Windows Server Clustering. Le partage d'objets compatible S3 permet à TrueNAS de fournir du stockage sous forme d'objet et de communiquer avec tous les principaux fournisseurs de sauvegarde cloud.

VAAI (vStorage APIs for Array Integration), est une API qui a pour objectif principal de déporter certaines tâches de manipulation de VMs (clones, gestion des clones, mouvement des vmdks) de l'hyperviseur vers la baie de stockage. Ces tâches ou encore fonctions sont appelées "primitives". VAAI permet donc de décharger les opérations liées au stockage, manipulations de fichiers vers la baie et permet ainsi de soulager l'hyperviseur en CPU et bande passante. VAAI était essentiellement présente sur des architectures en mode bloc (iSCSI,FC, FCoE).

La technologie Microsoft ODX est une API qui permet elle aussi d'améliorer les transferts de fichiers sur des liens iSCSI ou des liens de type FC.

TrueNas version entreprise

La version entreprise s'accompagne de matériels certifiés et validés par leurs services pour garantir le bon fonctionnement **TrueNas**. iXsystems commercialise des serveurs dédiés au stockage. Plusieurs formats sont disponibles. Voici un exemple de solutions disponibles sur : https://www.ixsystems.com/truenas/

TrueNas – Core

La version TrueNas Core (**C**ommunauty supported, **O**pen source, **R**apid development, **E**arly availability) restera la version gratuite de la distribution FreeNas, elle ne dispose pas des mêmes fonctionnalités que la version entreprise payante. Par exemple, la haute disponibilité n'est pas présente. Actuellement la version de TrueNas est encore en cours de développement et nous sommes sur un cycle de Beta, cependant au niveau de son API V2 (Application Programming Interface), elle reste pour le moment inchangée et elle est complète au niveau des fonctionnalités depuis la version 11.3 de FreeNas. Vous pouvez télécharger la version de votre choix :

- FreeNas : https://www.freenas.org/download-freenas-release/
- TrueNas : http://download.freenas.org/12.0/MASTER/latest/x64/

L'illustration 1 montre l'ensemble des fonctionnalités offertes par TrueNas dans sa future version 12.

Installer TrueNas dans une VM

Pour installer votre TrueNas, il faut préalablement avoir téléchargé l'Image ISO. Dans notre cas, nous allons utiliser comme logiciel de virtualisation VMware Workstation. Vous pouvez toutefois utiliser VirtualBox si vous le souhaitez. Pour ce qui est des prérequis, vous devez disposer d'un processeur permettant la virtualisation de système d'exploitation 64 bits avec les instructions Intel-VTX ou AMD-V. Un minimum de mémoire vive 8 Go étant le minimum. Il est cependant possible d'avoir une machine avec 4 Go de RAM. Pour l'espace de stockage, un disque dur de 20 Go d'espace est suffisant. Dans notre exemple notre machine aura les caractéristiques suivantes :

- Mémoire : 8Go
- CPU : 8 processeurs
- Stockage :
  - Système d'exploitation : 1 disque 20 Go
  - Partage : 3 disques de 90 Go

(1) FreeBSD est un système d'exploitation gratuit et open-source de type Unix issu de Berkeley Software Distribution (BSD), qui était basé sur Research Unix.

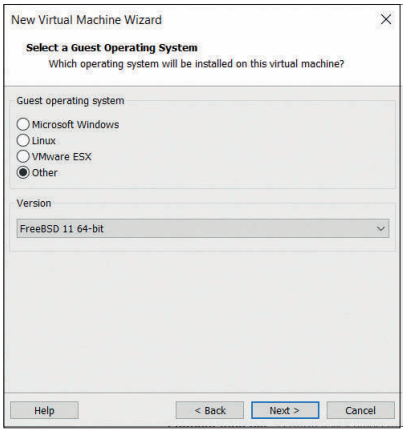
TrueNAS 12.0				
	CORE & Enterprise Shared Features			Enterprise Only Features
Mgmt.	Multi-Systems	TrueCommand, RBAC, Audit	SingleSignOn, Dataset Mgmt	Alerting, reporting, Analytics	Enclosure Views in TrueCommand
	Administration	Web UI, SNMP, Syslog	REST API, WebSockets API	NetData (plugin), Reports	vCenter Plugin
	Systems Utilities	Tasks, Cron Jobs, Scripts	In-Service Updates	Alerting, Email, Support	Proactive Support Monitoring
	Clients and Applications	*Windows, MacOS, Linux, UNIX, iOS, Android Clients*	*Many applications via SMB, NFS, or iSCSI*	*Integrated applications via ZFS and Jails/VMs*	**Certifications:** VMware, Veeam, Citrix
Services	Application Services	Jails, Plugins, VMs	Plex, Asigra, Iconik, NextCloud, other Plugins	*Linux/Docker/Kubernetes* (VM), FreeBSD (jails or VMs)	High Availability (HA) Plugins, VMs
	Directory Services	Active Directory, 2Factor	Local users and groups	NIS, LDAP, Kerberos	
	Storage Services	**File:** NFS v3/4, SMB1/2/3, AFP, FTP, WebDAV, rsync	**Block:** iSCSI, VAAI, OpenStack Cinder	**Object:** S3 Host, Scale-out, Cloud sync, Credentials	ALUA, Fibre Channel
ZFS	Data Management	Unlimited Snapshots, Pool checkpoints	Space-efficient Clones	Replication: Remote, Local, Auto-resume, to Linux ZFS	
	Data Protection	Accelerated Copy-on-Write, 2-Copy Metadata	RAID-Z1/Z2/Z3, Mirrors, Fast Resilvering, Fast Boot	Self-healing checksums, Background Scrubbing	
	Data Reduction	Thin/Thick Provisioning	In-line Adaptive Compression	Clones, Deduplication, Trim	
	Data Acceleration	All Flash, Fusion Pools, Metadata on Flash	Read Cache (arc/l2arc): RAM/Flash	Write Chache (slog/zil): Flash	NVDIMM, dual port SAS/NVMe
OS	Networking	IPv4, v6: 1 - 100GbE, DHCP	LAGG, VLANs	Jumbo frames, TCP options	Fibre Channel (8-32Gb)
	Data Security	Self-Encrypted Drives (TCG Opal), Dataset Encryption	Encrypted replication, WireGuard, OpenVPN	ACLs, IP Filtering	FIPS 140-2 SEDs, KMIP
	Foundation	FreeBSD, Boot mgmt, SSH	locale jails, Byhve VMs	System logging, NTP	Performance Autotune
HW	High Availability	Fast ZFS Replication	*Client-based Mirroring*	*Application-level Replication*	Dual Controller HA
	Hardware Support	IPMI Remote Mgmt	SAS JBODs, Global spares	SMART, SSD Wear Monitoring	Visual Enclosure Management
	Platforms	Any x86 system (CORE only) Improved AMD support	MIni E/X/XL+	iXsystems Servers	X-Series, M-Series
Support		Community Support - Forums, Documentation, Release Notes, Bug Ticketing			**Enterprise Support:** up to 24x7

Notes: *italics indicates the feature requires third party software separate from TrueNAS*
Blue Text are major features added with TrueNAS 12.0 (many additional, minor enhancements not listed)

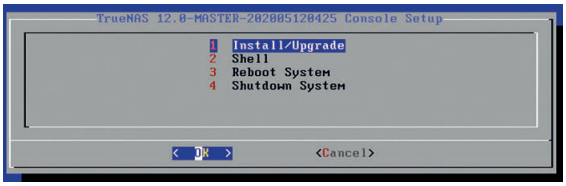
1

• Réseau : un réseau en bridge.

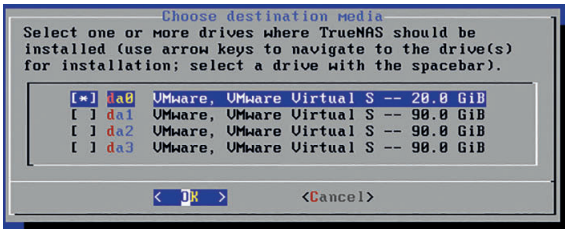
Dans VMware Workstation il faut choisir comme système d'exploitation **FreeBSD 11 64-Bit** comme le montre l'image ci-dessous.



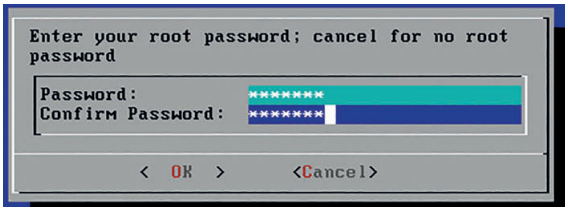
Il faut maintenant démarrer votre machine virtuelle et vous laissez guider par le menu d'installation.



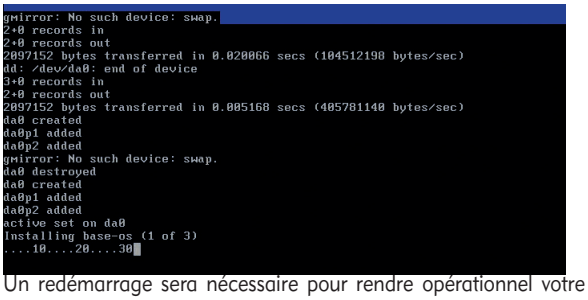
Le choix du disque ou des disques sur le(s)quel(s) vous souhaitez faire l'installation.



Aux cours de l'installation il vous est demandé de renseigner le mot de passe de l'administrateur. Ici le mot de passe du compte root. Attention le clavier au moment de l'installation est en QWERTY .

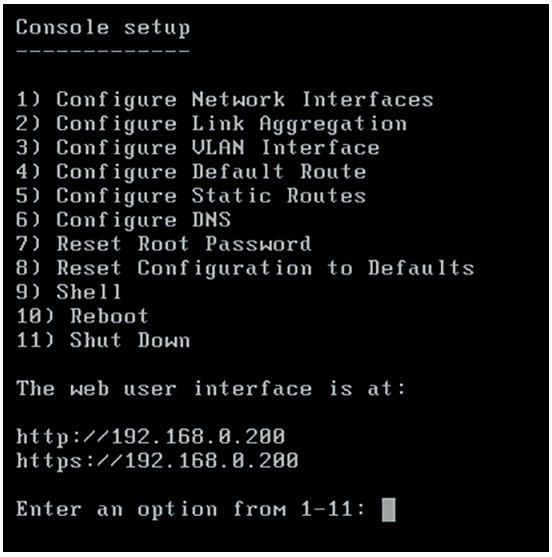


L'installation se poursuit :



Un redémarrage sera nécessaire pour rendre opérationnel votre TrueNas Core.

Après le redémarrage votre machine récupère une adresse par DHCP, sinon à l'aide du menu **1°)** vous pouvez lui configurer une adresse IPv4 ou IPv6.



Votre serveur est maintenant disponible à l'adresse 192.168.0.200 en HTTP et/ou HTTPS. 2

TrueNas – API

API et Powershell

Pour commencer il est important de poser la définition d'une API : une API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications. API est un acronyme anglais qui signifie « Application Programming Interface », que l'on traduit par interface de programmation d'application.

Il existe trois approches d'accès aux API :

- API privée : L'API n'est utilisable qu'en interne. Cette approche permet de garder un contrôle total sur l'API.
- API partenaires : L'API est partagée avec certains partenaires de l'entreprise. Cette approche peut générer de nouveaux flux de revenus sans compromettre la sécurité.
- API publique : L'API est accessible à tous. Cette approche autorise les tiers à développer des applications qui interagissent avec votre API et peut devenir source d'innovations.

Pour standardiser l'échange des informations entre les API toujours plus nombreuses, il a fallu développer un protocole : le « Simple Object Access Protocol », ou SOAP. Les API conçues d'après le protocole SOAP utilisent le format XML pour leurs messages et reçoivent des requêtes via HTTP ou SMTP. SOAP a pour objectif de simplifier l'échange des informations entre les applications qui s'exécutent dans des environnements différents ou qui ont été écrites dans des langages différents.

Le « Representational State Transfer », ou REST, est une autre tentative de normalisation. Les API Web qui respectent les contraintes de l'architecture REST sont appelées API RESTful. Ces deux éléments diffèrent sur un point fondamental : SOAP est un protocole, alors que REST est un style d'architecture. Cela signifie qu'il n'existe aucune norme officielle qui régisse les API Web RESTful. Les échanges de données peuvent être obtenu sous différents format le JSON, XML.

Avec Powershell, nous avons la possibilité d'interagir avec les API à l'aide d'une cmdlets Invoke-RestMethod nous utilisons des méthodes données par les verbes, POST, GET, DELETE. Voici un exemple de l'utilisation d'une API pour récupérer la météo d'une ville par exemple Lyon :

```
$city= "Lyon"
#récupérer l'ID de la ville
$info = Invoke-RestMethod -URI ("https://www.metaweather.com/api/location/search/?query=" + $city)
#récupérer la météo
Invoke-RestMethod -URI ("https://www.metaweather.com/api/location/" + $info.woeid)
```

Le résultat est le suivant :

```
consolidated_weather : {@{id=5567955184123904; weather_state_name=Showers; weather_state_abbr=s; wind_direction_compass=N;
                       created=2020-05-14T10:31:51.888944Z; applicable_date=2020-05-14; min_temp=9,03; max_temp=14,915;
                       the_temp=14,805; wind_speed=6,57825281583552; wind_direction=3,5000000000000036; air_pressure=1009,0;
                       humidity=83; visibility=12,041552334367296; predictability=73}, @{id=5826958891941888;
                       weather_state_name=Showers; weather_state_abbr=s; wind_direction_compass=NW;
                       created=2020-05-14T10:31:54.882628Z; applicable_date=2020-05-15; min_temp=8,805; max_temp=15,975;
                       the_temp=13,975; wind_speed=5,940822949722951; wind_direction=330,0; air_pressure=1014,5; humidity=71;
                       visibility=11,966987791298815; predictability=73}, @{id=5061306128269312; weather_state_name=Heavy Cloud;
                       weather_state_abbr=hc; wind_direction_compass=N; created=2020-05-14T10:31:59.027382Z;
                       applicable_date=2020-05-16; min_temp=9,16; max_temp=19,43; the_temp=17,835; wind_speed=5,9118699130014045;
                       wind_direction=1,9999999999999918; air_pressure=1019,0; humidity=62; visibility=13,340218126143323;
                       predictability=71}, @{id=5477167555355648; weather_state_name=Heavy Cloud; weather_state_abbr=hc;
                       wind_direction_compass=N; created=2020-05-14T10:32:01.023786Z; applicable_date=2020-05-17;
                       min_temp=9,760000000000002; max_temp=21,205; the_temp=18,965000000000003; wind_speed=6,8504662162353849;
                       wind_direction=0,4999999999999903; air_pressure=1021,5; humidity=56; visibility=14,134330908226636;
                       predictability=71}...}
time                 : 2020-05-14T15:11:33.699857+02:00
sun_rise             : 2020-05-14T06:10:07.525431+02:00
sun_set              : 2020-05-14T21:04:40.392032+02:00
timezone_name        : LMT
parent               : @{title=France; location_type=Country; woeid=23424819; latt_long=46.71,1.72}
sources              : {@{title=BBC; slug=bbc; url=http://www.bbc.co.uk/weather/; crawl_rate=360}, @{title=Forecast.io;
                       slug=forecast-io; url=http://forecast.io/; crawl_rate=480}, @{title=HAMweather; slug=hamweather;
                       url=http://www.hamweather.com/; crawl_rate=360}, @{title=Met Office; slug=met-office;
                       url=http://www.metoffice.gov.uk/; crawl_rate=180}...}
title                : Lyon
location_type        : City
woeid                : 609125
latt_long            : 45.759392,4.828980
timezone             : Europe/Paris
```

Le système TrueNas propose trois types d'API :

- Une API en version 1 dont la documentation est disponible à l'adresse suivante : https://api.freenas.org/ ;
- Une API RestFull et une API en Websocket dont les documentations sont disponible à l'adresse suivante https://api.ixsystems.com/freenas/

Le premier module Powershell dédié à interagir avec les systèmes FreeNas disponible sur la Powershell Gallery utilise la version 1 de l'API qui est en version 2.0.2.

L'API RestFull en version 2 est devenue pleinement fonctionnelle avec la version 11.2 de FreeNas, aujourd'hui nous sommes sur la version 11.3-Update2 de **FreeNas**.

À la suite du changement annoncé par **iXsystems** sur le rapprochement des branches de développement et sur la réalisation d'un seul OS : TrueNas. J'ai décidé de me focaliser sur l'API RestFull et donc de créer un nouveau module **TrueNas**

Les API de TrueNas nécessitent une authentification basique. Il faut envoyer dans notre en tête de notre requête le couple login et mot de passe, le mot de passe est converti en Bas64.

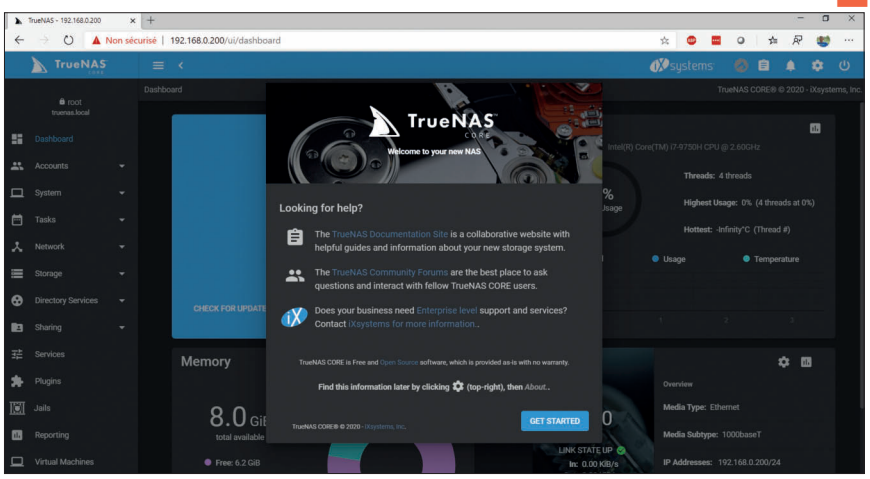
Etude de deux fonctions du Module TrueNas

Connection à un serveur

Voici le code de la commande Connect-TrueNasServer :

```
function Connect-TrueNasServer
{
  [CmdletBinding()]
  [Alias()]
  [OutputType([String])]
  Param
  (
    # Description d'aide Truenas
    [Parameter(Mandatory = $true)]
    [Alias("Truenas")]
    $Server,
    [Parameter(Mandatory = $false)]
    [String]$Username,
    [Parameter(Mandatory = $false)]
    [SecureString]$Password,
    [Parameter(Mandatory = $false)]
    [PSCredential]$Credentials,
    [Parameter(Mandatory = $false)]
    [switch]$httpOnly = $false,
    [Parameter(Mandatory = $false)]
    [switch]$SkipCertificateCheck = $false,
    [Parameter(Mandatory = $false)]
    [ValidateRange(1, 65535)]
    [int]$port
  )
```

2



```
Begin
{
}
Process
{
    $Script:SrvFreenas = $Server


    #If there is a password (and a user), create a credentials
    if ($Password)
    {
        $Credentials = New-Object -TypeName System.Management.Automation.PSCredential($User
name, $securecurepassword)
    }
    #Not Credentials (and no password)
    if ($NULL -eq $Credentials)
    {
     $Credentials = Get-Credential -Message 'Please enter administrative credentials for your TrueNas'
    }
    $cred = $Credentials.username + ":" + $Credentials.GetNetworkCredential().Password
    $base64 = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($cred))
    #headers, We need to have Content-type set to application/json...
    $script:headers = @{ Authorization = "Basic " + $base64; "Content-type" = "application/json" }
    $script:invokeParams = @{ UseBasicParsing = $true; SkipCertificateCheck = $SkipCertificateCheck }

    if ("Desktop" -eq $PSVersionTable.PsEdition)
    {
        #Remove -SkipCertificateCheck from Invoke Parameter (not supported <= PS 5)
        $invokeParams.remove("SkipCertificateCheck")
    }

    if ($httpOnly)
    {
        if (!$port)
        {
            $port = 80
        }

        $uri = "http://${Server}:${port}/api/v2.0/system/info"

    }
    else
    {
        if (!$port)
        {
            $port = 443
        }
        #for PowerShell (<=) 5 (Desktop), Enable TLS 1.1, 1.2 and Disable SSL chain trust
        if ("Desktop" -eq $PSVersionTable.PsEdition)
        {
            Write-Verbose -Message "Desktop Version try to Enable TLS 1.1 and 1.2"
            #Enable TLS 1.1 and 1.2
            Set-TrueNasCipherSSL
            if ($SkipCertificateCheck)
            {
                Write-Verbose -Message "Disable SSL chain trust"

                #Disable SSL chain trust...
                Set-TrueNasuntrustedSSL
            }

        }
        $uri = "http://${Server}:${port}/api/v2.0/system/info"

    }

    $script:port = $port
    $script:httpOnly = $httpOnly
    $Script:ApiVersion = $ApiVersion

    try
    {
        $result = Invoke-RestMethod -Uri $uri -Method Get -SessionVariable Truenas_S -headers
$headers @invokeParams
    }
    catch
    {
        Show-TrueNasException -Exception $_
        throw "Unable to connect"
    }

    if ($null -eq $result.version )
    {
        throw "Unable to get data"
    }

    Write-Host "Welcome on"$result.name"-"$result.version"-"$result.system_product""

    $Script:Session = $Truenas_S


}
End
{

}
}
```

Le Bloc de **param** nous permet de définir les variables comme le Login, le mot de passe, le type de communication HTTP ou HTTPS et l'adresse IP du Serveur.

Le bloc suivant permet de convertir le mot de passe en Base64 et de construire l'entête de ma requête en spécifiant le Content-type ici une **application/json.**

```
    $base64 = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($cred))
    #headers, We need to have Content-type set to application/json...
    $script:headers = @{ Authorization = "Basic " + $base64; "Content-type" = "application/json" }
    $script:invokeParams = @{ UseBasicParsing = $true; SkipCertificateCheck = $SkipCertificateCheck }
```

Enfin, dans la commande Try nous utilisons la commande Invoke-RestMethod pour tenter de se connecter au Serveur TrueNas. En cas d'erreur nous passons dans le Catch pour capturer l'erreur et la commande Show-TrueNasException nous permet d'afficher l'erreur. Voici un exemple de l'utilisation de la commande. 3
A ce stade, seul le compte root peut accéder à l'API.

Lister les disques

Cette commande nous permet de lister l'intégralité des disques durs disponibles dans notre serveur TrueNas. Par défaut l'API nous renvoie du JSON et Powershell le transforme en Objet.

```
function Get-TrueNasDisk
{
  [CmdletBinding()]
  Param( )
  begin
  {
    $uri = "api/v2.0/disk"

  }
  process
  {
    $results = Invoke-TrueNasRestMethod -Uri $Uri -Method Get

    foreach ($disk in $results)
    {
      $Name = ($disk.name)
      $Size_GB = ([Math]::Round($disk.size / 1024 / 1024 / 1024, 2))
      Write-Verbose -Message " Find the disk $name with the size $Size_GB"
      [PSCustomObject]@{
        Name   = ($disk.name)
        Number = ($disk.number)
        Size_GB = ([Math]::Round($disk.size / 1024 / 1024 / 1024, 2))
        Type   = ($disk.type)
        Model  = ($disk.model)

      }
    }
  }
  end
  {}
}
```

Cette commande envoie une URL partielle "api/v2.0/disk" qui est reconstituée dans la fonction PowerShell **Invoke-TrueNasRestMethod**. La variable **results** récupère donc l'objet **JSON** renvoyé par l'API et ensuite un nouvel Objet PowerShell est créé pour chaque objet contenu dans la variable **results**. Cette commande une fois exécutée sur notre TrueNas nous renvoie la sortie suivante. 4

Module PowerShell TrueNas

Pourquoi ce module

Ce module est disponible dans un repository Github à l'adresse suivante https://github.com/powerTrueNas. Mais il n'est pas encore disponible sur la Powershell Gallery. Cependant, il existe le module **FreeNas** qui est en version 2.0.2. Il s'agit du module qui utilise l'API en version 1 qui fonctionne toujours.
Ce module me permet d'automatiser la configuration des serveurs TrueNas et de configurer le stockage ainsi que le service iSCSI. Tout cela pour mettre à disposition à mes étudiants des environnements VMware vSphere composés de 3 ESXI avec un vCenter et un TrueNas ou FreeNas comme stockage mutualisé, pour mettre en œuvre les fonctionnalités de HA, vMotion, DRS et Fault Tolerance.

Démonstration

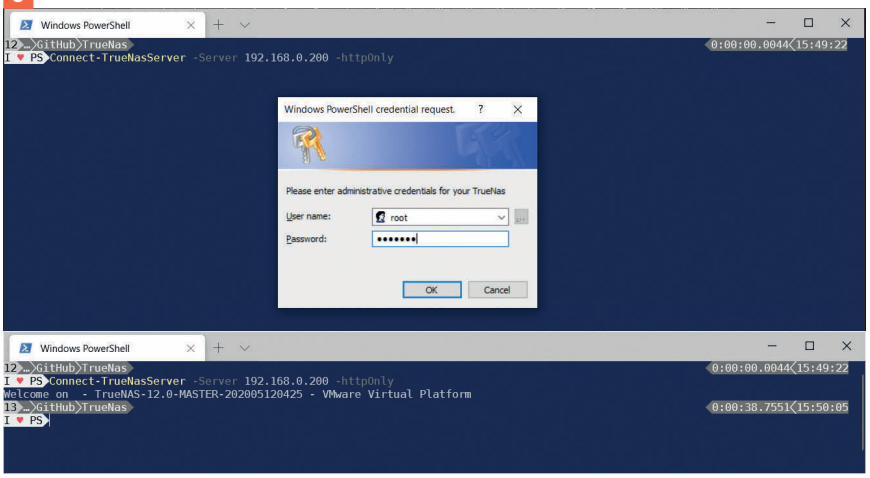
Plusieurs démonstrations de l'utilisation du module ont été faites, je fais partie du groupe French Powershell User Group (FRPSUG) animé par plusieurs personnes dont 3 MVP, François-Xavier Cat, Stéphane Van Gulick et Olivier Miossec.

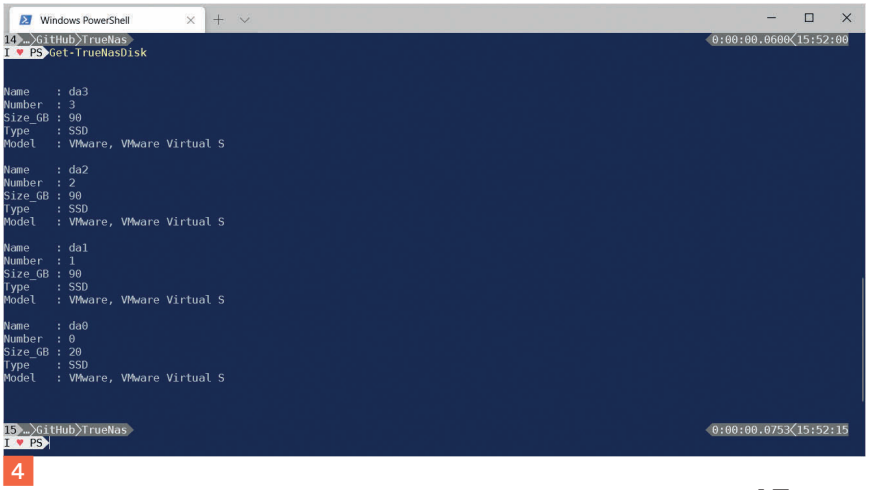
La première présentation du module a été faite au cours d'une présentation de 10 min pour le **FRPSUG. La chaîne Youtube est disponible à cette adresse :** https://www.youtube.com/channel/UCyxicOKZNm_u1opF_xAYfDA.
Chaque année le FRPSUG organise un évènement à Paris sur une journée, lors de l'évènement 2019, le French PowerShell Saturday, j'ai animé deux sessions une coanimées avec Damien Van Robaeys [MVP] sur le thème WPF et PowerShell et une sur mon module **FreeNas**.

Première présentation du module FreeNas au cours d'une présentation éclair de 10 min.

Cette vidéo montre comment utiliser le module FreeNas, couplé avec un script PowerCli vous êtes en capacité à automatiser le déploiement et la configuration d'un environnement de virtualisation.

3





4

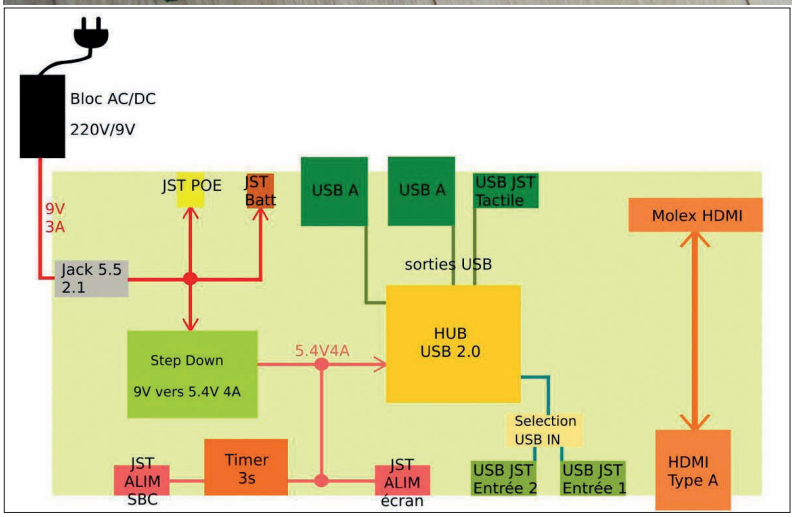
Guillaume Debray
Je suis passionné d'informatique depuis mes premiers ordinateurs : Amstrad CPC 464 et Amiga 500 en tête, suivi d'un PC Pentium 133. Autodidacte en informatique et en programmation, je suis tombé sur le Raspberry Pi en 2013 après avoir découvert Linux. Je travaille sur ce projet depuis 4 ans.

Diskio Pi : un terminal tactile open source

Voici un projet qui semble vouloir être à l'écoute des utilisateurs libres et de leurs attentes. Communément appelé « tablette tactile », l'appareil que propose la société 3DCRAN (créée pour l'occasion) est un peu plus que cela. Le terminal est livré en kit, compatible avec de nombreuses cartes SBC (small board computer), hackable, réparable, et modifiable à souhait.

Initié en 2016, le projet Diskio Pi a dû évoluer avec les avancées de l'écosystème Raspberry Pi, dont il est dépendant. C'est dans cette optique de compatibilité maximale qu'est né le projet, que ça soit avec les cartes nano PC, mais aussi avec du matériel de récupération, avec une volonté de réduire au maximum l'effet d'obsolescence programmée.

Cet article résume le long chemin qu'a traversé le projet : de l'idée initiale au produit fini proposé aujourd'hui.



Imaginer un design agréable et peu coûteux

L'idée m'est venue en installant Android sur une carte Orange Pi. Ayant déjà bricolé des écrans de récupération, je me suis tout de suite mis en tête d'assembler un écran d'un vieux pc portable 17" avec ce nano PC. Restait à ajouter un verre tactile USB et une coque, des batteries, et le tour était joué.

La conception d'un appareil multi-usages n'est pas sans difficultés pour un autodidacte. Comme pour un logiciel, il faut tout envisager, étudier les meilleures options, en gardant en tête le coût de fabrication. Et il faut de la rigueur pour concevoir l'ensemble des aspects matériels.

Pour le second prototype, j'ai choisi un écran de 13,3 full HD, avec de larges angles de vision (IPS), qui est resté pour la version finale. Mon idée première a été de pouvoir remplacer facilement la carte SBC. J'ai donc conçu une sorte de rack sous forme d'un tiroir afin de placer la carte mère à l'intérieur de la machine.

Le concept était intéressant mais plusieurs problèmes se sont avérés bloquants pour la suite : la rigidité de l'ensemble, et le coût de fabrication. Le connecteur fond de panier de 40 broches est un composant assez cher, et toute la rigidité de l'appareil tenait sur le verre tactile et l'écran, ce qui aurait été trop fragile. En 2018, il a donc fallu tout repenser sur de nouvelles bases. Pour le design, la finesse de l'appareil a été reconsidérée pour d'éventuels encastrements au mur, ainsi que le coût de production. Exit donc le rack, le bouton home intégré, et l'ouverture se fera par l'arrière, sans devoir à démonter l'écran à chaque ouverture. Pour la moitié du coût du connecteur fond de panier, un pied amovible a été modélisé, avec la possibilité de le placer à l'endroit ou à l'envers, selon l'utilisation de la machine.

Le prototype est dessiné en 3D avec un cahier des charges précis et des fonctionnalités établies dès le départ. Pendant l'avancement du modèle, les éléments non sur-mesure sont placés (verre tactile, écran,

cartes propriétaires), puis les cartes sur-mesure sont dessinées en fonction de la place restante. Pour pallier le problème d'évacuation de chaleur, un ventilateur en option est géré par la carte fille, avec une sonde de température externe, directement collée au radiateur.

La fabrication

Durant tout ce cheminement de designs différents, le calcul du prix d'achat est déterminant pour la suite. Il faut donc décortiquer le modèle et évaluer les coûts élément par élément, câble par câble, composant par composant, pour s'approcher au maximum du prix de vente optimum, trouvé préalablement grâce au sondage des personnes intéressées. Plus l'écran est grand, plus il coûte cher (à résolution et technologie identique), et plus il consomme. Inversement, plus l'écran est petit, plus la place est réduite pour insérer les cartes et les batteries. Il faut donc trouver un équilibre taille/prix/consommation électrique.

Les devis effectués principalement en Chine, il est pratiquement impossible de faire fabriquer sans fonds préalables.

Dans le cas du Diskio Pi, aucun fonds (publics ou privés) n'a été levé. Seuls les investissements des préventes ont été utilisés pour la fabrication. Ce mode de fonctionnement, à flux tendu, ne permet pas de fabriquer des moules pour la coque en plastique injecté. Elle est donc fabriquée en impression 3D, avec quelques machines grand format. Cela permet non seulement de corriger des oublis, d'améliorer certaines parties, mais aussi de proposer des versions de différentes couleurs, voire différentes matières. Et je souhaite que ces améliorations soient toujours compatibles avec les versions précédentes, bien que si besoin, il faille remplacer une partie entière de temps en temps.

Pour l'électronique, le 100 % open source n'est pas encore atteint. La gestion de l'écran (carte HDMI vers eDP) utilise une carte propriétaire : la technologie haute fréquence ne permet pas qu'elle soit étudiée sans un travail d'ingénieur, et les plans ne sont pas fournis par le fabricant, ni même les sources du firmware. Pour tout ce qui est alimentation et chargement des batteries, les cartes ont été conçues en France par un électronicien, et les plans sont disponibles sur le Git du projet. Dans l'ensemble, le terminal Diskio Pi arrive donc à être 80 % open source. J'espère que dans quelques années, la dépendance des cartes propriétaires aura évolué, avec l'intégration d'une carte écran 100 % étudiée par 3DCRAN. Ceci dit, le circuit du contrôleur USB du verre tactile, qui est fourni par le fabricant du périphérique, restera propriétaire.

Les difficultés techniques rencontrées

Durant tout le processus de prototypage, nous avons rencontré des problèmes qui ont été résolus un par un. Certains ont été relativement simples à résoudre, d'autres moins. Voici une liste, non exhaustive, des soucis techniques rencontrés :

• Le boot avec l'écran :

Ce bug a été rencontré dès le prototype 2, lors du démarrage du système. Au boot, le Raspberry Pi demande beaucoup d'énergie. L'écran aussi. Il y a donc un pic de consommation au démarrage, et nous avons longtemps cherché pourquoi le Raspberry Pi (3 à l'époque) ne bootait qu'une fois sur deux. Le problème identifié, la solution a consisté à démarrer d'abord l'écran, puis la carte avec un décalage de quelques secondes. Un temporisateur 555 est donc présent sur la carte fille.

• Les clics fantômes :

Ce genre de soucis est assez frustrant. Tout fonctionne bien, le tactile est reconnu par le kernel, le multitouch fonctionne, parfois un peu trop… En effet, parfois le curseur se déplace tout seul, parfois même il devient incontrôlable alors qu'aucun doigt ne touche la surface de l'écran.

Nous avons longtemps cherché d'où provenait le problème. Nous avons d'abord pensé à des perturbations électromagnétiques, le câble utilisé n'étant pas blindé. Donc, essais avec ferrites, de différentes tailles, puis filtre USB… Les clics fantômes étaient encore présents.

Puis une idée toute simple nous est apparu… La distance physique entre l'écran et le verre tactile. Si sur un téléphone portable cette distance peut être de 0,2mm, sur un écran plus grand 0,6 voire 0,9mm est indispensable. En remplaçant le double-face avec un modèle plus épais, le problème fut résolu.

• La compatibilité Raspberry Pi 4 :

En juin 2019 l'arrivée du Raspberry Pi 4 fut une énorme surprise pour nous tous. Nous l'attendions secrètement mais aucune rumeur ne le prédisait avant 2020. A ce moment-là, la carte fille était déjà bien avancée, et fonctionnait très bien avec le Raspberry Pi 3 et l'Odroid C2.

Mais les essais avec le Raspberry Pi 4 avec cette carte furent une déception. Comme 85 exemplaires du Diskio Pi étaient déjà en précommande en mars, la compatibilité avec une carte encore inconnue était difficile à prévoir. J'ai donc souhaité tout reprendre en juillet pour offrir aux premiers clients cette compatibilité Raspberry Pi 4.

La puissance délivrée par la carte a dû être mise à jour, et au lieu des 3A prévus, faire en sorte que la carte puisse délivrer du 5,2V à 4A. C'est maintenant chose faite !

Les derniers tests montrent une autonomie de 8 heures avec le Raspberry Pi 4, avec l'apparition de l'éclair « low power » du firmware Raspberry lorsqu'il est temps de recharger la batterie.

Pour compléter le test batterie par LED (analogique), un logiciel - codé en C++ - montrant la capacité de la batterie sera développé cet été. Il faudra simplement connecter un câble entre la carte chargeur et le Raspberry Pi (GPIO I²C) puis installer le paquet via un dépôt mis en place pour l'occasion. Il sera disponible sous Debian dans un premier temps, puis pour d'autres distributions, voire Android si besoin.

La commercialisation

Depuis l'avènement des sites de financements participatifs, il est possible de proposer un produit en prévente, sur simple présentation d'un prototype fonctionnel. C'est grâce à cette méthode que le Diskio Pi a pu voir le jour. Mais pas sans difficultés, là aussi !

Le premier financement sur Kickstarter a été un échec total. Le montant global, qui était l'équivalent de 500 unités pour proposer une coque en plastique moulé était trop élevé. Le design du prototype 2 était aussi peut-être en cause. Nous nous sommes donc recentrés en changeant de design, mais aussi en réduisant nos ambitions immédiates. Le premier financement sur ulule.fr a été nettement mieux accueilli, et nous a permis de financer le 3e prototype en 2018.

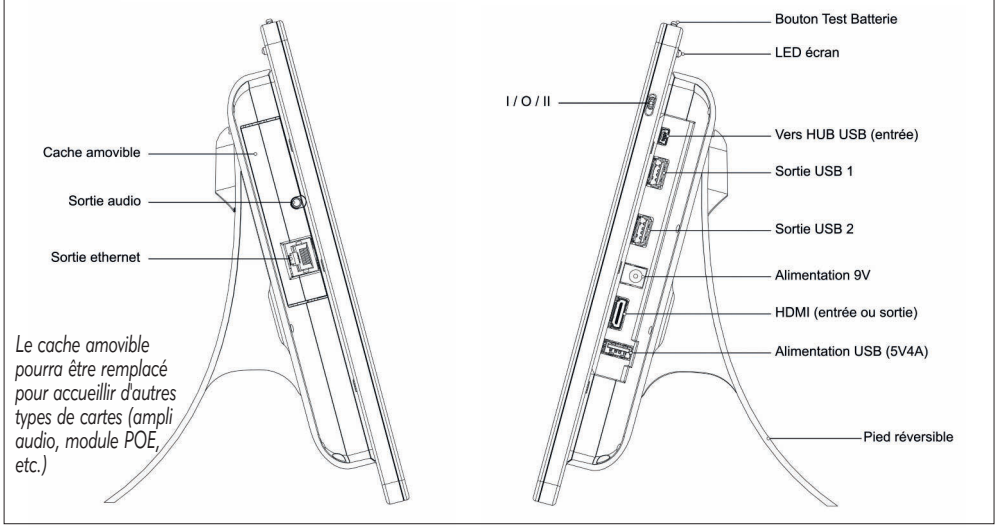
Puis, en 2019, le prototype prêt, nous avons lancé un second financement sur ulule, qui nous a lancé avec 85 préventes. En février 2020, un nouveau financement réussi sur diskiopi.com a confirmé que les gens étaient toujours présents pour nous soutenir.

Dorénavant, tous les kits et les pièces détachées seront mis en vente sur le site de Diskio Pi, sans passer par de sites tiers (notamment Amazon).

Le stock étant très difficile à constituer sans trésorerie, d'autres préventes seront organisées en 2020 et 2021, avec un minimum de commandes dans un court laps de temps. Puis, petit à petit, nous réduirons les délais de fabrication et ferons évoluer le concept avec la sortie de nouvelles options, et de nouvelles versions.

Pour mener à bien un projet comme celui-ci, la détermination et l'optimisme est la clé ! J'espère pouvoir faire évoluer l'idée et pouvoir proposer un matériel informatique libre au plus grand nombre. Merci à toutes les personnes qui ont participé aux différents financements participatifs, les premières livraisons devraient arriver cet été 2020.

LES ENTRÉES / SORTIES DE L'APPAREIL



SPÉCIFICATIONS DE L'APPAREIL

Écran

- Marque : Innolux
- Taille : 13.3 pouces
- Résolution : 1920*1080 px
- Traitement de surface : Brillant
- Type : TFT[IPS]
- Luminosité : 300 cd/m2 (Typ.)
- Angle de visualisation : 85/85/85/85/85/85
- Contraste Ratio : 800:1
- Interface de signal : eDP 30 broches
- Lampe Type : WLED

Interface tactile (optionnel)

- Structure : G/G (verre/verre)
- Technologie : capacitive, 10 points
- Interface : USB 2.0
- Dureté : ≥6H

Carte fille

- Genesys Logic USB 2.0 Hub x3 (externe 2x + interne 1x)
- Convertisseur Buck 9V en 5V@4A (Circuit de puissance)
- Régulation de température (avec radiateur/sonde en option)
- Connexions avec les SBCs par câbles : (USB, Ethernet, alimentation, HDMI).
- Sortie USB 2.0 pour utiliser le verre tactile avec un SBC ou un ordinateur externe
- Entrée externe HDMI

Carte Ethernet

- Carte d'extension simple, connecteur Ethernet non blindé sans LED
- Connecteur jack audio stéréo 3,5 mm (à venir)

Carte contrôleur écran

- Convertisseur HDMI 1.4 vers eDP
- DAC stéréo via HDMI, amplificateur stéréo classe D (1W)
- Haut-parleurs : 2 x 1W
- Fonction "écran portable" : Nappe interne HDMI.

Adaptateur secteur

- AC/DC (9V-3A) avec prise EU, UK ou US

Entrées/sorties externe

- 2x sortie USB 2.0 (500 mA)
- 1x prise 9V
- 1x HDMI in
- 1x USB 2.0 in
- 1x sortie d'alimentation USB 2.0 (4A)

Batteries (optionnel)

- Circuit de charge / décharge / équilibrage des batteries (2S3P), avec :
- 6 cellules LiPo 7.4V, 12000mAh
- Autonomie Raspberry Pi 4 : environ 7 heures IDLE

Taille de l'appareil et poids

- Largeur : 335 mm
- Hauteur : 240 mm
- Épaisseur max : 35 mm
- Épaisseur min : 12 mm
- Poids avec batterie 6 cellules et tactile : 1650 g

Protection

- IP 20 : non étanche, pas de protection aux chocs ni à la poussière.

Couleur et matériau du boîtier

- PETG mat, noir, gris ou blanc. (Impression 3D)

Prix

- Entre 198,50€ et 303€ selon les options

Plus d'informations :

www.diskiopi.com

NOUVEAU

PYBStick Standard 26 série Programmez! : la carte multi-usage

Elle est enfin là ! La série Programmez! de la PYBStick standard 26 est disponible dès le 3 juillet ! Elle utilise un microcontrôleur STM32F411 à 100 MHz, 512 Ko de stockage Flash sur la carte et de 128 Ko de RAM. Elle est compatible MicroPython et Arduino ! Dans Programmez! 241, nous avons parlé des caractéristiques de la carte. *La Rédaction.*

La carte est suffisamment puissante pour faire ce que l'on veut : température, servomoteurs, neopixel, mini-robot, etc. La PYBStick est idéale pour découvrir le monde maker, du DIY et de la programmation. Cette carte a été réalisée par Garatronic et MC Hobby.

Par défaut, on dispose de deux fichiers .py : boot.py et main.pay. Boot.py sert à configurer les fonctionnalités de la carte au démarrage. La configuration par défaut de boot.py convient à la plupart des cas de figure, il ne sera donc pas nécessaire de le modifier. Après l'exécution de boot.py, MicroPython charge le fichier main.py qui contient le script utilisateur, le "programme" à exécuter. C'est donc dans main.py qu'il faut modifier pour insérer votre script Python.

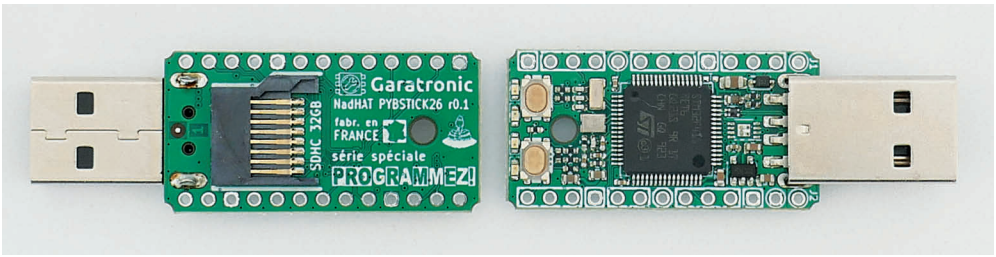
Hello world sauce NeoPixels

Que serait la programmation sans un hello world ? Nous aurons besoin de :

1 Carte PybStick ;
2 Hearders (soudés) ;
3 fils ;
1 NeoPixels 8 LEDS

Soudez sur la NeoPixel trois fils : GND (près de DIN), DIN (données) et 5VDC. Laissez le GND (sous le 5V) vide. On connecte la GND sur GND, DIN sur la broche S19 (MISO) et enfin sur la broche 3,3V.

Côté logiciel, il faut récupérer la librairie ws2812 que l'on met directement à la racine (mais vous pouvez créer un dossier lib si vous en avez plusieurs) puis le code d'exemple :
https://github.com/mchobby/pyboard-driver/blob/master/PYBStick/examples/neopixel_simple.py



Pour aller plus vite, on colle le code directement dans le main.py (méthode sale, mais la plus rapide).

En soi, le code n'a rien de méchant : on déclare la librairie from ws2812 import NeoPixel. Et déclare en même temps le timer (time / sleep). On définit les caractéristiques du NeoPixel :

np = NeoPixel(spi_bus=1, led_count=8, intensity=1). Le SPI est donc le bus utilisé, le led_count donne le nombre de LED RGB présente sur le stick.

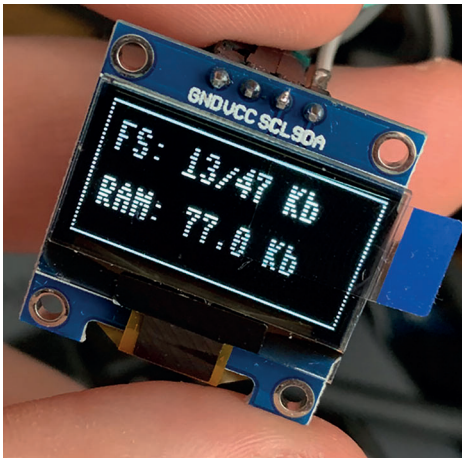
Chaque LED est paramétrable et comme il s'agit de RGB, on gère le rouge, le vert et le bleu. Le code est très simple : np[0] = (255,0,0). [x] indique la LED. Notez que l'on démarre par la LED 0 et non la LED 1, donc on manipule de 0 à 7 et non de 1 à 8. Les montages présentés sont repris des tutoriels sur la carte de MC Hobby.

Hello world sauce OLED !

Les petits écrans OLED 0,97'' sont parmi nos écrans favoris en IoT et les prototypes divers et variés. Peu cher (-1,9 $), ils sont petits et faciles à programmer.

Pour ce faire, il faut un OLED, 4 fils et bien entendu, la PybStick. La connexion est très simple :

SDA -> S3
SCL -> S5
GND -> GND
VCC -> 3,3V



Côté logiciel, il nous faudra installer la librairie SSD1306 (lib classique dans ce type d'écran). L'exemple donné par MC Hobby affiche le stockage interne libre et la RAM disponible. La subtilité du code est d'afficher une jauge. La SSD1306 gère plutôt bien le graphisme.

Si vous ne souhaitez pas l'OLED, vous pouvez parfaitement utiliser une NeoPixel pour afficher le niveau en activant ou non les différentes LED.

Support d'Arduino

MicroPython c'est top, mais que diriez-vous d'un peu d'Arduino ? La carte, ayant un microcontrôleur SMT32, est depuis quelques semaines nativement supportée

sur Arduino IDE et ça, c'est que du bonheur. Il suffit d'installer les cartes SMT32 ou mettre à jour si vous l'avez déjà dans la liste des cartes reconnues par votre IDE. Et vous verrez l'édition Programmez!

3 choses à retenir :

- Pour utiliser la carte, il faut impérativement être en mode DFU (quand vous insérez la carte sur le port USB, pressez le bouton DFU). Le mode DFU doit être sélectionné dans la configuration de la carte (Arduino IDE) ;
- L'outil SMT32Cube Programmer doit être installé. Il est possible d'utiliser dfu-utils (non testé à la rédaction) ;
- Les broches à considérer sont celles de la STM32 et non liées au MicroPython.

Attention : penser à redémarrer Arduino IDE après l'installation de la carte dans l'IDE et l'outil dfu.
Attention 2 : comme nous sommes en mode DFU, vous n'avez plus accès à la partie MicroPython…

Par exemple, si on met une LED sur la broche S19, elle correspondra à la broche 11 en Arduino. Donc, il faut la déclarer en 11, par exemple : int led=11 ;

Header gauche (de haut en bas)		Header droite (de haut en bas)	
Arduino	MicroPython	MicroPython	Arduino
3,3v		VBUS	
0	3	VIN	
1	5	GND	
2	7	8	3 / A0
GND		10	4 / A1
6	11	12	5 / A2
7/17 / A6	13	RESET	
8/18 / A7	15	16	9/19
3,3v		18	10/20
11/A3	19	GND	
12	21	22	14
13/A4	23	24	15
GND		26	16 / A5

Attention : la numérotation se fait avec le microcontrôleur sur le dessus.
Attention 2 : vérifiez bien la qualité de la soudure pour éviter tout mauvais contact.
Pour les différentes fonctions I2C, SPI, reportez-vous au schéma des broches.

Typiquement, en Arduino, vous allez pouvoir accéder aux broches digitales et analogiques. Pour faciliter les branchements, nous vous conseillons une planche à pain.

Documentation & montages

Quelques exemples pour démarrer :
https://github.com/mchobby/pyboard-driver/tree/master/PYBStick

Projets divers avec la Pybstick :
https://github.com/mchobby/pybstick-projects

Informations sur la PyBoard :
http://docs.micropython.org/en/latest/pyboard/general.html

La carte sur Framboise314 :
https://www.framboise314.fr/carte-pybstick26-micropython-avec-le-raspberry-pi/

Le blog de MC Hobby : https://arduino103.blogspot.com

Review de la préversion :
https://arduiblog.com/2020/04/27/presentation-de-la-pybstick/

Conseils & précautions

Installer un bouton RESET

La carte ne possède pas de bouton RESET. Vous pouvez en installer un très simplement. Il vous faut un bouton poussoir, des fils. Connectez les broches GND et /Reset (entre les S12 et S16). Puis installez le bouton poussoir. C'est tout ! Attention : le RESET interrompt toutes les transactions et opérations en cours sur la carte. À utiliser avec prudence.

Que faire si la carte est corrompue ?

Contrairement à Arduino, la PYBStick gère, avec la mémoire flash de son microcontrôleur, un espace de stockage de données. Du coup, il ne faut pas la débrancher n'importe quand. Depuis le bureau, il faut l'éjecter par logiciel comme pour une clé USB avant de la débrancher du port de l'ordinateur.

Si une LED orange clignote tout le temps ou si en montant la carte, un message indique une corruption de la carte, une remise à 0 devrait suffire. Pour ce faire, il faut reformater le système de fichier : appuyer sur le bouton USER et le relâcher quand les LED verte et jaune s'allument en même temps. Après quelques secondes, une LED rouge s'allume puis s'éteint. La carte est de nouveau fonctionnelle. Vous vous retrouverez avec les fichiers .py (main & boot).

Notre conseil : ayez toujours une copie de fichiers et librairies.

Pour en savoir plus, consultez la documentation de la PyBoard :
http://docs.micropython.org/en/latest/pyboard/general.html#boot-modes

Voltage

La carte supporte une tension de 3,3V. Si les capteurs et composants nécessitent du 5V, vous aurez des soucis de fonctionnement.

Un boîtier

Pour protéger la carte et en faciliter son utilisation, vous pouvez imprimer votre propre boîtier. Les fichiers sont disponibles ici : https://www.thingiverse.com/thing:4275160

Stickers pour identifier les broches

Pour vous aider à identifier les broches, vous pouvez imprimer ce petit guide à coller sur les headers :
https://github.com/mchobby/pyboard-driver/blob/master/PYBStick/docs/sticker-connectors.pdf

Où acheter la carte ?

- La série Programmez! est disponible sur notre site internet : programmez.com
- Tarif : **13,99 €** / exemplaire (+ 0,45 € de frais postaux soit un total de 14,44 €)
- Achat en volume à partir de 5 cartes : nous consulter.
- Headers inclus (non soudés). Carte SD non incluse.

Philippe MARTIN
Consultant Freelance
Président de l'association LudikSciences
ludiksciences@gmail.com

PybStick26 + écran TFT 2.4" Ili9341

Habitué à programmer mes Raspberry Pi en Python, j'apprécie la percée actuelle de MicroPython (Python sur les microcontrôleurs). Non pas que je n'aime pas le C++ mais plutôt en raison du temps de compilation et téléversement préalable à l'exécution du code. Un blink est très rapide mais dès qu'un projet devient conséquent, le temps entre deux modifications de code ressemble à une éternité.

Actuellement, une de mes plateformes de prédilection est l'ESP32 notamment en raison de ses 2 Mo de mémoire flash, du Bluetooth et du Wifi. Ayant eu le privilège de pouvoir tester la dernière carte de Garatronic, la PybStick26 j'ai orienté mes tests en privilégiant 2 caractéristiques qui la démarquent d'un ESP32 : le port USB OTG (On-The-Go) et le lecteur de carte SD.

Micropython, carte SD et port USB OTG

Ces 3 éléments sont de réels atouts pour développer des projets. En premier lieu, l'OTG permet d'accéder au contenu de la carte et donc du code comme une simple clef USB. MicroPython étant un langage interprété, toute modification de code est possible avec un simple éditeur de texte et accessible quasi instantanément. Enfin la carte SD apporte une capacité de stockage phénoménale et une vitesse de lecture et d'écriture plus que confortable par rapport aux performance des mémoires flash. A titre de comparaison : ATMEGA328P (Arduino) 32Ko, ESP32 2Mo, PybStick26 47ko mais jusqu'à 32Go avec une microSD. Un fichier de 150Ko se transfère sur la mémoire flash d'un ESP32 en un peu moins de 2min30 contre moins de 2 secondes sur une PybStick26 avec carte SD.

Mon setup

- 1 carte PybStick26
- 1 carte SD de 16Go classe 10
- 1 écran ILI9341 environ 15€ en version tactile, -8 € en version non tactile

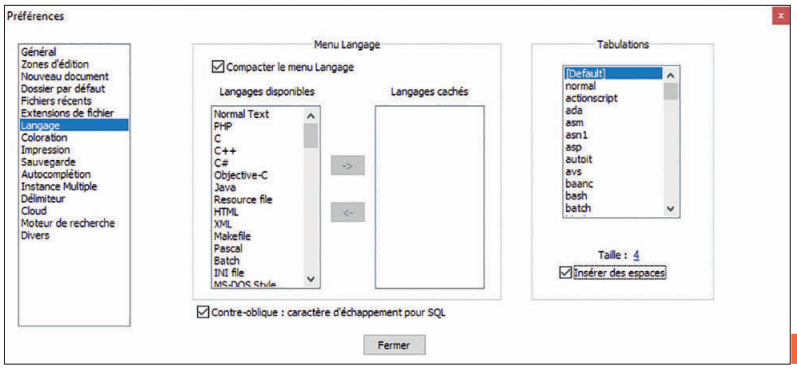
Je travaille sous Windows et pour réaliser ce projet, j'ai utilisé le logiciel PuTTY et l'éditeur de code Notepad++ qu'il convient de configurer pour faire du Python, à savoir éviter le mélange tabulation et espaces (Menu Paramètres/Préférences…) 1

Une fois connectée la carte PybStick est reconnue comme une clef USB. S'il existe une carte SD, le lecteur correspond à celle-ci sinon aux 47ko de mémoire flash. 2

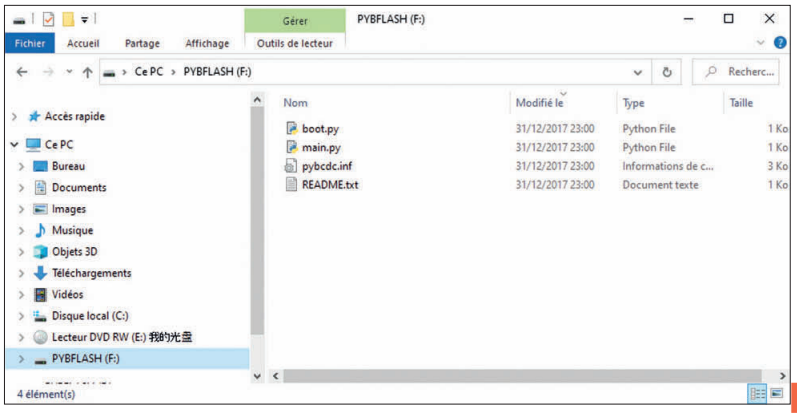
Les deux fichiers nécessaires à son bon fonctionnement sont :

- boot.py exécuté au démarrage de la carte. Il permet de configurer des options si nécessaire.
- main.py qui contiendra le programme exécuté après boot.py .

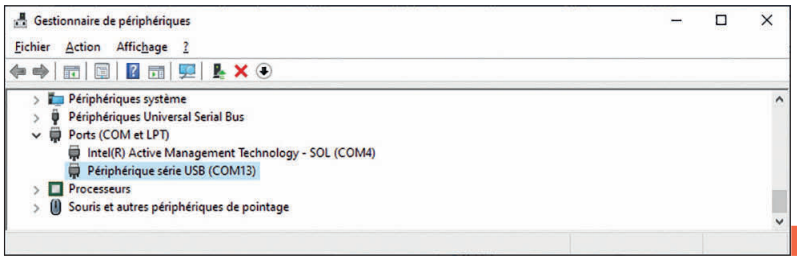
Pour se connecter à REPL sous PuTTY, il nous faut connaître le port COM utilisé. On utilisera le gestionnaire de périphériques : ici il s'agit du COM 13 (il suffit de débrancher/rebrancher la carte pour identifier le port). 3



1



2



3

A travers PuTTY : 4 5

Cette fenêtre nous sera utile pour voir les retours de notre programme, tester une syntaxe ou rebooter la carte via la commande CTRL+D

Câblage

Ecran ILI9341	PybStick26
VCC (+5V)	VIN
GND	GND
CS	S11
RESET (facultatif)	S3
DC	S5
SDI(MOSI)	S19
SCK	S23
LED	3,3V
SDO(MISO)	S21

Nous alimenterons l'écran en 5V et les LED en 3,3V en continu. 6
Nous utiliserons une liaison SPI (Serial Peripheral Interface) synchrone créée par Motorola qui fonctionne en full duplex (dans les 2 sens simultanément), la communication est réalisée selon un schéma maître-esclave et nécessite 4 pins :

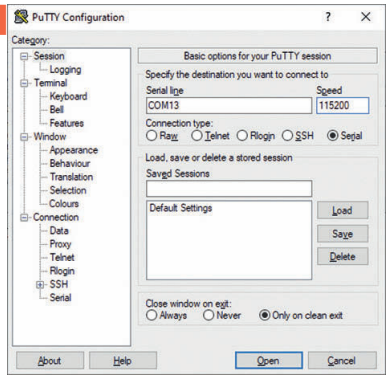
- SCLK : Serial Clock, Horloge (généré par le maître).
- MOSI : Master Output, Slave Input (généré par le maître).
- MISO : Master Input, Slave Output (généré par l'esclave).
- SS : Slave Select, Actif à l'état bas (généré par le maître) et appelé ici CS.

Le pin DC est utilisé pour distinguer commande et valeur (0=commande, 1=data) et enfin le pin RESET sert à initialiser ou réinitialiser la puce.

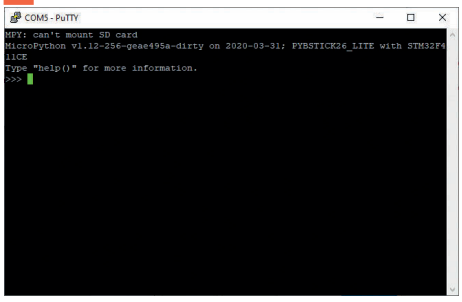
ECRAN TFT 2.4" SPI 240x320 avec ILI9341

Les spécifications de cette puce font 245 pages.
Pour simplifier, ce type de puce est piloté par des commandes suivies de données envoyées sous la forme de bits (0 ou 1) généralement constitués en octets (Byte en anglais).
Dans notre cas, je vais utiliser une bibliothèque la plus simple possible afin d'éviter les problèmes de dépendances, d'arborescence de fichiers ou d'utiliser du code qui pourrait ralentir notre affichage.

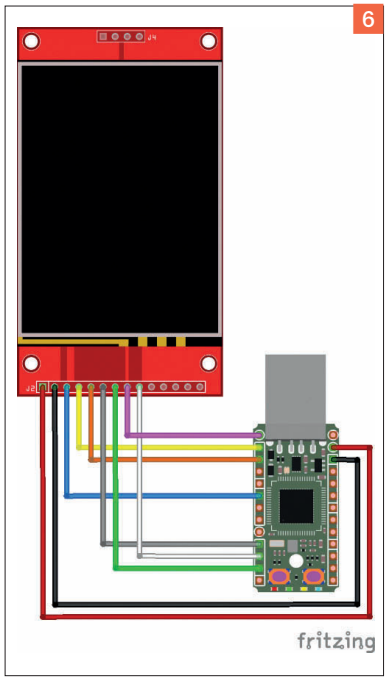
4



5



6



J'ai choisi une bibliothèque développée par Radomir Dopieralski. Il suffira de copier/coller le fichier ili9341.py sur la carte SD et elle nous servira essentiellement à ne pas devoir coder la séquence d'initialisation de la puce.
Niveau caractéristiques, l'écran a une résolution de 240x320 pixels. Il accepte les couleurs en deux formats de 16 bits et 18 bits. Pour introduire la difficulté que nous allons devoir surmonter, il suffit de constater que 16bits correspondent à 2 octets et qu'une image fait 240x320x2 = 153 600 octets soit 150ko (1ko = 1024 octets). Notre microcontrôleur dispose de 96ko dont environ 65ko réellement exploitables, ce qui nous prive de pouvoir monter en mémoire une image de la taille de l'écran en une fois.

Généralités sur les bits, les octets et les valeurs hexadécimales en Python

Pour comprendre la suite, des opérations, il convient d'avoir quelques notions pour passer d'une unité à l'autre.
Un octet est composé d'une suite de 8 bits et il se représente de la manière suivante :

Position	7	6	5	4	3	2	1	0
Poids	128	64	32	16	8	4	2	1
Binaire	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**

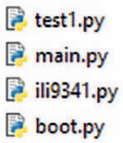
150 soit (128+16+4+2) s'écrit : 0b10010110
Pour vous exercer, vous pouvez lancer dans PuTTY, les commandes suivantes :

```
>>> bin(150)
'0b10010110'
>>> hex(150)
'0x96'
>>> hex(0b10010110)
'0x96'
>>> bin(0x96)
'0b10010110'
```

Alors que vient faire l'hexadécimal ? Eh bien c'est une représentation en base 16 donc intéressante pour des pixels en 16bits et plus compacte en écriture que les bits.

Premier programme de test

Notre écran propose une résolution de 240x320 pixels et nous allons commencer par remplir une page en bleu, histoire de voir comment les pixels s'affichent. Pour cela on dépose le fichier test1.py sur la carte SD :



Puis on lance via PuTTY, la commande **import test1** qui va exécuter le fichier test1.py

```
import ili9341 # Import de la bibliothèque ili9341
from time import * # Nécessaire pour mesurer les temps d'affichage
from machine import Pin, SPI # Bibliothèque de gestion des entrées sorties

# Initialise la liaison SPI à une fréquence de 24Mhz avec 3 pins matériels dédiés PybStick26
```

```
spi = SPI(baudrate=24000000,miso=Pin("S21"), mosi=Pin("S19", Pin.OUT),
 sck=Pin("S23", Pin.OUT))
# Objet display basé sur la classe ILI9341 et initialisé avec le SPI et les deux
derniers pins de notre montage
display = ili9341.ILI9341(spi, cs=Pin("S11"), dc=Pin("S5"), rst=Pin("S3"))

t0 = ticks_ms()
display.fill(ili9341.color565(0x00, 0x00, 0xff))
print(ticks_ms()-t0, "ms pour afficher une page bleue")
```

La méthode **fill** va remplir l'écran de pixel à partir d'une couleur en 24 bits (8bits pour le rouge, 8bits pour le vert et enfin 8 bits pour le bleu) via la fonction **color565** qui convertit cette couleur en 16 bits. hex(0) = 0x00 et hex(255)= 0xff, on retrouve ainsi notre couleur bleue à savoir (R,V,B) = (0,0,255)=(0x00,0x00,0xff). Hommage à nos camarades daltoniens. 7

⇒ Dans Putty on affiche : 501 ms pour afficher une page bleue

Attention, en Python, le contenu d'une bibliothèque n'est exécuté qu'une seule fois si elle est importée plusieurs fois. Pour cette raison, si on souhaite réutiliser la commande import test1, il conviendra au préalable de rebooter la carte via CRTL+D dans PuTTY.

Comment ça marche ?

Si vous avez été attentif, vous avez identifié que l'écran se remplit du bas vers le haut mais sans doute pas si les lignes se remplissent de droite à gauche ou l'inverse. Cette méthode **fill** de la classe ILI9341 réalise deux actions distinctes :

1) définir la géométrie de la zone à alimenter ;
2) alimenter en pixels 2 octets par 2 octets.

Nous allons voir 3 commandes nécessaires pour effectuer ces opérations « manuellement » en copiant le fichier test2.py sur notre carte SD.

```
import ili9341 # Import de la bibliothèque ili9341
import ustruct # Module de conversions des valeurs Python en structures C
from time import * # Nécessaire pour mesurer les temps d'affichage
from machine import Pin, SPI # Bibliothèque de gestion des entrées sorties

# Initialise la liaison SPI à une fréquence de 24Mhz avec 3 pins matériels dédiés PybStick26
spi = SPI(baudrate=24000000,miso=Pin("S21"), mosi=Pin("S19", Pin.OUT), sck=Pin
("S23", Pin.OUT))
# Objet display basé sur la classe ILI9341 et initialisé avec le SPI et les deux derniers pins
de notre montage
display = ili9341.ILI9341(spi, cs=Pin("S11"), dc=Pin("S5"), rst=Pin("S3"))

# On définit la taille de la zone : ici la totalité de l'écran
width = 240
height = 320

t0 = ticks_ms()
display._write(0x2a, ustruct.pack(">HH", 0, width-1))
display._write(0x2b, ustruct.pack(">HH", 0, height-1))
display._write(0x2c)
display._data(b'\x00\x00'*(50*width+60))
display._data(b'\x00\x1f'*(51*width-(width-60)))
print(ticks_ms()-t0, "ms pour afficher 50 + 1/4 lignes noires et 50 + 3/4 lignes bleues")
```

On ajoute la librairie ustruct incluse dans MicroPython qui correspond à struct en Python. Pour les détails de syntaxe, voir https://docs.python.org/3/library/struct.html .

On adresse la commande 42 (0x2a en hexadécimale) qui correspond à la définition des adresses de colonnes suivie des données correspondantes (numéro de colonne de départ et numéro de colonne de fin). Dans notre cas, toute la largeur de l'écran.

Puis la commande 43 (0x2b en hexadécimale) qui correspond à la définition des adresses de lignes suivie des données correspondantes (numéro de ligne de départ et numéro de ligne de fin). Dans notre cas, toute la hauteur de l'écran.

Ensuite, on adresse la commande 44 (0x2c en hexadécimale) qui correspond à la mémoire associée à la zone définie avec les commandes précédentes.

On envoie la couleur noire b'\x00\x00' encodée sur 2 octets sur 50 lignes plus 60 pixels.

Suivie de la couleur bleu b'\x00\x1f' sur 51 lignes moins les 60 pixels de la ligne noire. 8

7

8

⇒ Le décroché de pixels noirs se situant à droite, on sait à présent que l'écran se rafraîchit de bas en haut et de droite à gauche.

Pour choisir d'autres couleurs :

```
>>> import ustruct, ili9341
>>> ustruct.pack(">H",ili9341.color565(0x00,0x00,0xff))
b'\x00\x1f'
```

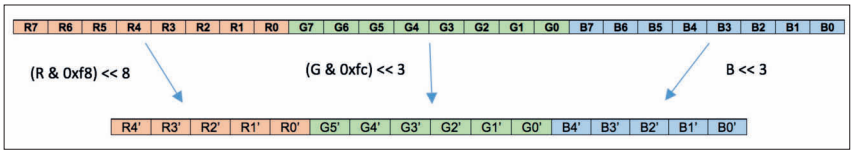
Analyse d'un fichier BMP :

L'intérêt d'un fichier BMP est qu'il est sans compression. Les pixels sont décrits un par un ce qui correspond au fonctionnement de la mémoire de notre écran. Pour comprendre comment est constitué une image BMP, je vous renvoie vers l'excellent article Wikipedia.

Dans notre cas de figure, un fichier BMP de 240 pixels de large sur 320 de haut est constitué d'un entête de 54 octets puis d'une série de pixels en 24 bits soit 3 octets par pixels pour décrire les quantité (0-255) de rouge, vert et bleu. Les pixels sont structurés en lignes et le premier pixel est celui situé en bas à gauche de l'image puis les lignes progressent jusqu'au haut de l'image.

76560	76561	76562	…	76799
…				
240	241	242	…	479
0	1	2	…	239

3 octets par pixel ! Mais l'écran n'en attend que 2. Il va donc falloir faire une conversion.



Par chance celle-ci est largement documentée et elle correspond à la méthode color565 que nous avons à disposition dans la classe ILI9341.

Deuxième sujet d'importance, dans un fichier BMP, le balayage se fait de bas en haut et de gauche à droite alors que notre écran est balayé de droite à gauche. Cela provoquerait un effet miroir peu esthétique. 9

9

Modifier l'orientation de l'écran

L'orientation se situe au niveau de la commande 54 (0x36 en hexadécimale). Cette commande propose un octet de pilotage avec les règles suivantes :

bit 7	Top to Bottom Or Bottom to Top = 1
bit 6	Left to Right Or Right to Left = 1
bit 5	Normal Mode Or Reverse Mode = 1
bit 4	LCD Refresh Top to Bottom Or Bottom to Top = 1
bit 3	RGB Or BGR = 1
Bit2	LCD Refresh Left to Right Or Right to Left = 1
bit 1	Switching between Segment outputs and RAM = 0
bit 0	Switching between Segment outputs and RAM = 0

Par défaut, la bibliothèque initialise l'écran à la valeur hexadécimale b'\x48' correspondant à l'entier 72 ce qui nous donne le paramétrage suivant :

Position	7	6	5	4	3	2	1	0
Poids	128	64	32	16	8	4	2	1
Binaire	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

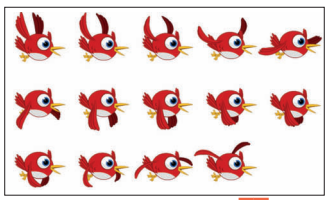
Il suffit de modifier le bit 6 en le passant de 1 à 0, ce qui nous donne : l'entier 8 soit b'\x08'

10

Afficher une image

On va donc copier sur la carte SD deux fichiers, imageV.bmp et test3.py : **Code complet sur programmez.com & github**

Au-delà de la ligne concernant l'effet miroir que nous aurions eu avec le paramétrage initial de l'écran, on ouvre notre fichier bmp, survole les 54 octets d'entête puis récupère les pixels un par un en convertissant leurs valeurs RGB 24bits en 16bits avant de les envoyer vers l'écran. 10


11

La performance est médiocre car il faut 20 secondes pour afficher une image. Cet algorithme peut s'améliorer notamment en gérant ligne par ligne au lieu de pixel par pixel mais je n'ai pas pu descendre en dessous des 5 secondes car la bascule 24 bits/16 bits reste nécessaire pour chaque pixel.

Afficher l'image plus rapidement

Un moyen d'afficher l'image plus rapidement est de la traiter en amont et de finalement « streamer » le fichier à l'écran. Nous utiliserons le fichier bmp_conversion.py depuis le PC. En positionnant l'image imageV.bmp dans un sous-répertoire Pictures. Ce script supprime l'entête et convertit les pixels en 16bits pour une lecture directe.
Il va créer un nouveau fichier nommé imageV16.bmp qui devra être copié sur la carte SD avec le fichier test4.py :
Code complet sur programmez.com & github


12

Le nom de fichier a été remplacé par notre nouveau fichier et nous créons des blocs de pixel à « streamer » puisque notre fichier fait 150Ko et que nous ne disposons que de 65ko de mémoire. Il sera lu en 4 fois. Notre temps d'affichage de l'image est passé de 20 secondes à 563ms soit à peine 62ms de plus que l'affichage de l'image bleue.

L'animation

A présent, il est évident qu'essayer d'animer la totalité de l'écran n'est pas réaliste avec seulement 2 images/seconde. J'ai récupéré, pour l'exemple, une image avec des sprites. A l'aide d'un script Python, j'ai découpé cette image en 14 images de 123x114 pixels. 11
Ces images ont été transformées à l'aide du script bmp_converter.py en 14 fichiers de données brutes qui seront chargés les uns à la suite des autres sur une partie de l'écran. Pour obtenir l'animation, copiez les 14 fichiers nommés birdXX.bmp ainsi que le fichier animation.py sur la carte SD :
Code complet sur programmez.com & github

On ajoute deux variables pour la taille des sprites puis l'écran est basculé en blanc. La zone de mémoire est fixée à la taille d'un sprite centré sur l'écran. Deux variables sont créées pour compter les images qui vont être utilisées et définir une durée d'animation. Tant que le temps d'animation n'est pas dépassé, on ouvre en boucle les 14 fichiers l'un après l'autre. 12
L'animation est fluide et elle se joue à 8,4 images/seconde

Conclusion

Nous avons vu un exemple d'exploitation des microcontrôleurs en fonction de leurs points forts. Le code présenté se porte sans réelle difficulté sur ESP32/MicroPython. Il faut modifier les lignes de déclaration de pins et remplacer les appels high() et low() par value(1) et value(0) dans la bibliothèque Ili9241. A titre de comparaison, 1426ms pour la page bleue, plus de 36 secondes pour l'image avec décodage des pixels à la volée, 1469ms pour l'image streamée et enfin une animation moins fluide à 4,2 images/s.
Bien entendu, il ne s'agit que d'un POC et il serait intéressant d'intégrer ce code directement dans une bibliothèque pour faciliter l'écriture de routines plus complexes mixant d'autres types d'affichages. Une piste d'optimisation pourrait être de coupler toutes les images à la suite dans un même fichier. De même, une vidéo convertie via ffmpeg en ajustant résolution et framerate au format rgb565le devrait être diffusable suivant le même principe.

Sources

Lien GitHub vers les fichiers :
https://github.com/philippeminerve/PYBStick26/tree/master/micropython/ILI9341_POC
Documentation de la puce ILI9341 :
https://www.newhavendisplay.com/app_notes/ILI9341.pdf
Bibliothèque ili9341 par Radomir Dopieralski :https://bitbucket.org/thesheep/micropython-ili9341/src/default/ili9341.py
Il existe d'autres bibliothèques en MicroPython et l'une des plus intéressantes est sans doute celle développée par Roman Podgaiski :
https://github.com/ropod7/pyboard_drive/tree/master/ILI9341
Je vous conseille d'y accéder via cette page du Github de MCHobby pour pouvoir bénéficier de documentation notamment sur la gestion des fontes : https://github.com/mchobby/freetype-generator
Documentation Wikipedia Format fichier BMP :
https://en.wikipedia.org/wiki/BMP_file_format
Sprites exemples par Vlad Alekseev :
https://www.kindpng.com/imgv/wobRbi_math-book-covers-flying-png-logo-sprite-sheet/

Franck Dubois
Video Game Codeur
Développeur agile
Formateur javascript / node.js / Unity / GDevelop

Coder un classic 2D avec GDevelop

A l'aide de l'outil GDevelop (version 5 téléchargeable depuis https://gdevelop-app.com/fr/download/), je vais vous faire découvrir quelques aspects du codage et de la création d'un jeu vidéo 2D en prenant comme modèle le célèbre Bubble Bobble.

Focus sur le gameplay 1

Votre objectif en tant que joueur est de passer d'un niveau à l'autre en éliminant au préalable tous les robots ennemis dudit niveau. Pour ce faire, vous disposez d'une arme prenant la forme d'une bulle que vous lancez. Cette dernière a la faculté d'emprisonner les ennemis si ceux-ci ont le malheur de la croiser. Dans ce cas et après emprisonnement, la bulle éclate après 4 secondes ou lorsqu'elle atteint le haut de l'écran. Le robot est alors éliminé.

Chaque niveau se compose d'un écran fixe dans lequel sont disposées des plateformes sur lesquelles vous pouvez vous déplacer ou encore sauter. Les robots, en nombre limité, démarrent leur course du haut de l'écran, pour rejoindre le bas en se déplaçant sur les plateformes. S'ils chutent des plateformes du bas de l'écran, ils réapparaissent en haut. C'est aussi le cas pour le joueur. Les ennemis ne sont pas inoffensifs puisque s'ils vous touchent, vous perdez une vie sur les trois qui vous sont données au démarrage du jeu.

Enfin, sachez que si vous perdez une vie, tout n'est pas perdu puisque certains bonus (des fruits) qui chutent après la mort d'un ennemi peuvent vous en faire récupérer une. D'autres bonus pourraient être ajoutés comme par exemple l'accélération de votre vitesse de déplacement.

Les déplacements du joueur se font au clavier à l'aide des touches directionnelles, les sauts sont exécutés sur appui de la barre espace, et la touche ctrl gauche lance les bulles tueuses.

Au titre d'extensions, les robots pourraient être plus ou moins rapides ou encore plus agressifs en étant équipés d'une autre arme telle qu'un laser.

1

Les ressources

Tous les éléments graphiques du jeu sont distribués dans le cadre de la CC BY-NC-SA 3.0. Ils sont empruntés à Dan Malone, graphiste de renommée internationale, qui les a créés pour « **Code The Classics – Volume 1** », livre consacré à la création de jeux vidéo en langage Python.

Ces assets sont téléchargeables depuis l'adresse https://www.video-game-codeur.fr/download/programmez/cavern.programmez.zip

Petite présentation de GDevelop et sa philosophie

GDevelop est un outil open source de développement de jeux vidéo 2D. Ce qui rend intéressant la chose est d'une part un IDE fort bien fait et surtout une utilisation de concepts, certes moins poussés, mais connexes à ceux utilisés par Unity ou Godot-Engine. Le tout avec un bon assistant syntaxique et une galerie d'extensions. Bref, un système bien intégré qui rend intuitive la création d'un jeu vidéo. Il en devient alors une excellente mise en jambe à celui, profane inclus, qui souhaite se familiariser avec ces concepts, le code informatique et la création de jeux vidéo avec de rares limites.

Vous êtes débutant ? Ne fuyez pas et lisez plutôt la suite.

"GDevelop is an easy-to-use game creator with no programming language to learn". Beaucoup de choses se font visuellement : vous allez créer vos objets et les animer en quelques clics de souris directement sur l'espace de jeu. Vous donnez vie à ces objets sans avoir de langage à apprendre. Il n'en reste pas moins que vous allez devoir utiliser des instructions non pas exprimées en pur langage informatique, mais avec une syntaxe proche des langues que vous utilisez au quotidien : l'anglais ou encore le français, l'IDE est aussi polyglotte. Par cette approche, le débutant se familiarise avec la programmation et ses logiques, sans s'attacher sur un langage en particulier. Pour les autres, l'entrée en matière est simple. Tous les exemples donnés dans cet article ont été réalisés sur une configuration en langue anglaise.

Certains développeurs polémistes entameront un débat stérile prétextant que ce n'est pas du code ou que ça n'a rien à voir avec du développement. J'ai tendance à être un développeur pragmatique plutôt que sectaire et j'inviterai ces personnes à passer plus de temps à développer pour vraiment appréhender certaines réalités qu'ils semblent ignorer. Le travail réalisé par Florian Rival, créateur de **GDevelop**, mérite incontestablement du respect.

Autres cordes à son arc, cet outil est cross-platform et permet distribuer vos builds à votre guise sur un maximum de plateformes mobiles (publicité admob incluse) ou pas. Il existe par ailleurs de nombreux exemples et tutos disponibles sur le site ainsi qu'une documentation bien fournie et en grande partie traduite.

Deux inconvénients mineurs que sont la génération des livrables se faisant en ligne et limitée en nombre par tranche de 24h et l'ajout d'un splash screen **GDevelop** au lancement du jeu. Toutefois, il vous est possible de souscrire à une option payante qui vous per-

mettra de faire un plus grand nombre de builds et de retirer, si vous le souhaitez, le splash screen.
Une dernière chose, et bien que je déconseille cette approche, **GDevelop** propose aussi des jeux préfabriqués (plateforme, shoot'em up, space invaders, 3d isométrique...) qu'il vous suffira de modifier pour en faire les vôtres.
Voilà qui est dit. Je vais maintenant vous parler des incontournables à connaître de **GDevelop** pour partir sur de solides bases.

Les incontournables à connaître pour utiliser GDevelop

Scène

Tout comme le 7e art, un jeu vidéo comporte différentes scènes. Pour faire simple, chaque écran d'un jeu vidéo est une scène : une cinématique, un menu, un niveau, les meilleurs scores, etc.
Sous **GDevelop**, un jeu vidéo est donc constitué de scènes indépendantes les unes des autres : elles peuvent être travaillées indépendamment les unes des autres. Elles sont ensuite reliées, tout comme sont montés les films au cinéma, pour donner naissance au jeu dans sa globalité. Les scènes, par nature, n'ont pas vocation à coexister à un même moment lors de l'exécution du jeu.

Préfabriqués

Comme je le disais précédemment, **GDevelop**, à l'instar de Unity, possède aussi des projets préfabriqués qu'il suffit de modifier pour les prendre à votre compte et créer vos propres jeux. Il est ainsi possible de créer rapidement un jeu de plateforme, un shoot'em up et bien d'autres. Un bémol : vous allez produire du jeu vidéo stéréotypé, sans âme et sans originalité. On retrouve le même phénomène avec Unity et ses projets préfabriqués.

Objets

Votre jeu vidéo sera constitué uniquement d'objets et rien que des objets auxquels vous pouvez adjoindre :
- des caractéristiques sous des forme de propriétés ;
- et des comportements (behaviors) facultatifs à choisir en fonction de votre besoin.

Comportements

Ces comportements sont là pour vous faciliter la vie. Associer un comportement à un objet permet d'implémenter des fonctionnalités, sans coder. Sans comportement, votre objet est inerte. Il vous appartient alors de l'animer manuellement avec du code ou via les comportements.
Pour être plus parlant, si à un objet Sprite (une image que vous pouvez déplacer sur l'écran) vous donnez un comportement de type « **PlatformerObject** », vous en faites alors un personnage joueur (PJ) que vous pouvez déplacer et faire sauter avec le clavier comme dans un jeu de plateforme style Mario Bros.
Un autre exemple est d'associer un comportement physique « **Physics2** » pour que votre Sprite puisse être affecté par la gravité ou encore d'autre forces. J'en profite pour faire un lien puisque ce comportement physique peut être Dynamic, Static ou en Kinematic, tout comme les Rigidbody sous Unity.
Pour le comportement « **Physics2** », le type **Dynamic** est celui par défaut. Dans ce mode, votre Sprite est affecté par la gravité (il tombe), peut se mouvoir de tout côté, se cogner à tout autre objet (collision) et en être affecté : sa trajectoire peut alors être modifiée. En mode **Static**, il se cogne uniquement (entrer en collision) aux objets en mode **Dynamic**, et n'est affecté ni par la gravité ou tout autre force. Enfin le mode **Kinematic** est identique au mode **Static** avec la possibilité de donner une vitesse de déplacement et d'entrée en collision avec les sprites en mode **Dynamic**.
Avec **GDevelop**, vous ne produisez que du jeu en 2D. Vous déplacez donc les objets uniquement sur deux dimensions en largeur (abscisses) et hauteur (ordonnées) dans un système de coordonnées cartésiennes orienté vers la droite et le bas de l'écran.

L'IDE en quelques mots

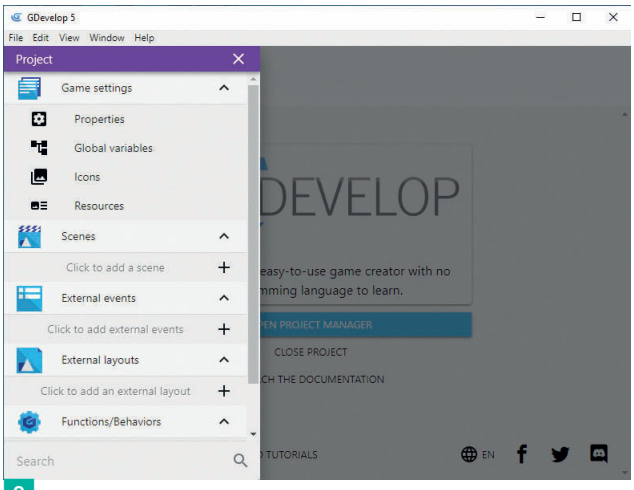
Lorsque **GDevelop**, est lancé vous avez le choix entre deux options :
- ouvrir un projet existant (OPEN A PROJECT) ;
- créer un nouveau projet (CREATE A NEW PROJECT).

Choisissez la 2e option puisque nous démarrons le projet de rien, puis sur la liste de projets préfabriqués, sélectionnez le seul qui n'en est pas un : « **Empty Game** ». L'interface du projet s'ouvre alors. Pensez dès à présent à sauvegarder le projet via le menu « **File** / **Save As** ». 2
Vous pouvez basculer l'IDE en français par le biais du menu « **File/Language** ».
Ceci fait, l'éditeur est composé d'un menu sur la partie gauche de l'écran subdivisé en sous-menus, notamment :
- **Game settings** pour accéder aux paramètres du jeu (nom, version, résolution...) ;
- **Scenes** pour créer vos différents écrans de jeu ;
- **Functions/Behaviors** pour créer vos propres fonctions ou comportements d'objet.

Pour accéder au sous-menu, il suffit de cliquer sur le chevron « ^ ».

Organisation du projet

Avant de commencer, il faut organiser l'arborescence de votre projet afin d'y placer les ressources. Rien de compliqué puisqu'il suffit d'abord de créer un dossier pour le projet que vous nommez **cavern** puis d'y reprendre à l'intérieur l'arborescence des assets du jeu (images, music and sound). Ensuite, sauvegardez le projet dans ce même dossier **cavern** via le menu « **File/Save As** » : le système ouvre une fenêtre dans laquelle vous indiquez le nom du fichier



2

projet, qui au passage est au format json, ainsi que le dossier dans lequel il est créé.

Création de la première scène

La scène qui nous intéresse est celle du jeu lui-même. Dans le menu **Project**, cliquez sur le « ^ » situé à côté de **Scenes**, puis dans le sous-menu, cliquez sur « + » pour ajouter une scène. Une scène nommée « **NewScene** » apparaît dans ce même sous-menu. Nous renommons tout de suite la scène en « **Level1** » via le menu « **Rename** » accessible à partir des 3 points en vis-à-vis du nom de la scène « **NewScene** ». Entrez dans l'éditeur en double-cliquant sur le nom de la scène « **Level1** ».

L'éditeur de scène de jeu comporte deux onglets :

- le premier qui est l'espace graphique dans lequel vous placez vos objets graphiques : 3

Auquel est rattachée la barre d'icônes suivantes :

0 - afficher le menu projet
1 - lancer la scène
2 - debugger
3 - afficher l'éditeur d'objets
4 - afficher l'éditeur de groupes d'objets
5 - afficher les propriétés
6 - lister les objets de la scène
7 - lister les calques ou layers
8 - afficher une grille
9 - zoomer l'éditeur de la scène

- le second qui est l'éditeur de code, des actions inscrites dans une boucle infinie d'événements exécutés 60 fois par seconde. 4

Auquel est rattachée la barre d'icônes suivantes :

10 - créer une condition/événement
11 - créer une sous-condition/événement
12 - ajouter un commentaire
13 - ajouter une structure de code
14 - supprimer une condition/événement
15 - annuler le dernier changement
16 - remettre le dernier changement annulé

Gardez en visibilité les icônes et la numérotation puisque j'y ferai référence dans la suite de cet article.

Création du décor

Le décor est constitué du fond d'écran qui n'a pas vocation à interagir ainsi que des plateformes sur lesquelles se déplaceront tous les protagonistes du jeu. Pour une question d'organisation, vous allez créer et utiliser un calque spécifique pour chacun : le premier nommé « **background** » et le deuxième « **game** ». Les calques sont des feuilles virtuelles superposables les unes aux autres : les éléments d'un calque placés au-dessus d'un autre masquent les éléments du calque du dessous. Sous **GDevelop**, l'ordre de superposition des calques dans l'éditeur correspond à l'ordre de leur superposition dans le jeu : le premier de la liste étant celui au premier plan et le dernier au plan le plus inférieur.

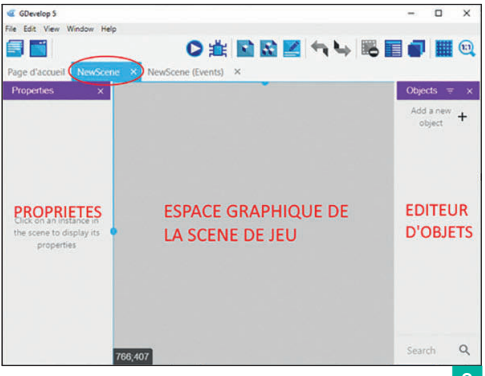
Affichez la liste des calques en cliquant sur l'icône 7 dans laquelle figure un calque créé par défaut nommé « **Base layer** ». Pour ajouter un calque, cliquez sur le bouton « **ADD A LAYER +** » : un nouveau calque apparaît en tête de liste avec l'éditeur du nom activé. Il ne vous reste plus qu'à nommer le calque créé « **background** ». Répétez l'opération pour le calque « **game** ». 5

Vous pouvez les déplacer les uns par rapport aux autres par un simple glisser/déposer (drag & drop) et les rendre visibles ou invisibles. Il existe un calque par défaut nommé « **Base layer** » dans lequel sont placés tous les objets que vous ajoutez manuellement à la scène que vous ne pouvez pas supprimer. Je vous invite à le placer au plan le plus inférieur. Le bouton « **ADD EFFECT** » quant à lui permet d'appliquer des filtres sur le calque tels qu'un effet écran CRT ou encore effet pixélisé. Bref, il y en a toute une série à découvrir.

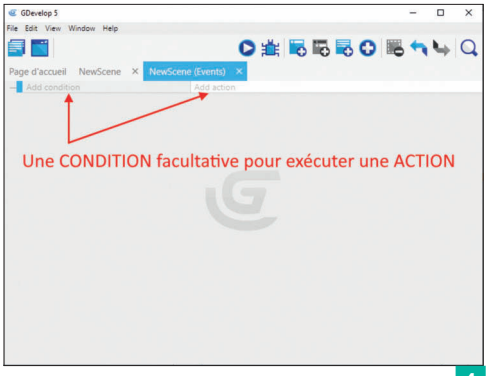
Le dossier « **images/level1** » des ressources contient les fichiers spécifiques au niveau 1. Nous ne nous attardons ici que sur le niveau 1 et c'est donc le fichier « **level1.png** » qui nous intéresse dont les dimensions sont de 800 sur 480 pixels.

Par défaut, les scènes ont une dimension de 800 sur 600 pixels. Deux solutions : redimensionner l'image avec des imperfections possibles ou coller au mieux l'image en redimensionnant la scène. La seconde solution ayant ma préférence, cliquez sur l'icône 0 de la barre pour afficher le menu « **Game settings** », et cliquez sur « **Properties** ». C'est depuis cette fenêtre que se font les paramétrages du jeu et notamment la résolution « **Resolution and rendering** » : remplacez 600 par 480 dans le champ « **Game resolution height** » et validez en cliquant sur « **Apply** ». Cliquez sur la scène pour revenir sur l'onglet « **Level1** ».

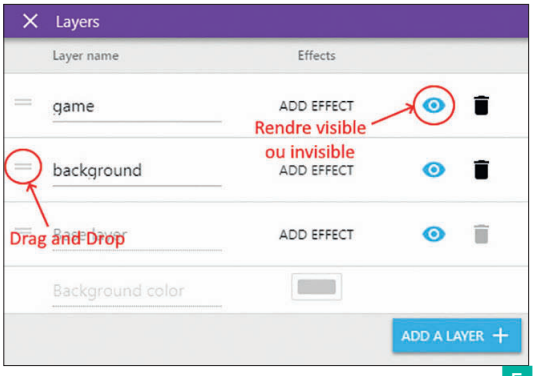
Ajoutez le fond d'écran « **level1.png** » en cliquant sur le « + » situé à côté de « **Add a new object** ». L'éditeur vous propose alors différents types d'objets à créer. Sélectionnez le premier de la liste « **Sprite** ». Le fond d'écran n'est pas un sprite en tant que tel puis-



3



4



5

qu'il n'a ni vocation à interagir ou à bouger. Une fois placé, il va donc falloir verrouiller sa position.

Vous avez sélectionné « **Sprite** », l'éditeur affiche un popup pour nommer l'objet dans le champ « **Object name** », tapez « **background** » comme nom. Vous devez ensuite lui associer l'image « **level1.png** ». Cliquez sur « **ADD A ANIMATION +** », ensuite importez le fond d'écran du dossier « **images/level1** » en cliquant sur le bouton « **+ ADD** ». Il reste juste à intégrer l'objet « **background** » dans la scène par un glisser/déposer (drag and drop) et d'ajuster son placement afin qu'elle recouvre la totalité de la scène. Pensez à lui affecter le layer « **background** » depuis le menu « **Properties** » sur la partie gauche de l'écran, via la propriété « **Layer** ».

Passons maintenant aux plateformes qui ont pour base le fichier image **block1.png** faisant une taille de 25 pixels sur 25 que vous trouvez aussi dans le dossier « **images/level1** ». Premier travail, importer ce fichier dans le projet : cliquez sur le « + » situé à côté de « **Add a new object** ». Dans la liste des objets, sélectionnez « **Tiled Sprite** ». Dans le popup qui apparaît, nommez l'objet « **platform** », choisissez l'image « **block1.png** » par le biais du champ « **Select image** ». La taille du bloc doit correspondre à la taille de l'image : 25x25 pixels. Remplacez les 2 valeurs des champs « **Default width** » et « **Default height** » (par défaut à 32) par 25. Vous n'avez plus qu'à sélectionner l'onglet « **BEHAVIORS** » pour lui associer un comportement. 6

Cliquez sur le bouton « **ADD BEHAVIOR TO THE OBJECT +** » pour choisir le comportement désiré. 7

Parmi ceux proposés, celui qui nous intéresse ici est « **Platform** », sélectionnez-le et cliquez sur le bouton « **CLOSE** ». Dans les propriétés du comportement décochez « **Ledges can be grabbed** », option qui permet de se suspendre à une plateforme et validez par le bouton « **APPLY** ».

Avec un seul bloc, vous allez pouvoir dessiner autant de plateformes que nécessaire en les glissant dans la scène par de simples drag and drop de l'objet « **platform** ». 8

Dans les propriétés de l'objet qui apparaissent à gauche de l'écran lorsque l'objet est dans la scène, sélectionnez le calque « **game** », le calque de 1er plan. Vous n'avez plus qu'à redimensionner l'image à la souris pour lui donner la taille souhaitée, et la placer là où vous le souhaitez. Pour chaque nouvelle plateforme, il suffit de glisser une nouvelle fois l'objet sur la scène et de répéter les opérations précédentes, ou bien de faire un simple copier-coller dans l'objet qui est déjà placé. Pour vous faciliter les placements vous pouvez afficher la grille via l'icône 8 (**Show grid**) en paramétrant (**Setup grid**) les cellules (Cell) à 25 sur 25 pixels.

Le personnage joueur et ses animations

La méthode est ici identique à celle utilisée pour le fond d'écran (objet **Sprite**), la seule différence étant que vous allez y ajouter des animations et un comportement. Nommez cet objet « **player** », puis cliquez sur le bouton « **ADD A ANIMATION +** » pour chaque animation que vous créez, chacune devant être nommée par le biais du champ « **Animation #n** » pour pouvoir être appelée.

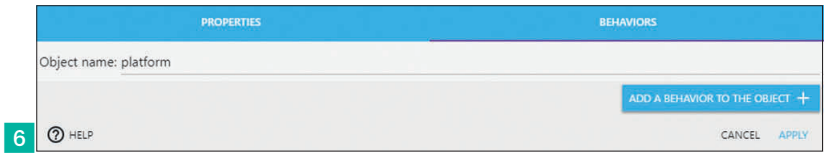
Pour le personnage, il y a 9 animations à créer : le joueur ne bouge pas (**idle**), il se déplace vers la gauche (**moveLeft**), vers la droite (**moveRight**), lance une bulle à gauche (**blowLeft**) et à droite (**blowRight**), saute à gauche (**jumpLeft**) et à droite (**jumpRight**), et meurt (**angel**). Les images à utiliser se situent dans le dossier **images/player**. Pensez à cocher la case « loop » pour les animations « **moveLeft** » et « **moveRight** » : l'animation se joue alors sans interruption.

Pour ajouter une image à une animation, cliquez sur le bouton « **+ ADD** ». Voyez en image ci-dessous, les principales options possibles parmi lesquelles il y a la vitesse et le jeu en continu. 9

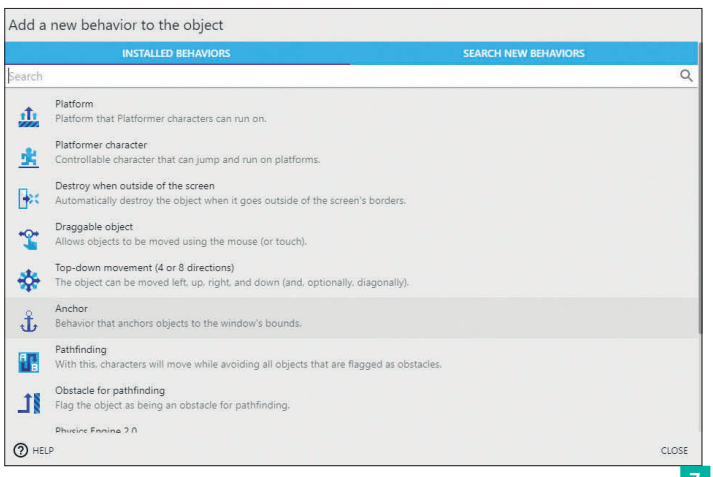
Les déplacements du personnage joueur

Vos animations créées, il vous reste à ajouter le comportement adéquat pour que le joueur puisse se déplacer à l'aide du clavier. Cliquez sur l'onglet « **BEHAVIORS** », puis sur le bouton « **ADD A BEHAVIOR TO THE OBJECT +** ». Comme pour la plateforme, vous avez une série de comportements proposés, choisissez « **Platformer character** » en laissant tous les paramètres par défaut. Une dernière chose, faites un glisser/déposer de l'objet « **player** » vers la scène en le plaçant de préférence sur une plateforme (désactivez la grille pour le placer plus finement). N'oubliez pas de lui affecter le calque « **game** » dans la propriété « **Layer** ».

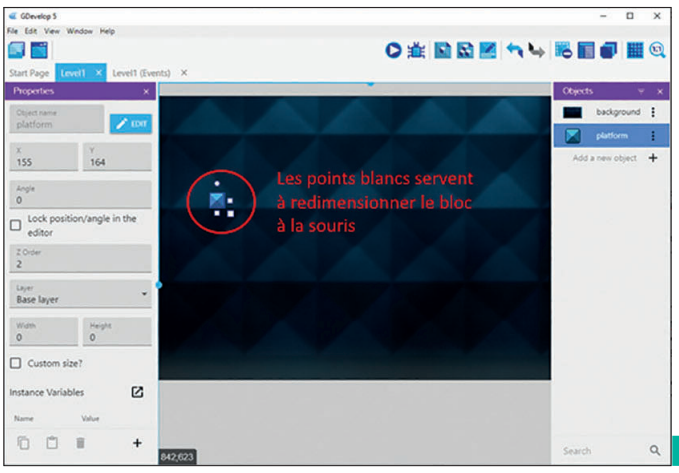
Pour voir la magie s'opérer, lancer la scène en cliquant sur l'icône 1.



6



7



8

Vous bougez votre personnage à l'aide des flèches directionnelles du clavier, et le faites sauter à l'aide de la barre espace. Vous devriez constater aussi qu'il n'y a aucune animation, et c'est là que va entrer en jeu la programmation, le code.

Sélectionnez l'onglet « **Level1 (events)** ». En cliquant sur l'icône 10, vous créez votre 1ère instruction composée d'une condition facultative et d'une action. Pour activer l'animation correspondante du joueur lorsqu'il se déplace vers la gauche, la condition préalable est l'appui sur la touche « » du clavier. Même chose avec la touche « → » pour le déplacement vers la droite. 10

Qu'il s'agisse des conditions ou les actions, elles sont toutes classées en 2 types : celles dédiées aux objets « **OBJECTS** » et les autres « **OTHER CONDITIONS / ACTIONS** ». Pour créer votre 1ère condition, cliquez sur « **add condition** », sélectionnez l'onglet « **OTHER CONDITIONS** » / « **Keyboard** » dans la liste, puis la condition choisie relative au clavier « **Key pressed** » qui correspond à l'évènement « **touche pressée** ». Indiquez ensuite le nom de la touche (« **Left** ») et validez via le bouton « **OK** ». Pour l'action associée, cliquez sur « **add action** », sélectionnez l'onglet « **OBJECTS** », l'objet dont vous modifiez l'animation (le sprite « **player** »), puis « **ANIMATIONS AND IMAGES** », et « **Change the animation (by name)** ». Renseignez le nom de l'animation « **moveLeft** ». Le nom de l'animation est le nom que vous lui avez donné lorsque vous les avez importées dans l'objet « **player** ». Pour l'animation du mouvement lors du déplacement vers la droite, cliquez sur l'icône 10 pour ajouter une nouvelle instruction et répétez les mêmes opérations.

Condition	Action
Left key is pressed	Set animation of **player** to « moveLeft »
Right key is pressed	Set animation of **player** to « moveRight »

Si vous testez la scène en l'état, vous constatez que les animations fonctionnent, mais que le sprite du joueur non seulement ne retourne pas à sa forme de départ, mais en plus qu'il continue de courir en position sur place. L'explication est facile : vous lancez les animations mais ne les stoppez pas. Pour corriger cela, vous devez donc simplement positionner l'animation du sprite sur « **idle** » lorsqu'aucune touche est utilisée. La condition « **Any key is pressed** » permet de tester si une touche (sans distinction) est pressée, la condition inverse répond à la condition qui nous intéresse. Même mode opératoire que précédemment en ajoutant une nouvelle instruction (icône 10) et en sélectionnant la condition « **Any key is pressed** » et en activant l'option « **Invert condition** » pour qu'elle soit inversée.

Condition	Action
Any key is pressed	Set animation of **player** to « idle »

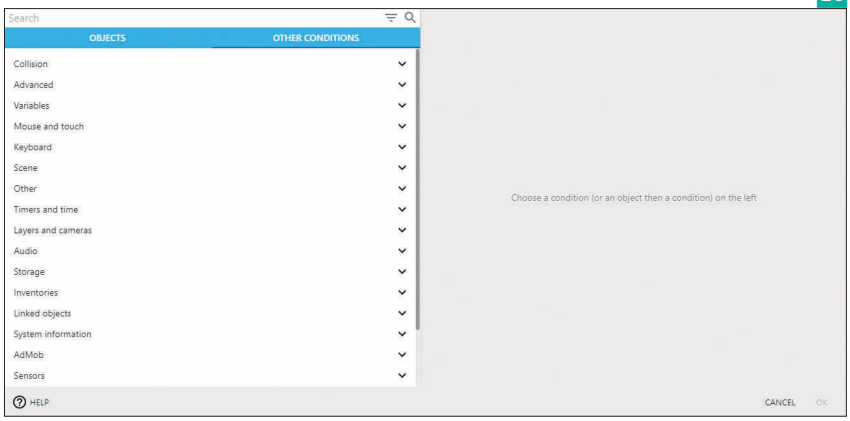
Les sauts du personnage joueur

Comme dit plus haut, l'appui sur la barre espace du clavier fait sauter le personnage, mais il est incapable de sauter sur une plateforme située au-dessus de lui. Ceci est dû aux caractéristiques du comportement « **PlaformerObject** » et son paramètre « **Gravity** » qui correspond, vous l'aurez bien compris, à la gravité trop élevée subie par votre avatar. En la passant à une valeur de 800, cela résout le problème du saut de plateforme en plateforme. Je vous laisse explorer les autres paramètres entre autres les vitesses de déplacement, de saut, ou de chute, ou encore l'accélération et décélération. Pour accéder aux paramètres du comportement, double-cliquez sur l'objet et sélectionnez l'onglet « **BEHAVIORS** ».

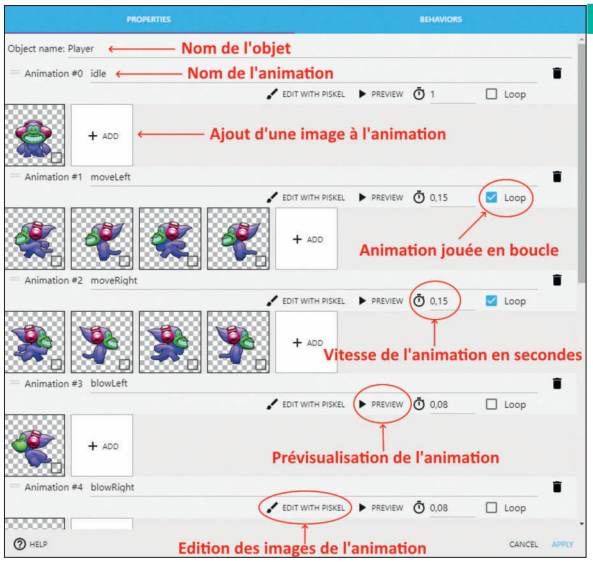
Pour ce qui est de l'animation du saut, les choses se compliquent un peu puisqu'il faut adapter l'animation du saut en fonction du déplacement du joueur : déplacement à gauche => image de saut à gauche et déplacement à droite => image de saut à droite. Incontournable est la création d'un indicateur du sens du personnage (gauche ou droite) pour ensuite positionner la bonne animation (**jumpLeft** ou **jumpRight**) lors de l'appui sur la barre espace. Cet indicateur prend la forme d'une variable d'objet que vous créez à partir du sous menu de l'objet « **Edit object variables** » et un simple clic sur le bouton « **+ ADD** ». 11 12

Nommez la « **direction** » sans lui donner de valeur. Dans l'éditeur de code (« **Level1 (Events)** »), vous positionnez sa valeur lors du déplacement du personnage (« **left** » lorsque c'est vers la gauche, « **right** » lorsque c'est vers la droite). Pour ce faire, cliquez sur « **Add action** », puis sous l'onglet « **OBJECT** », sélectionnez « **player** », « **VARIABLES** »/ « **Modify the text of a variable of an object** », choisissez la variable « **direction** », choisissez l'opérateur « **=** » puis indiquez la valeur à affecter « **left** » ou « **right** » selon le cas.

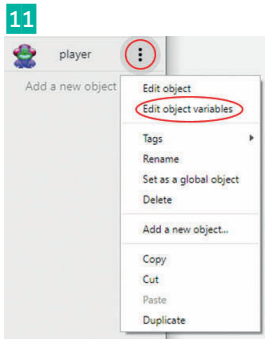
10



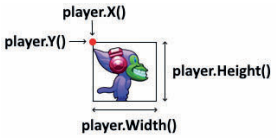
9



11



12



13

Condition	Action
Left key is pressed	Set animation of **player** to « moveLeft » Change the text of variable **direction** of player set to « **left** »
Right key is pressed	Set animation of **player** to « moveRight » Change the text of variable **direction** of player set to « **right** »

Utilisez cette même variable « **direction** » pour afficher la bonne animation de saut lors de l'appui sur la barre espace.

Condition	Action
Space key is pressed	
Sous-condition	
The text of variable **direction** of **player** = « **left** »	Set animation of **player** to « **jumpLeft** »
The text of variable **direction** of **player** = « **right** »	Set animation of **player** to « **jumpRight** »

Sous **GDevelop**, il existe 3 types de variables :
- **Globales** : accessibles et modifiables depuis toute scène du jeu et utilisées pour stocker des valeurs partagées comme par exemple un score ou encore le nombre de joueurs ;
- **Scènes** : accessibles et modifiables depuis la scène du jeu à laquelle elles sont rattachées comme le niveau d'énergie d'un boss de fin de niveau ;
- **Objets** : accessibles et modifiables depuis l'objet auquel elles sont rattachées, elles sont comparables à des propriétés d'objet tel l'état ouvert ou fermé d'un objet porte.

Notre personnage obéit au doigt et à l'œil mais un problème persiste, il peut sortir de l'écran. Pour résoudre ce détail, créez un objet « **Tiled sprite** » (nommez-le « **wall** ») auquel vous appliquez un comportement « **Platform** » sans lui associer d'images. Il suffit de glisser 2 fois cet objet sur la scène et de le mettre à même dimension que les colonnes des bords en prenant garde à le placer sur le calque « Base layer ».

Création, déplacement et animation de l'arme bulle (orb)

Attaquons-nous maintenant à l'arme du joueur et ses animations au nombre de 3 : le lancement (« **start** »), le déplacement horizontal (« **moveHorizontal** ») suivi du déplacement vertical (« **moveVertical** »). Les images à utiliser se situent dans le dossier **images/orb**. Opérez de la même manière que pour l'objet « **player** » pour créer le Sprite « **orb** » et ses animations sans lui appliquer de comportement.

Pour lancer une bulle, 3 questions : la touche du clavier utilisée, la position de départ de la bulle ainsi que le sens de sa course (vers la gauche ou la droite). Utilisez la touche « **Left Control** » (touche Ctrl à gauche de la barre espace) pour créer la condition avec le même mode opératoire que pour les touches directionnelles. A la différence que la condition est sur la touche relevée (released au lieu de pressed) pour garantir qu'une seule et unique bulle soit lancée à chaque appui. La position de départ de la bulle est quant à elle, déterminée par le sens du déplacement du joueur représentée par la variable « **direction** ». ». Tout comme pour la barre espace, deux sous-condition à créer.

Pour chacune des sous-conditions, la position de la bulle dépend de celle du joueur représentée par « **player.X()** » et « **player.Y()** » mais aussi de sa taille représentée par « **player.Width()** » et « **player.Height()** ». 13

A son démarrage et pour sa position horizontale, nous plaçons la bulle 10 pixels à gauche du joueur (**player.X()-10**) ou 10 pixels à droite du joueur (**player.X()+player.Width()+10**). Pour la position verticale, approximativement au niveau de la bouche (**player.Y()+player.Height()/3**). Pour le déplacement horizontal, vous appliquez une force dans un sens ou dans l'autre. Une valeur positive fait déplacer l'objet vers la droite, vers la gauche avec une valeur négative.

Créez une nouvelle instruction via l'icône 10, ajoutez la condition sur la touche « **Left Control** ». Pour l'action, créez l'objet à partir du code : sous l'onglet « **OBJECTS** », sélectionnez « **orb** », « **OBJECTS** » / « **Create an object** », renseigner sa position « **X position** » et « **Y position** » et enfin le calque dans lequel il est créé « **game** ». Pour donner le mouvement, il vous suffit de lui appliquer une force via l'onglet « **OBJECT** » / « **orb** » / « **MOVEMENT** » / « **Add a force** », indiquer son intensité sur les 2 axes X et Y en pixels par seconde et cocher l'option « **PERMANENT** ». Le déplacement se voulant horizontal, aucune force ne s'applique sur l'axe Y avec une valeur à 0.

Condition	Action
LControl key is released	
Sous-condition	
The text of variable **direction** of **player** = « **left** »	Create object **orb** at **position player.X()-10** ;**player.Y()+ player.Height()/3**Add to **orb** a **permanent** force of **-100** p/s on X axis and **0** p/s on Y axis
The text of variable **direction** of **player** = « **right** »	Create object **orb** at **position layer.X()+player.Width()+ 10**;**player.Y()+player.Height()/ 3**Add to **orb** a **permanent** force of **100** p/s on X axis and **0** p/s on Y axis

Pour que le lancer de bulle soit plus stylé, il est tout à fait possible de dérouler pour le joueur l'animation « **blowLeft** » ou « **blowRight** » selon la direction : lorsque vous appuyez sur la touche et la maintenez enfoncée, vous spécifiez l'animation qui prépare au lancer, lorsque la touche est relâchée, la bulle se lance.

Condition	Action
LControl key is pressed	
Sous-condition	
The text of variable **direction** of **player** = « **left** »	Set animation of **player** to « **blowLeft** »
The text of variable **direction** of **player** = « **right** »	Set animation of **player** to « **blowRight** »

En prenant un peu de recul avec tout ce qui a été écrit jusqu'à présent, un écueil saute aux yeux : les 2 tests sur la variable « **direction** », et à chaque fois sous forme de sous-conditions, apparais-

sent 3 fois. Un code qui se répète est toujours une mauvaise pratique qui nécessite de la réécriture optimisée de code (refactoring) : modifier ce test impliquerait 3 modifications aux 3 endroits où il se trouve alors qu'il n'en nécessiterait qu'une seule si le code était savamment réagencé.

Condition	Action
Any key is pressed	Set animation of **player** to « idle »
Left key is pressed	Set animation of **player** to « moveLeft » Change the text of variable **direction** of player set to « **left** »
Right key is pressed	Set animation of **player** to« moveRight » Change the text of variable **direction** of player set to « **right** »
The text of variable **direction** of **player** = « **left** »	
Sous-condition	
Space key is pressed	Set animation of **player** to « **jumpLeft** »
LControl key is released	Create object **orb** at **position** **layer.X()+10** ;**player.Y()+player.Height()/3** Add to **orb** a **permanent** force of **-100** p/s on X axis and **0** p/s on Y axis
LControl key is pressed	Set animation of **player** to « **blowLeft** »
The text of variable **direction** of **player** = « **right** »	
Sous-condition	
Space key is pressed	Set animation of **player** to « **jumpRight** »
LControl key is released	Create object **orb** at **position** **player.X()+player.Width()+10**;**player.Y()** **+player.Height()/3**Add to **orb** a **permanent** force of **100** p/s on X axis and **0** p/s on Y axis
LControl key is pressed	Set animation of **player** to « **blowRight** »

Lorsque la bulle se déplace, elle commence par un déplacement horizontal, s'ensuit un déplacement vertical dont il faut décider sur quel critère il s'enclenche : par exemple lorsqu'elle est à une certaine distance du joueur.

Nouvelle instruction : ajoutez une condition en sélectionnant dans l'onglet « **OBJECTS** », l'objet « **orb** » puis dans la liste des conditions « **POSITION** » / « **Distance between two objects** », indiquez le nom du 2nd objet (**player**) et renseignez la distance en pixels (**300**). « **Distance between two objects** » teste si la distance entre les 2 objets est inférieure à celle indiquée. Or le changement de direction ne doit se faire que si la distance est atteinte soit la condition inverse. Activez l'option « **Invert condition** » pour que la condition inverse soit appliquée. Vous n'avez plus qu'à stopper les forces qui étaient exercées jusqu'à présent « **OBJECTS** » / « **orb** » / « **MOVEMENT** » / « **Stop the object** » et à en appliquer une nouvelle de -100 pixels par seconde sur l'axe Y « **OBJECTS** » / « **orb** » / « **MOVEMENT** » / « **Add a force** ».

Condition	Action
orb distance to **player** is below **300** pixels	Stop **orb** (remove all forces) Add to **orb** a **permanent** force of **0** p/s on X axis and **-100** p/s on Y axis

Les personnages non joueurs (PNJ)

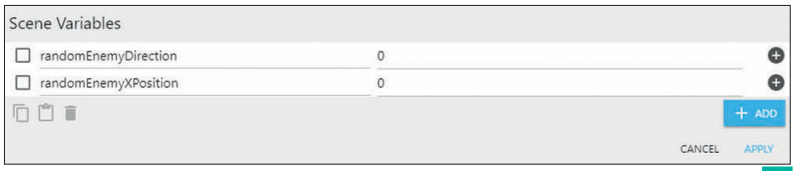
La 1ère chose à faire est d'ajouter un objet « **Sprite** » « **enemy** » avec les animations « **moveLeft** » et « **moveRight** » qui vont bien (sous **images/npc**) et lui adosser un comportement « **Platformer character** » en décochant la case « **Default controls** » sans quoi notre ennemi serait un PJ. La 2e chose est de créer l' « **enemy** » et le déplacer. 14

Le mouvement consiste à le faire chuter du haut du tableau, puis lorsqu'il rencontre une plateforme, le faire se déplacer sur la gauche ou la droite jusqu'à une prochaine chute. Une chute en bas de tableau le fait réapparaître en haut.

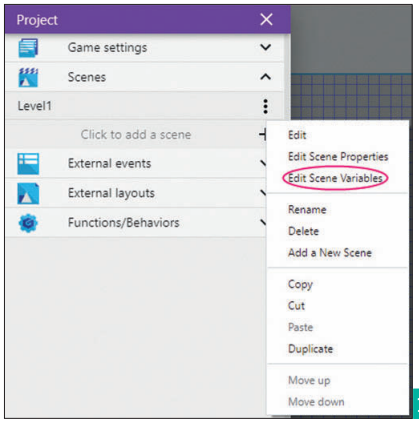
Il y a toute une phase d'initialisation du PNJ consistant à sa création et son positionnement aléatoire sur l'horizontale dans la scène. Pour tirer au sort cette position, est nécessaire la création d'une variable de scène appelée « **randomEnemyXPosition** » qui stocke la position de départ. Cliquez sur l'icône 0 pour afficher le menu projet, afficher le sous-menu de la scène « **Level1** » et sélectionnez « **Edit Scene Variables** ». Cliquez sur le bouton « **+ ADD** » pour ajouter une variable. 15 16

« **At the beginning of the scene** » est la condition dont toutes les actions sont appelées une seule fois au démarrage de la scène, c'est lors de cet événement que vous initialisez votre scène. Et c'est durant cette phase qu'une loterie donne la position de départ du

14



15



16

PNJ. Nouvelle instruction (icône 10) : avec la condition « **Add condition** » / « **OTHER CONDITIONS** » / « **Scene** » / « **At the beginning of the scene** », et l'action « **Add action** » / « **OTHER ACTIONS** » / « **Variables** » / « **Value of a scene variable** », choisissez la variable « **randomEnemyXPosition** » et l'opérateur « **=** ». La valeur est un appel à la fonction de tirage au sort : cliquez sur l'icône, « **Random** » / « **Random integer in range** » avec en minimale 300 et en maximale 500, cette fonction renvoie un nombre entier entre 300 et 500. Il n'y a plus qu'à utiliser « **randomEnemyXPosition** » pour positionner le PNJ sur l'horizontale, pour la verticale choisissez la négative de la taille du sprite « **-enemy.Height()** », permettant de le placer juste à l'extérieur du haut du tableau. Sans oublier de replacer « **enemy** » dans le bon calque « **game** ».

A ce stade, le PNJ est en place mais reste immobile. Pour le mouvoir vers la gauche ou la droite, il faut lui appliquer la force adéquate dont la valeur dépend aussi du sens de déplacement. Une nouvelle loterie pour choisir le sens de déplacement avec en réceptacle une autre variable de scène « **randomDirection** » à créer et appelant la fonction « **Random** » / « **Random integer** » avec 1 en valeur maximale pour avoir comme résultats possibles 0 ou 1 : 0 ou 1 pour aller respectivement à gauche ou droite.

A partir de la valeur de « **randomDirection** » et en sous-condition de « **At the beginning of the scene** », sont appliquées la force du mouvement uniquement sur l'horizontale et la bonne animation. Le mode opératoire est le même que pour la bulle.

Condition	Action
At the beginning of the scene	Change the scene variable **randomEnemyXPosition** := **RandomInRange(300,500)** Create object **enemy** at position **Variable(randomEnemyXPosition);-enemy.Height()** Change the scene variable **randomDirection** := **Random(1)**
Sous-condition	
The scene variable **randomDirection** = 0	Add to **enemy** a permanent force of **-100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveLeft**"
The scene variable **randomDirection** = 1	Add to **enemy** a permanent force of **100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveRight**"

Ce n'est pas terminé parce qu'en l'état, le PNJ va s'arrêter sur les murs latéraux. La détection de cette collision permet d'inverser sa direction sur la base de son mouvement en le stoppant et en appliquant la force opposée.

Condition	Action
enemy is collision with **wall**	Stop **enemy** (remove all forces)
Sous-condition	
The animation of **enemy** is **"moveLeft"**	Add to **enemy** a permanent force of **100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveRight**"
The animation of **enemy** is **"moveRight"**	Add to **enemy** a permanent force of **-100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveLeft**"

Vous avez ci-dessus 2 conditions qui se succèdent : si vous passez dans la 1ère condition vous passez aussi systématiquement dans la 2nde. Ce n'est pas le comportement souhaité et une simple structure du type « Si…Alors…Sinon » suffirait pour passer outre cette difficulté. **GDevelop** ne proposant pas cette structure, il vous faut le gérer vous-même par le biais d'un indicateur, une variable de scène qui indique si oui ou non il est nécessaire de faire le 2ème test. Appelez-le « **testCollisionEnemyWallRight** », initialisez-le à 1 au démarrage de la scène, ajoutez un test sur sa valeur à 1 dans la 2ème condition. Lorsque les 2 sous-conditions sont passées, replacez la valeur de « **testCollisionEnemyWallRight** » à 1.

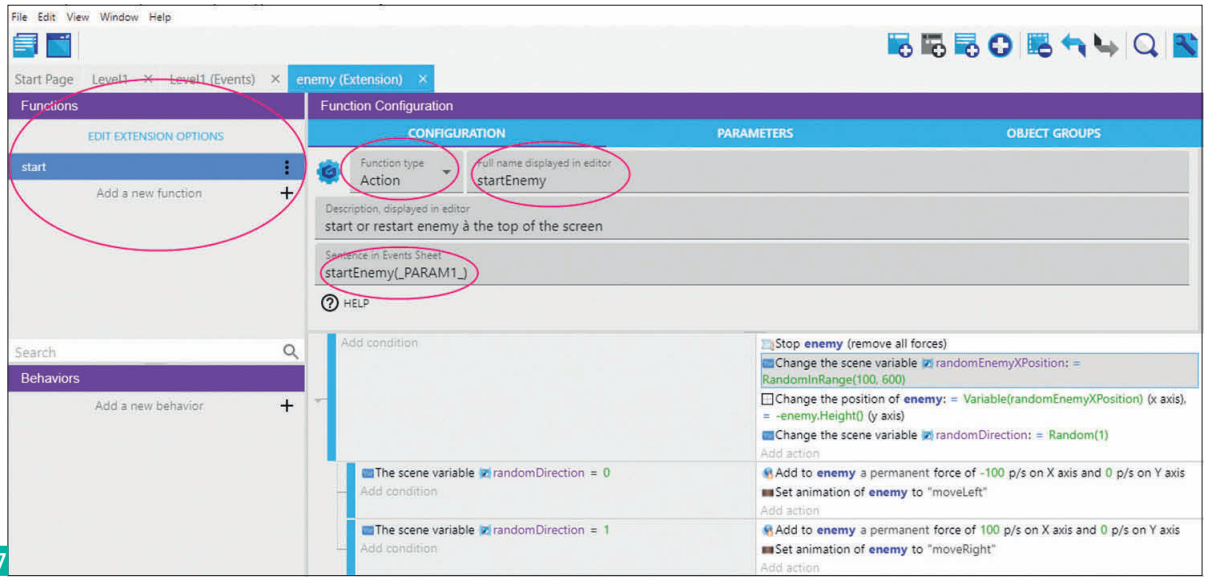
Condition	Action
enemy is collision with **wall**	Stop **enemy** (remove all forces)
Sous-condition	
The animation of **enemy** is **"moveLeft"**	Add to **enemy** a permanent force of **100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveRight**" Change the scene variable **testCollisionEnemyWallRight**=0
The animation of **enemy** is **"moveRight"** The scene variable **testCollisionEnemyWallRight**=1	Add to **enemy** a permanent force of **-100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveLeft**"
	Change the scene variable **testCollisionEnemyWallRight**=1

Désormais, le petit personnage se déplace de haut en bas en roulant sur les plateformes. Il n'y a plus qu'à faire en sorte qu'une fois être sorti par le bas de l'écran, il réapparaisse en haut. C'est juste un nouveau test sur la position verticale du PNJ : a-t-elle dépassé la taille de l'écran ? S'il est vrai, il replace le PNJ en haut de l'écran exactement comme cela est fait dans la phase d'initialisation.

Une nouvelle instruction avec pour condition « **OBJECTS** » / « **enemy** » / « **POSITION** » / « **Compare Y position of an object** », pour « **Sign of the test** » renseignez « **>** » et « **Value to compare** » renseignez « **ScreenHeight()** ».

Condition	Action
The Y position of **enemy** > **ScreenHeight()**	Stop **enemy** (remove all forces) Change the scene variable **randomEnemyXPosition** := **RandomInRange(100,700)** Change the position of **enemy** = **Variable(randomEnemyXPosition)** (x axis), = **-enemy.Height()** (y axis) Change the scene variable **randomDirection** := **Random(1)**
Sous-condition	
The scene variable **randomDirection** = 0	Add to **enemy** a permanent force of **-100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveLeft**"
The scene variable **randomDirection** = 1	Add to **enemy** a permanent force of **100** p/s on X axis and **0** p/s on Y axis Set animation of **enemy** to "**moveRight**"

C'est du code déjà vu, une répétition. Et afin de l'éviter, une bonne pratique est de l'encapsuler dans une fonction appelée en lieu et



17

place du code existant. Cliquez sur l'icône 0, sous le menu « **Functions/Behaviours** », cliquez sur « + » pour ajouter un groupe de fonctions, et renommez « **NewExtension** » en « **enemy** ». Pour ouvrir l'onglet d'édition, cliquez sur les 3 points en face du nom et sélectionnez « **Edit** » dans le sous-menu qui apparaît.

La création d'une nouvelle fonction passe par un clic sur « + » du cadre « **Functions** », choisissez « **Action** » et renommez « **NewFunctions** » en « **start** » qu'il suffit de sélectionner ensuite pour afficher l'onglet « **CONFIGURATION** » de l'éditeur. Renseignez « **Function type** » avec « **Action** », « **Full name displayed in editor** » avec « **startEnemy** », une description.

Cette fonction n'a d'utilité que pour l'objet « **enemy** », il va donc falloir lui passer l'« **enemy** » en paramètre. Cliquez sur l'onglet « **PARAMETERS** », cliquez sur le bouton « **+ ADD A PARAMETER** », nommez le paramètre « **enemy** », typez-le en « **Objects** » puis sélectionnez « **Sprite** » dans le champ « **Object type** ». En label, entrez « **enemy** ». Saisissez le code de la fonction.

Revenez sur l'onglet « **CONFIGURATION** » et renseignez « **Sentence in Events Sheet** » avec la forme sous laquelle l'appel de la fonction apparaît dans l'éditeur « **startEnemy(_PARAM1_)** ».

Ceci fait, dans l'éditeur de la scène, remplacez le code répété existant par l'appel de la fonction « **Add action** » / « **OTHER ACTIONS** » / « **enemy** » / « **startEnemy** ». Choisir en paramètre l'objet « **enemy** ». C'est beaucoup plus lisible.

Condition	Action
At the beginning of the scene	Create object **enemy** at position Change the scene variable **randomDirection** := **Random(1)** StartEnemy(**enemy**)

Condition	Action
The Y position of **enemy** > **ScreenHeight()**	StartEnemy(**enemy**)

Les collisions en PJ et PNJ

Durant cette collision, le joueur devient ange, et le robot continue sa course.

Nouvelle instruction : « **Add Condition** » / « **OBJECTS** », choisissez le 1er objet impliqué « **enemy** », ensuite sélectionnez « **COLLISION** » / « **collision** » et terminez en plaçant le 2ème objet impliqué « **player** ».

Condition	Action
enemy is in collision with **player**	Set animation of **player** to "angel"

Pour l'action, il y a juste à changer l'animation.

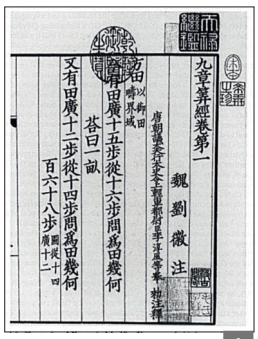
En conclusion

Beaucoup de choses essentielles ont été abordées, et volontairement j'ai introduit des termes un peu techniques tels que propriété, ou refactoring. L'assimilation va probablement vous demander un peu de pratique, mais l'essentiel est là. J'écouterai attentivement vos remarques et répondrai vos questions, soyez juste un peu patient pour la réponse.

Jean Michel Torres
Chef de Projets -IBM Quantum France
IBM Systems Center - Montpellier

Qu'est-ce qu'un algorithme ? Et pourquoi est-ce important ?

Profitez de l'été pour faire découvrir la programmation et la notion d'algorithme à vos (grands) enfants, au travers d'un peu d'histoire des mathématiques.



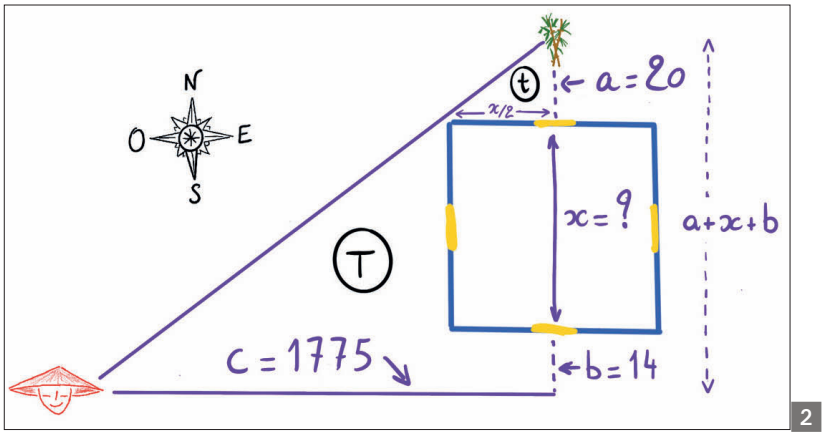
1

Cet article est inspiré par mon expérience de volontaire dans un certain nombre d'initiatives mises en place par mon employeur (voir en particulier ibm.org, #GoodTechIBM) visant à mener des actions auprès d'étudiants. Ceci dans le cadre des collèges et lycées, avec un réseau d'associations (entre-autres CGénial, YesWeCode) et souvent en partenariat avec d'autres entreprises. Le but est d'amener les élèves à épouser l'idée que les domaines scientifiques et techniques peuvent être passionnants, que les filles y ont leur place et que chacun y a sa chance.

Je partage donc ici un exemple d'initiation aux techniques que je produis auprès d'étudiants. Le début de l'exposé ci-dessous peut convenir à des élèves de fin de collège. Progressivement, les exemples vont plutôt concerner ceux de lycée, et la dernière partie est pour vous !

Lorsque l'on veut expliquer ce qu'est un algorithme, on utilise souvent cette comparaison : « c'est comme une recette de cuisine », une suite d'étapes à effectuer pour arriver à un résultat à partir d'ingrédients. Cette analogie convient pour un ordinateur, un algorithme permet de lui indiquer quoi faire avec quelles données pour arriver au résultat désiré. Et comme pour une recette de cuisine on utilise un langage spécialisé, que l'on appelle langage de programmation, par exemple : « battre les blancs en neige » ne veut rien dire hors d'une cuisine.

Les recettes sont plus ou moins précises, complexes, efficaces… Il en va de même pour les algorithmes en informatique. Voyons quelques exemples mais tout d'abord un peu d'Histoire…



2

Premiers pas et balade dans l'Histoire

Remontons aux origines … bien avant les ordinateurs, les mathématiques ont petit à petit construit les solutions aux problèmes qui se posaient à nos ancêtres et des « livres de recettes » sont apparus. L'un des plus célèbres nous vient de Chine et date d'environ deux mille ans : « L'art mathématique en neuf chapitres ». Il réunit près de 250 procédures de calcul utiles dans les domaines de l'agriculture, du calcul des surfaces, du commerce, de la fiscalité, ainsi que de la résolution d'équations ou de l'étude des triangles. 1

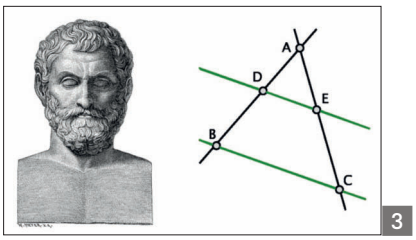
On y trouve par exemple le problème suivant : **Soit une ville carrée, avec une porte située au milieu de chacun de ses cotés. A 20 *bù* au-delà de la porte Nord, se trouve un arbre. Si l'on sort par la porte Sud, après avoir marché 14 *bù* vers le Sud, on se dirige vers l'Ouest après avoir parcouru 1775 *bù*, on aperçoit l'arbre. On demande quelle est la taille de la ville.**

(note : un *bù* (步) vaut actuellement 5 pieds chinois *chǐ* (尺) de 32cm, soit 1,6m)

Le livre contient la solution, exprimée dans le même style que l'énoncé : pas de formule, pas de variable, aucune équation, difficile à programmer…

Voici ce que donne une représentation schématique du problème de la ville carrée : 2

Il reste à utiliser un des théorèmes les plus célèbres des mathématiques, il nous vient



3

des écoles de géométrie de la Grèce Antique : le théorème de Thalès (Thalès de Milet, environ 600 avant J.C.). 3

Le « petit » triangle t est semblable au « grand » triangle T, et donc d'après le théorème de Thalès : $\frac{a}{x/2} = \frac{a+x+b}{c}$, que l'on peut réécrire : $x^2 + (a+b) = 2ac$ (avec a=20, b=14 et c=1775).

Vous reconnaîtrez sans doute une équation du second degré, mais cette notion n'est pas encore formalisée à ces époques.

Attendons encore quelques siècles et la naissance de l'Algèbre… Intéressons-nous à Abou Abdallah Muhammad Ben Musa 'al Khuwârizmî, savant originaire de la région de Khwârizm, actuelle Khiva située en Ouzbékistan. Il rédige, entre 813 et 833, « Hisâb al-jabr wa'l-muqqâbala » (Science de la transposition et de la réduction) d'où l'on tire « Algèbre » (al-jabr) qui désigne la partie des mathématiques s'intéressant entre autres à la science des équations. 4

Il n'y a toujours pas les notations telles que nous les connaissons aujourd'hui mais Al-Khuwârizmî propose un regroupement

en catégories des différents types d'équations du premier et second degré, ainsi que des procédés de résolutions et des démonstrations : des recettes dont on sait qu'elles fonctionnent avec des ingrédients différents. Le nom d'Al Khuwârizmî, a produit le mot « algorithme » par déformations successives (et donc ce mot ne s'écrit pas avec un « y »).

On lui doit aussi l'ouvrage « Traité du système de numération des Indiens » qui a contribué à la diffusion des « chiffres arabes » en Orient et en Europe.

Voici la méthode qu'Al-Khuwârismî nous propose pour résoudre l'équation correspondant au problème de la ville carrée.

Il place cette équation dans la catégorie : « des biens et des racines égalent un nombre » (avec les notations modernes cela s'écrirait : $X + a\sqrt{X} = b$, où a est la quantité des racines ou simplement « les racines », b est « le nombre », X est « les biens » c'est-à-dire la quantité recherchée).

Il formule un exemple de la façon suivante : « un capital et dix Dirhams sont multipliés par son tiers, alors on obtient treize » (ce qui s'écrirait maintenant : $\frac{x}{3}(x+10) = 13$ ou encore en multipliant des deux côtés par 3 et en développant la partie à gauche du signe « = » on a l'équation suivante : $x^2 + 10x = 39$, qui a cette formulation équivalente $X + 10\sqrt{X} = 39$, alors « les racines » valent 10, et « le nombre » est 39.

Voici maintenant la recette :

Prends la moitié des racines (5)
Multiplie ce résultat par lui-même (25)
Ajoute le résultat au nombre (25+39 = 64)
Prends la racine de ce résultat (8)
Soustrais-en la moitié des racines (8 - 5 = 3)

Nous y sommes ! Nous sommes prêts pour trouver la taille de la ville carrée chinoise, rappelons notre équation : $x^2 + (a+b)x$ (avec a=20, b=14 et c=1775).

Nous savons maintenant que pour la terminologie d'Al Khuwârizmî, les « racines » valent 20+14 et le « nombre » vaut 2×1775×20.

Avec le langage de programmation Python, cela donne : **Code 1 villecaree.py**

Je vous propose une variante de ce problème à résoudre vous-même : pour une autre ville carrée, un arbre se trouve à 30 *bù* de la porte nord. En sortant par la porte ouest, après avoir marché 750 *bù* on aperçoit cet arbre. On demande également la taille de la ville carrée.

Plus compliqué : le calcul des décimales du nombre π

Alors avançons encore dans le temps : après la Renaissance, les mathématiques se sont largement enrichies et l'on découvre par exemple des manières de calculer de plus en plus précisément la valeur du nombre π (3,141592…)

Une variante éloignée de la méthode de l'aiguille de Buffon (1733) est la méthode de Monte-Carlo dont le nom fait référence aux célèbres Casinos de Monaco, car elle s'appuie sur le hasard.

Par exemple si l'on veut mesurer la surface d'une forme pour laquelle il n'y a pas de formule mathématique (disons un lac), si l'on sait encadrer cette surface dans un rectangle dont on connaît la surface, et si l'on choisit des points au hasard dans le rectangle, alors l'aire de la surface cherchée correspond au nombre de points à l'intérieur de la surface divisé par le nombre total de points tirés au hasard et multiplié par la surface du rectangle. 5

Voyons ce que cela donne pour un quart de cercle de rayon 1 dans un carré de côté 1 : 6

Code 2.1 monteCarlo.py

Ici, il a fallu 10 millions de points et plus de 4 secondes pour un résultat avec seulement 3 décimales correctes. Cette méthode est très largement utilisée dans de nombreux cas mais elle est très consommatrice en puissance et en durée de calcul.

A nouveau, des mathématiciens nous viennent en aide, voyons deux formules.

La formule de John Wallis (1616 – 1703). 7

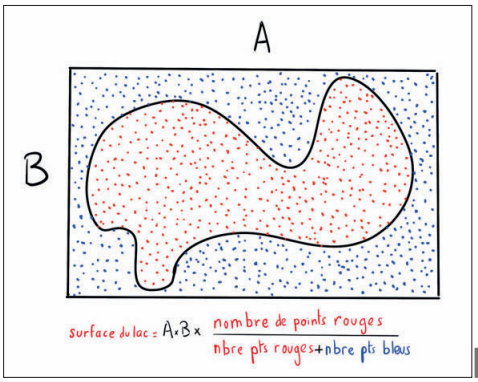
Le signe et un « pi » majuscule et signifie qu'on fait la multiplication des termes qui sont à sa droite.

Il n'est pas possible de calculer un nombre infini de termes, il s'agit donc de calculer un produit de N termes, c'est assez simple :

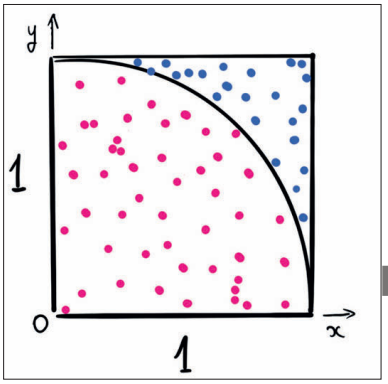
Code 2.2 wallis.py

Le calcule s'approche de la valeur de π avec une meilleure efficacité qu'avec la méthode de Monte-Carlo, ici avec un million de termes nous avons obtenu 5 décimales correctes. Si l'on augmente le nombre de termes à calculer, on voit toutefois que nous sommes limités dans la précision que nous pouvons atteindre, du fait du format des nombres à virgule que manipule Python. Il y a une formule antérieure mais qui donne des résultats encore meilleurs


4



5



6

$$\frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{4n^2}{4n^2 - 1}$$

7

$$\pi = 2 \times \frac{2}{\sqrt{2}} \times \frac{2}{\sqrt{2+\sqrt{2}}} \times \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}} \times \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}} \times \cdots$$

8

c'est la formule de François Viète (1540-1603) : 8

Pour le calcul, remarquons simplement qu'au dénominateur de chaque terme, on retrouve le dénominateur du terme précédent auquel on ajoute 2 et dont on prend à nouveau la racine carrée.

Ainsi on peut écrire l'algorithme voulu en « pseudo-code » : 9

Le code en Python est alors assez simple :

Code 2.3 viete.py

Vous pouvez faire différents essais et voir que la formule de Viète semble plus efficace pour calculer π en un plus petit nombre d'itérations, on dit qu'elle « converge plus rapidement ». Il y a cependant une nuance : la formule de Wallis a l'avantage de ne pas utiliser la fonction d'extraction de racine carrée qui est assez coûteuse en temps de calcul si l'on souhaite une grande précision. De ce point de vue l'algorithme utilisant la formule de Wallis présente un avantage. Rappelons-nous aussi qu'au XVIIème siècle les calculs ne pouvaient se faire qu'à la main. Il existe d'autres formules et surtout d'autres méthodes qui permettent de calculer les décimales de π : le record actuel, obtenu en mars 2019 dépasse 31 000 milliards de décimales.

```
Entrez le nombre de termes à calculer
Initialisation du produit à la valeur 2
Initialisation du denominateur à sqrt(2)
Pour chaque terme :
      produit = produit * 2 / denominateur
      denominateur = racine (2 + denominateur)
Fin Pour
Affichage du produit
```

9

L'efficacité d'un algorithme

Revenons sur la question de la performance pour mieux l'illustrer avec un algorithme aussi simple que classique : le tri d'une liste de nombres.

Le plus simple et le premier algorithme de tri que l'on programme s'appelle le tri à bulles. Il consiste à parcourir la liste à trier, d'un bout à l'autre et d'échanger la place de deux éléments voisins si le second est plus petit que le premier. Ainsi les éléments les plus petits (disons légers) vont petit à petit « remonter » vers le « haut » de la liste, d'où le nom de tri à bulles, comme des bulles qui remontent dans un liquide. Il faut recommencer cette étape tant que la liste n'est pas complètement triée.

Le codage est assez simple et en regardant bien on voit que si N est la taille de la liste à trier alors chaque passage utilise une boucle de N itérations, et qu'il faut faire au maximum N passages. Au total il faudra donc exécuter N*N boucles, on dit que l'algorithme est d'ordre « N au carré », on peut donc s'attendre à ce que le temps d'exécution augmente comme le carré de la longueur de la liste à trier.

Code 3.1 TriABulles.py

Pour étudier le comportement de cet algorithme lorsque la longueur de la liste augmente, vous pouvez utiliser ce code (en ordonnées du graphique, on voit le temps d'exécution du programme en secondes, en abscisses la longueur de la liste à trier) : 10

Vous pouvez facilement modifier le programme pour en améliorer un peu l'efficacité en arrêtant le tri dès qu'une passe n'a pas produit de swap (plutôt que de parcourir systématiquement la liste à N reprises).

Enfin, comparons avec un algorithme de tri nettement plus efficace, le « quick-sort » ou tri rapide. Il s'agit d'un algorithme récursif (c'est-à-dire qu'il contient une fonction qui s'appelle elle-même). Cet algorithme a été décrit par Tony Hoare en 1961 :

- La fonction récursive de tri doit être construite pour s'appliquer à une partie de la liste à trier.
- Cette fonction prend comme entrée une partie de la liste à trier (partie définie par la position de début et la position de fin), le premier appel à cette fonction se fait sur la liste entière.
- Tant que les deux positions (début et fin de la sous liste) sont différentes, la fonction calcule une « position pivot » et s'appelle elle-même deux fois : une fois pour la partie de la liste « à gauche » de la position pivot et l'autre pour la partie « à droite ».
- Lorsque les deux positions sont égales, le tri est terminé.
- Pour calculer la position pivot d'une liste : on initialise un index à 0, et on parcourt la liste (avec un indice) du début à la fin. A chaque fois que la valeur de la position courante est inférieure à la valeur de la dernière position, on met cette valeur à la

In [ ]:

```
%matplotlib inline
import random
import time
import matplotlib.pyplot as plt

def get_bubble_sort_time(n):
    arr = [random.random() for i in range(n)]
    start_time = time.time()
    for i in range(n):
        for j in range(n-1):
            if arr[j+1] < arr[j]:
                arr[j+1],arr[j] = arr[j],arr[j+1]
    end_time = time.time()
    return  end_time - start_time

times = []
sizes = [500,1000,1500,2000,2500,3000,3500,4000,4500]
for n in sizes:
    times.append(get_bubble_sort_time(n))

plt.plot(sizes,times,color="blue")
```

10

In [ ]:

```
%matplotlib inline
import random, time
import matplotlib.pyplot as plt

def quicksort(arr, start, end):
    if start >= end:
        return
    index = partition(arr, start, end)
    quicksort(arr,start,index - 1)
    quicksort(arr,index + 1, end)

def partition(arr,s,e):
    pivotindex = s
    pivotvalue = arr[e]
    for i in range (s,e):
        if arr[i] < pivotvalue:
            arr[i], arr[pivotindex] = arr[pivotindex], arr[i]
            pivotindex +=1
    arr[pivotindex], arr[e] = arr[e], arr[pivotindex]
    return pivotindex

def get_quicksort_time(n):
    arr = [random.random() for i in range(n)]
    start_time = time.time()
    quicksort(arr,0,n-1)
    end_time = time.time()
    return  end_time - start_time

times = []
sizes = [50000,100000,150000,200000,250000,300000,350000,400000]
for n in sizes:
    times.append(get_quicksort_time(n))

print(times)
plt.plot(sizes,times,color="blue")
```

11

position de la liste correspondant à la valeur de l'index (et on incrémente alors cet index). A la fin on met la valeur de l'élément de fin de liste à cette position, on a donc à sa gauche tous les éléments de valeur inférieure et à droite tous les éléments de valeur supérieure. On recommence pour ces deux sous listes (gauche et droite) ainsi définies.

Code 3.2 qicksort.py

Comme il y a cette notion de couper la liste en deux à chaque passe d'appels à la fonction récursive, la durée d'exécution a quelque chose à voir avec le logarithme de la longueur de la liste (autrement dit : lorsque l'on double, il faut ajouter une passe). En fait l'algorithme s'exécute en un temps proportionnel à $N \times logN$ ce qui est beaucoup plus petit que N^2 lorsque N grandit.

Vérifiez par vous-même : on peut maintenant trier des centaines de milliers d'éléments plutôt que des milliers d'éléments avec le tri à bulles, dans un temps comparable. Ce dernier code construit la courbe représentant la durée d'exécution en fonction de la taille de la liste : 11

Complexité algorithmique

Nous venons de voir quelques exemples d'algorithmes plus ou moins « efficaces » : ceci se définit et se généralise avec la notion de complexité algorithmique. Une manière de classer les algorithmes est d'évaluer leur « ordre », qui représente le nombre d'étapes à accomplir pour résoudre un problème donné en fonction de sa « taille ». Pour reprendre l'exemple du tri, on peut dire que le tri à bulles est d'ordre n^2 car le nombre d'opérations à effectuer varie comme le carré de la taille de la liste à trier. Et on a vu que pour le tri rapide, l'ordre devient $n*log(n)$. Ces deux fonctions sont croissantes mais de manières très différentes lorsque n devient grand. Par exemple, si n vaut 1 000, alors n^2 vaut 1 000 000, alors que $n*log(n)$ vaut environ 7 000.

On dit aussi que le tri à bulles est de classe polynomiale (car son ordre est une puissance de sa taille n). Cette classe, appelée P, regroupe l'ensemble des problèmes pour lesquels on connaît un algorithme d'ordre polynomial pour les traiter (n^2, n^3, n^4, etc). Ces problèmes peuvent nécessiter un temps considérable, voire même trop important pour être résolus dans certains cas, mais cela reste modeste par rapport à la classe des problèmes dont le temps de résolution est exponentiel.

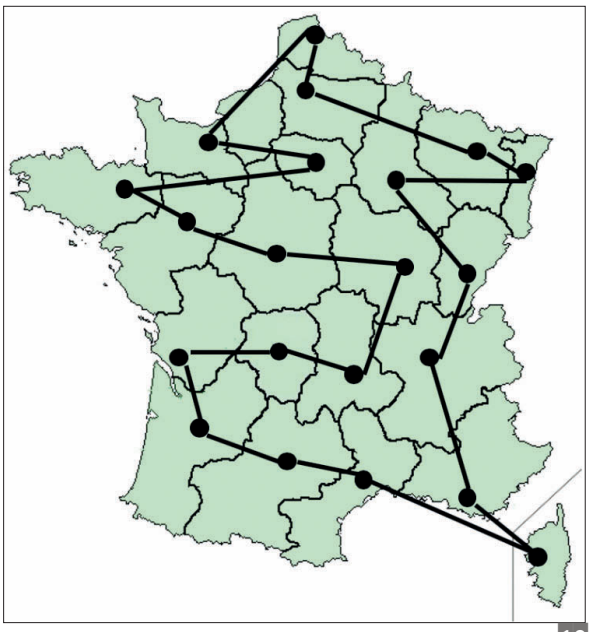
Dans une période plus récente de l'Histoire des mathématiques, en 1900, le mathématicien David Hilbert a listé 23 problèmes non résolus qui présentaient une importance significative pour les mathématiques. A ce jour, 5 d'entre eux résistent toujours. Encore plus récemment, en 2000, l'institut Clay de Mathématiques a présenté 7 problèmes tout aussi difficiles et importants, connus comme « les 7 problèmes du millénaire ». Pour chacun d'eux, une prime d'un million de dollars américains récompensera celui ou celle qui le résoudra. L'un d'entre eux concerne les algorithmes de classe P.

Une autre classe, appelée NP contient des problèmes pour lesquels on ne connaît pas d'algorithme de classe polynomiale (on connaît souvent un algorithme de classe exponentielle), mais pour lesquels on peut vérifier en un temps polynomial si une solution est correcte ou pas. Par exemple, résoudre une grille de Sudoku à n lignes et n colonnes est un problème d'ordre « exponentiel », mais vérifier si une grille complétée est juste ou pas est un problème d'ordre polynomial. C'est bien un problème appartenant à la classe NP. Au contraire, trouver une position optimale pour un jeu d'échecs, ou vérifier qu'elle l'est, sont deux problèmes aussi difficiles l'un que l'autre.

Le problème à un million de dollars consiste à déterminer si l'ensemble des problèmes de classe NP est égal (ou pas) à l'ensemble des problèmes de classe P.

Un indice si vous vouliez vous lancer : on connaît une liste de problèmes appelés « NP-difficiles », et on a déjà démontré qu'il suffit de trouver un algorithme d'ordre polynomial qui résout un seul des problèmes de cette liste pour conclure que P = NP. 12

Venons-en maintenant à un exemple célèbre de problème NP-difficile : le voyageur de commerce. L'énoncé est simple : étant donné n villes, quel est le plus court chemin qui les relie toutes une seule fois ? Les applications pratiques sont nombreuses : tournée de livraison, itinéraire d'avion, déplacement de machine outil. Il existe des algorithmes dits « heuristiques » qui permettent d'obtenir de « bons résultats » en temps « raisonnable », mais les seuls algorithmes connus permettant de garantir que l'on a trouvé la solution optimale sont d'ordre exponentiels (pour n villes : chaque chemin possible consiste à



12

choisir la première ville parmi n villes, la seconde parmi $n-1$, la troisième parmi $n-2$, et ainsi de suite), le nombre de chemins à comparer est donc de l'ordre de $n!$ (fonction factorielle). Pour seulement 21 villes comme sur la carte de France ci-dessus, le nombre de chemins à comparer est supérieur à 51 milliards de milliards. Si l'on peut tester un milliard de chemins par seconde sur un processeur, et que l'on dispose de 25 000 processeurs, il faudra près de 65 000 années pour obtenir le résultat !

Voici un code en Python pour traiter ce problème. Attention, au-delà de 9 villes, c'est très long. Évidemment, il y a d'autres stratégies que la « force brute », par exemple avec les algorithmes génétiques ou les algorithmes de colonies de fourmis. On parle aussi de l'informatique quantique comme solution possible à ce genre de difficulté : **Code 4 voyageur.py**

Conclusion

J'espère que cette « ballade algorithmique au travers de l'Histoire des mathématiques » vous a permis d'apprendre et surtout vous a donné envie d'en savoir plus. J'espère aussi vous avoir convaincu qu'avoir un bon algorithme c'est important, mais ce n'est généralement pas facile à inventer. Vous pourrez retrouver les codes ici pour éviter d'avoir à les écrire : https://github.com/jmit34/HS-ETE-2020 . Assurez-vous d'avoir chargé Python 3 et la bibliothèque `matplotlib` pour les graphiques.

François Mocq
Electronicien, radioamateur (F1GYT), Technicien de maintenance puis formateur d'adultes en maintenance informatique, réseaux et télécom. Créateur du blog **www.framboise314.fr** *et auteur des livres "Raspberry Pi 4" et "Scratch et Raspberry Pi" parus aux Editions ENI*

Passerelle et nœud LoRaWan

L'IoT utilise différents systèmes de communication radio. Parmi ceux-ci, LoRaWan est un protocole à très bas débit et faible consommation. La passerelle LoRa peut être utilisée en autonome (site industriel, milieu agricole, ville intelligente, campus…) mais elle peut aussi se connecter à Internet. Les données recueillies par les nœuds peuvent alors être partagées, mises à disposition en OpenData.

L'origine du projet

J'ai un atelier situé à plus d'un kilomètre et je souhaite avoir régulièrement des informations sur la température, l'humidité et une éventuelle intrusion. Il serait bien également de pouvoir monter la température avant de m'y rendre quand les températures sont basses.

Le pont WiFi était une solution mais pour avoir quelque chose de fiable, le prix piquait un peu car il fallait partir sur une solution ayant une portée de 3 Km. L'autre solution, c'était de mettre en place une passerelle LoRa. Les anglais de PiSupply [1] proposent tout un ensemble de matériels adaptés au Raspberry Pi mais aussi à Arduino ou micro:bit. J'ai arrêté mon choix sur une gateway [2] (passerelle) pour le Raspberry Pi et une carte node [3] (nœud) pour le Raspberry Pi Zero.

Pour pouvoir consulter et piloter à distance, j'ai opté pour The Things Network [4] (TTN) qui est un réseau ouvert permettant de construire un réseau pour l'IoT à moindre coût. Ce réseau collaboratif regroupe des milliers de passerelles réparties sur le monde et utilisées par des millions de personnes. Le chiffrement de bout en bout permet de faire transiter les données par ces passerelles en sécurité.

Le matériel utilisé

Pour la passerelle

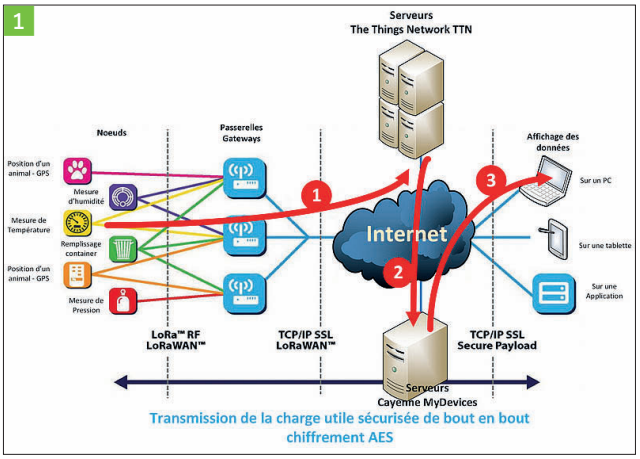
- Carte passerelle LoRa Gateway HAT pour le Raspberry Pi (170€)
- Raspberry Pi 4 – 2Go RAM (39€)
- Carte SD 16Go class 10 (10€)
- Alimentation 5v/3A (8€)
- Boîtier (6€)

Pour le noeud

- Carte LoRa Node pHAT pour le Raspberry Pi (40€)
- Rapsberry Pi Zero WH (15€)
- Carte SD 16Go class 10 (10€)
- Alimentation 5v/3A (8€)

Principe de la réalisation

L'image 1 représente le synoptique de cette réalisation. A gauche le **Nœud LoRa** envoie les valeurs mesurées par radio vers la passerelle. Il n'y a qu'un nœud dans ce projet, mais plusieurs nœuds pourront envoyer tout un tas de données. La passerelle est chargée de recevoir les infos et de les relayer vers le serveur **The Things Network** (TTN). Ici il n'y aura qu'une passerelle. Lorsque plusieurs passerelles sont accessibles et qu'un nœud envoie les infos vers



plusieurs passerelles, le serveur se charge de supprimer les doublons, et ne garde que la trame provenant de la gateway qui reçoit le mieux le signal du nœud. Ce sera aussi sur cette passerelle qu'un éventuel envoi de données vers le nœud sera effectué. Dans mon cas j'ai un seul nœud et une seule passerelle, ce qui simplifie le schéma.

Une fois que les données sont arrivées sur le serveur TTN elles sont stockées pendant un temps limité. Il faut donc les renvoyer vers une base de données (BDD) pour les enregistrer de façon pérenne. Vous pouvez utiliser votre propre base de données. J'ai choisi la solution toute faite de MyDevices Cayenne dans sa version gratuite. Il suffit de créer un compte pour bénéficier des services de base de la plateforme. On pourra créer sur MyDevices un tableau de bord (DashBoard) qui sera accessible via internet sur tout périphérique (PC, tablette, smartphone…).

LoRa et LoRaWan

LoRa est l'acronyme de Long Range (Grande portée). C'est une technologie radio (sans fil) dans laquelle un émetteur à très faible consommation envoie de petits paquets de données à faible vitesse (0.3kb/s à 5,5kb/s) à un récepteur situé à grande distance.

LoRaWAN est un protocole de télécommunication permettant la communication à bas débit, par radio. Il est utilisé pour des objets à faible consommation électrique communiquant selon la technologie LoRa et connectés à l'Internet via des passerelles, participant ainsi à l'Internet des objets (IoT). (source Wikipedia)

L'image 1 montre l'architecture d'un réseau LoRaWAN. A gauche les "nœuds" collectent les données. Ils sont capables de les envoyer en consommant une énergie minime, grâce à une transmission à

bas débit. La portée est très variable mais on atteint couramment plusieurs kilomètres. La carte LoRa Node pHAT de Pi Supply est annoncée pour 15Km environ. Dans de (très) bonnes conditions, un signal LoRa a été reçu à 766Km, l'émetteur avait une puissance de 25mW. C'est actuellement le record du monde [13].

Les Gateways sont chargées de collecter les données des nœuds et de les transférer à un serveur qui pourra les traiter, les redistribuer… Il existe des solutions propriétaires, pour ma part j'ai choisi d'utiliser The Things Network (TTN) un réseau international gratuit, open source et communautaire. Les serveurs de TTN vous permettent de recevoir dans vos applications les données de vos appareils, même à des milliers de kilomètres de chez vous !

Actuellement (janvier 2020) le réseau se compose de plus de 40000 contributeurs regroupés en plus de 400 communautés dans 90 pays ayant déployé plus de 4000 passerelles. Il est possible pour les particuliers, universités, entreprises, ou encore les communes de contribuer au déploiement ou d'utiliser gratuitement The Things Network.

En Europe, la bande réservée à LoRaWan est la bande 868MHz qui est une bande ISM *(industriel, scientifique, et médical)* utilisable sans autorisation préalable des autorités. La puissance d'émission et le taux d'utilisation temporel sont limités par la réglementation. Pour les autres régions du monde, référez-vous à la réglementation locale.

La modulation LoRa utilise un étalement de spectre appelé **Chirp Spread Spectrum** pour coder l'information. SigFox utilise l'Ultra Narrow Band. Un chirp ou gazouillis est un signal sinusoïdal dont la fréquence varie au cours du temps. Initialement destinée à des applications militaires, cette modulation est également utilisée par les radars et les sonars. Elle est robuste contre différents types de perturbations.

L'image 2 donne une indication de la portée du signal en fonction du facteur d'étalement (SF). La portée réelle dépend des conditions locales (immeubles, montagne, vallée…). Si l'étalement augmente, la portée augmente, mais le débit se réduit et la durée de transmission augmente, réduisant l'autonomie du dispositif. Il faudra adapter ces facteurs ainsi que la puissance pour obtenir la portée souhaitée.

On confond souvent LoRa et LoRaWAN, ce sont deux couches différentes du système :

LoRa correspond à la couche physique (radio et modulation) ;

LoRaWAN gère les couches MAC (Media Access Control) et Application.

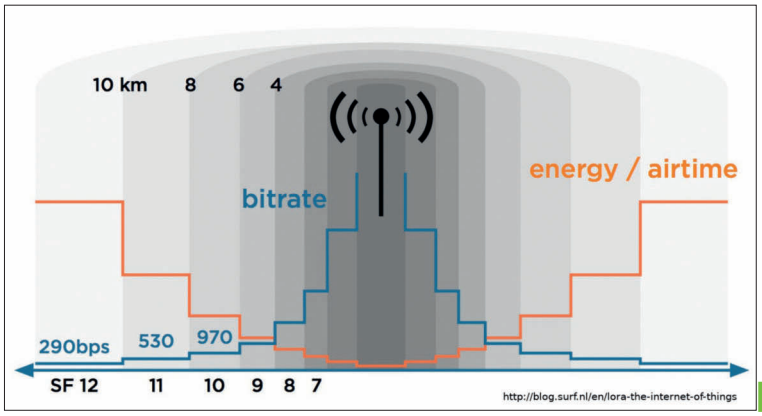
The Things Network

The Things Network (TTN) est un réseau LoRaWAN communautaire et open source conçu pour l'Internet des Objets (IoT en anglais). Il assure un chiffrement robuste de bout en bout. Il faudra créer un compte avant de commencer à utiliser le réseau. TTN envoie un mail de confirmation pour valider l'inscription.

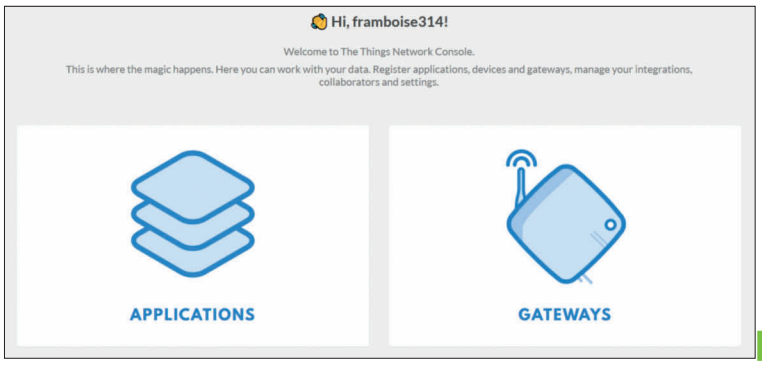
Inscription

Avant de démarrer la passerelle, il faut créer le compte TTN pour récupérer la clé de la passerelle. Sur la page d'accueil cliquez sur **Sign In** et créez un compte. 3

Cliquez sur l'icône **Gateways** pour déclarer votre passerelle. Dans



2



3

la fenêtre qui s'ouvre, vous allez remplir les champs suivants :

- **Gateway ID/ID de la passerelle** : il doit s'agir d'un identifiant unique, qui peut être n'importe quoi, mais qui doit être unique.
- **Description** : texte libre, c'est pour que vous ayez une description de la passerelle.
- **Frequency plan/Plan de fréquences** : il s'agit de la fréquence de la passerelle qui doit correspondre à la région. Comme nous sommes en Europe, j'ai choisi le plan **Europe 868Mhz**.
- **Router** : c'est le serveur sur lequel la passerelle va se connecter. Il devrait être le plus proche de votre lieu de résidence. Comme nous sommes en Europe, j'ai sélectionné **ttn-router-eu**, mais vous trouverez peut-être un routeur plus proche de vous.
- **Location/Emplacement** : cette indication n'est pas obligatoire mais indiquer l'emplacement de la passerelle peut être utile à l'avenir pour la cartographie de la couverture du signal LoRa. Cliquez à l'emplacement de la passerelle, sur la carte.
- **Antenna placement/Placement de l'antenne** : indiquer si la passerelle et l'antenne sont à l'intérieur ou à l'extérieur. Pour moi elle est actuellement à l'intérieur, ce qui réduit la zone de couverture.

Vous pourrez revenir sur tous ces paramètres, excepté l'ID de la passerelle qui ne peut être modifié. Une fois ces informations validées, une fenêtre **Gateway Overview** vous donne un résumé des paramètres. La clé **Gateway Key** est visible en cliquant sur l'œil à gauche de la zone. Vous pourrez la copier en cliquant sur l'icône de la feuille à droite de la zone. Cela vous servira un peu plus tard car il faudra coller cette clé lors de la configuration de la passerelle sur le Raspberry Pi.

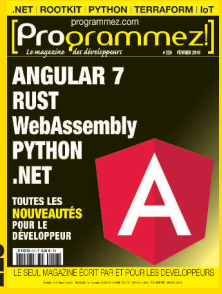
Il reste à modifier quelques paramètres. Cliquez sur l'onglet "**Settings**" et vous devriez pouvoir voir les paramètres que nous avons configurés, puis sur l'onglet "**Informations**" à gauche.

nouveau !

Boutique Programmez!

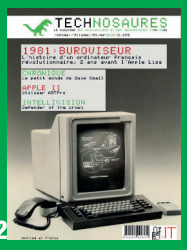
Les anciens numéros de PROGRAMMEZ!

Le magazine des développeurs



Technosaures

Le magazine à remonter le temps !



Prix unitaire : **7,66 €**
(frais postaux inclus)

Histoire de la micro-informatique 1973-2007

12,99 €
(frais postaux inclus)

☐ 219	: __ ex	☐ 235	: __ ex
☐ 220	: __ ex	☐ 238	: __ ex
☐ 225	: __ ex	☐ 239	: __ ex
☐ 226	: __ ex	☐ 240	: __ ex

soit __ exemplaires x 6,50 € = ____ €

Technosaures ☐ N°1 ☐ N°2 ☐ N°3
soit __ exemplaires x 7,66 € = ____ €
☐ Histoire de la Micro-informatique 12,99 €

..... ____ € soit au **TOTAL** = ____ €

Commande à envoyer à :
Programmez!
57 rue de Gisors - 95300 Pontoise

☐ M. ☐ Mme ☐ Mlle Entreprise : ____ Fonction : ____

Prénom : ____ Nom : ____

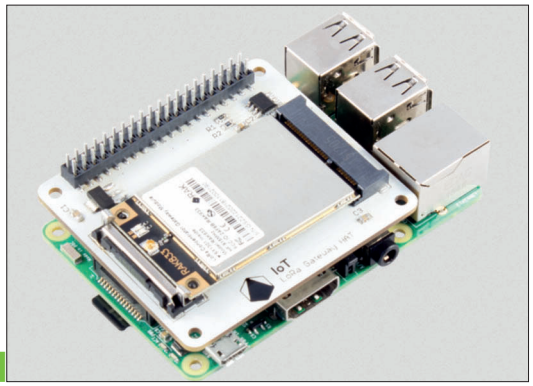
Adresse : ____

Code postal : ____ Ville : ____

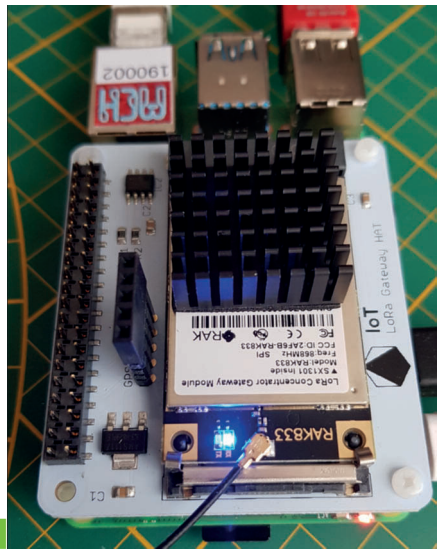
E-mail : ____ @ ____

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

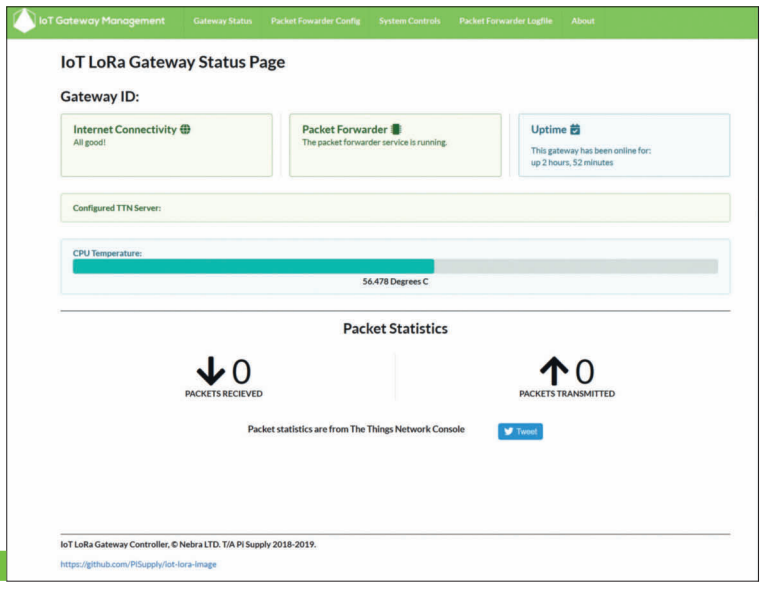
Offres pouvant s'arrêter à tout moment, sans préavis SaRP c´él eO



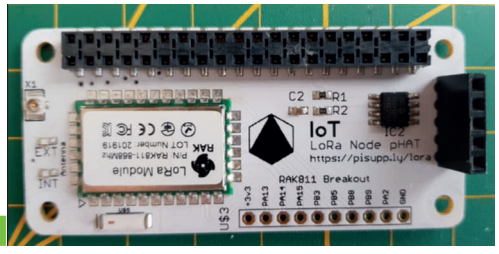
4



5



6



7

Indiquez la marque "**passerelle multicanal Raspberry Pi**" et pour le modèle, sélectionnez "**Raspberry Pi avec Pi Supply Gateway HAT**". Ne connaissant pas le type exact de l'antenne je n'ai rien indiqué. Pour terminer cliquez sur **Update Gateway** pour enregistrer toutes les informations relatives à la passerelle que vous venez de créer. Il est maintenant possible de rattacher la passerelle Raspberry Pi à ce compte.

Passerelle LoRaWAN

La passerelle IoT LoRa Gateway HAT [5] pour Raspberry Pi (868MHz/915MHz) est proposée par PiSupply (4). Elle utilise le module concentrateur de passerelle LoRa RAKWireless **RAK833** mPCIe qui est basé sur la technologie LoRa Semtech SX1301 et reçoit les 8 canaux simultanément. Elle est entièrement compatible LoRaWAN. La passerelle est fournie sous forme de carte HAT. Un radiateur non représenté ici doit obligatoirement être collé sur le RAK833 pour assurer son refroidissement. Il faudra le mettre en place AVANT de connecter la carte (5).

Le module RAK833 comporte une prise miniature u.FL sur laquelle l'antenne fournie sera connectée.

PiSupply a facilité la mise en œuvre de cette passerelle en mettant à disposition une distribution toute prête, basée sur Raspbian Lite. Elle est compatible avec toutes les versions de Raspberry Pi, y compris le Raspberry Pi 4. Elle est disponible en téléchargement [6] et représente environ 500Mo. Après avoir récupéré la distribution, transférez-la sur une carte micro SD avec Etcher si vous êtes sous Windows ou `dd` sous Linux. La Fondation Raspberry Pi a également développé une solution disponible sur Windows, Mac et Ubuntu : Raspberry Pi Imager.

Connectez un câble Ethernet au Raspberry Pi et mettez l'ensemble sous tension. Le nom de la passerelle est ***iotloragateway***, retrouvez son adresse sur votre box ou utilisez **http://iotloragateway.local**. Ouvrez un navigateur web sur cette adresse pour ouvrir la page web de configuration de la passerelle (6), cliquez sur **Packet Forwarder Config** dans la barre supérieure.

Saisissez les identifiants par défaut (*pi*, *raspberry*). Renseignez votre adresse électronique, puis les champs correspondant au compte que vous avez créé dans The Things Network : identifiant et clé de la passerelle (**Gateway Key**) fournie précédemment par TTN. Cliquez ensuite sur **Update Configuration**.

Pour démarrer la passerelle, nous devons maintenant aller dans l'onglet "**System Controls**" dans le menu du haut puis cliquer sur le bouton **Restart** pour prendre en compte les modifications et redémarrer la passerelle. Une barre de progression indique l'évolution de l'opération et à la fin, la lumière bleue du module LoRa devrait s'allumer et vous êtes redirigé(e) vers la page d'accueil. Cette page indique l'ID de votre passerelle, le nombre de paquets qu'elle a reçus et d'autres informations pratiques.

La passerelle est prête, elle peut recevoir et retransmettre les paquets envoyés par un nœud vers les serveurs TTN. Le nœud peut être celui que nous allons mettre en place maintenant, ou tout objet LoRaWAN qui sera reçu par la passerelle.

Le nœud LoRaWAN

La carte LoRa Node pHAT [3] utilisée provient également de PiSupply (7). Elle est au format du Raspberry Pi Zero. La carte

utilise le module **LoRa RAKWireless RAK811** qui est basé sur la technologie LoRa Semtech SX1276 (8). La carte peut être configurée pour différentes fréquences, pour l'Europe les fréquences vont de 863 à 870 MHz.

La carte communique avec le Raspberry Pi par l'UART en utilisant seulement 3 broches GPIO. Les autres restent libres pour connecter des capteurs. Elle prend en charge les connexions LoRaWAN et LoRaP2P. Elle dispose d'une antenne intégrée et d'un connecteur u.FL permettant de choisir son antenne et de faciliter l'intégration en boîtier. J'ai utilisé une queue de cochon u.FL – SMA et une antenne SMA 868MHz. La carte consomme moins de 50mA en émission.

La carte est fournie avec un rehausseur et deux jeux d'entretoises. Si vous utilisez la carte seule sur un Raspberry Pi, les petites entretoises feront l'affaire. Si vous souhaitez brancher autre chose pardessus la carte sur le GPIO, il faudra mettre en place le rehausseur de GPIO et les grandes entretoises. Les broches du rehausseur débordent alors sur le dessus du connecteur et permettent de connecter d'autres périphériques. J'utiliserai un BME680 pour relever les valeurs de **Température**, **Pression** et **Humidité** qui seront transmises via LoRa.

9 La carte Comporte deux straps (résistances de 0 ohm) qui relient l'antenne **INT**érieure **et** l'antenne **EXT**érieure à la sortie du circuit. L'antenne embarquée est le rectangle en haut de l'image. L'utilisation simultanée des deux antennes en parallèle perturbe l'adaptation d'impédance et le rayonnement. Pour mon utilisation (plus d'un Km de portée) j'ai utilisé l'antenne extérieure et supprimé le strap **INT** avec le fer à souder. Il faut brancher le connecteur u.FL sur la prise. ***Attention, le petit connecteur u.FL n'est pas conçu pour être branché/débranché de nombreuses fois.***

Installation du nœud LoRaWAN

Les instructions pour installer le nœud LoRaWAN sont fournies [en anglais] par Pi Supply sur leur site web [7]. Ce qui suit en est une adaptation.

Montez la carte Node pHAT sur le Raspberry Pi Zero.

Créez une carte micro SD avec Raspbian Lite (utilisez la dernière version lite - sans Desktop texte seulement). Mettez le système à jour. Éventuellement faites les mises à jour sur un Pi 3 ou un Pi 4… Ça ira plus vite. Pour utiliser le Raspberry Pi Zero il va falloir effectuer quelques réglages.

Lancez raspi-config :

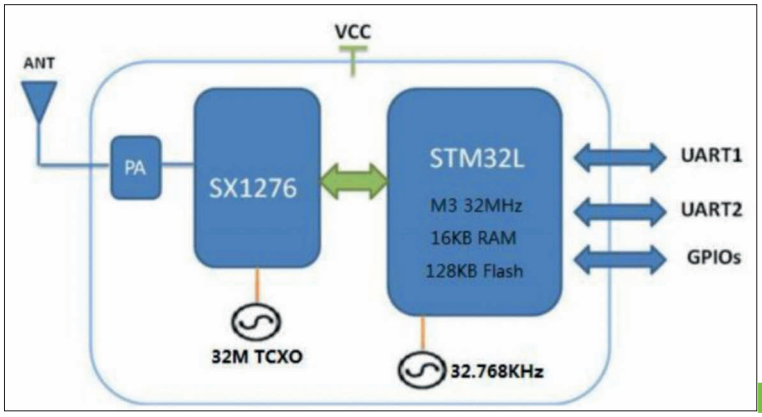
```
sudo raspi-config
```

Configurez les locales pour franciser le système puis dans **2 Network Options > N2 WiFi** configurez les paramètres pour vous connecter à votre réseau WiFi (SSID et passphrase).

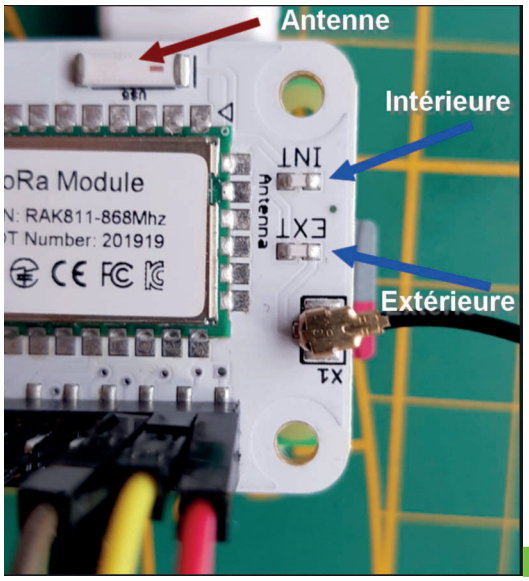
On va maintenant modifier la configuration pour utiliser l'UART matériel et désactiver la console.

Dans **5 Interfacing Options > P6 Serial** Répondez <**Non**> (voulez-vous qu'un login shell soit accessible via le port série) puis sélectionnez <**Oui**> pour activer le port série matériel.

Enfin on va intervertir les interfaces série pour que la meilleure interface (matérielle) attribuée par défaut au Bluetooth, soit attribuée à la carte LoRa. Dans un terminal saisissez :



8



9

```
sudo nano /boot/config.txt
```

Positionnez le curseur en bas du fichier et ajoutez :

```
dtoverlay=pi3-miniuart-bt
```

Sauvegardez et fermez le fichier avec "**CTRL+O**" et "**CTRL+X**". Redémarrez le Raspberry Pi pour prendre les modifications en compte.

Nous allons maintenant installer la bibliothèque RAK811 :

```
sudo apt install python3-pip
sudo pip3 install rak811
```

A partir de là, Raspbian a les éléments pour fonctionner avec la carte LoRa.

Connexion au compte TTN

Il existe deux méthodes différentes pour rejoindre le réseau The Things Network à partir du nœud Raspberry Pi pHAT : OTAA et ABP.

Activation en direct (OTAA)

L'activation en direct (OTAA) est le moyen recommandé (et le plus sûr) pour se connecter au réseau The Things Network. Les appareils exécutent une procédure pour rejoindre le réseau, au cours de

laquelle une Adresse de module Dynamique (DevAddr) est attribuée et des clés de sécurité sont négociées avec le module.

Activation par personnalisation (ABP)

Dans certains cas, vous devrez peut-être adresser en dur le DevAddr ainsi que les clés de sécurité de l'appareil. Cela signifie qu'il faudra utiliser l'activation par personnalisation (ABP). Cette stratégie peut sembler plus simple, car vous évitez la procédure utilisée pour rejoindre le réseau, mais elle présente certains inconvénients liés à la sécurité. J'ai donc opté pour la Connexion OTAA
Pour connecter un nœud au réseau TTN, il faut d'abord déterminer l'EUI (Extended Unique Identifier = identificateur unique sur 64 bits) de notre appareil. Cette valeur unique pour chaque composant permet 2^{64} valeurs, soit plus de 1.8×1019 possibilités. (de 0 à 18 446 744 073 709 551 615). Même avec un grand nombre de dispositifs LoRa, on devrait être tranquille un moment.
Ouvrez un terminal et tapez les deux lignes suivantes pour obtenir l'EUI : (la première ligne resette le rak811, vous n'avez besoin de l'utiliser qu'une seule fois après avoir démarré le système).

```
rak811 hard-reset
rak811 get-config dev_eui
```

Ce qui donne :

```
pi@raspberrypi:~ $ rak811 get-config dev_eui
3038383659386A0C
```

Et vous récupérez l'identifiant unique de votre matériel.

Le programme Python

Le programme est chargé de relever les valeurs T, P, RH toutes les minutes et de les envoyer à la passerelle. Pour ajouter un capteur, vous devrez ajouter les bibliothèques correspondant au capteur que vous utilisez. Ici j'ai installé celle du BME680.

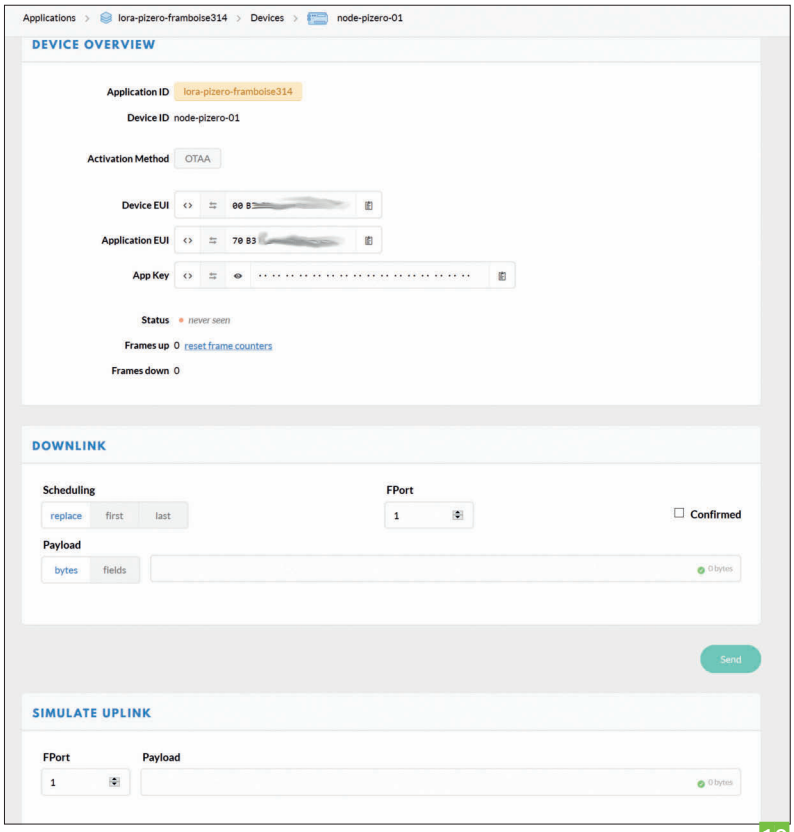
```
#!/usr/bin/env python3
"""RAK811 OTAA demo.

Ajout de la lecture du BME680
Ajout de pression, température, humidité
Minimalistic OTAA demo
Copyright 2019 Philippe Vanhaesendonck
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
   http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
SPDX-License-Identifier: Apache-2.0
"""
from random import randint
from sys import exit
from time import sleep
import bme680
from rak811 import Mode, Rak811
# from ttn_secrets import APP_EUI, APP_KEY

lora = Rak811()

# Most of the setup should happen only once...
print('Setup')
# Crée le capteur en testant les deux adresses possibles 0x76 et 0x77
try:
  capteur = bme680.BME680(bme680.I2C_ADDR_PRIMARY)
except IOError:
  capteur = bme680.BME680(bme680.I2C_ADDR_SECONDARY)

# Initialisations LoRa
lora.hard_reset()
lora.mode = Mode.LoRaWan
lora.band = 'EU868'
lora.set_config(app_eui='70xxxxxxxxxxxxF',
          app_key='05C0xxxxxxxxxxxxxxxxxxxxxxxxx47F')

print('Joining')
lora.join_otaa()
# Note that DR is different from SF and depends on the region
# See: https://docs.exploratory.engineering/lora/dr_sf/
# Set Data Rate to 5 which is SF7/125kHz for EU868
lora.dr = 5

print('Sending packets every minute - Interrupt to cancel loop')
print('You can send downlinks from the TTN console')
try:
  while True:
    # Si les données du capteur sont lues
    if capteur.get_sensor_data():
      # Afficher la température
      Temp = int(10*round(capteur.data.temperature, 1))
#       print(f'Température : {capteur.data.temperature:.1f} °C')
      print('Température : ', Temp/10)
      # Afficher la pression
      Pression = int(10*round(capteur.data.pressure, 1))
      print('Pression : ', Pression/10)
#       print(f'Pression : {capteur.data.pressure:.1f} hPa')
      # Afficher l'humidité relative
      HR = int(2*capteur.data.humidity)
#       print(f'Humidité : {capteur.data.humidity:.1f} %HR')
      print('Humidité : ', HR/2)
      print("==============")

    print('Envoi du paquet de données')
    # Cayenne lpp random value as analog
#     lora.send(bytes.fromhex('0167{:04x}'.format(randint(0, 0x7F))))
    lora.send(bytes.fromhex('0167{:04x}0273{:04x}0368{:02x}'.format(Temp, Pression, HR)))

    while lora.nb_downlinks:
      print('Received', lora.get_downlink()['data'].hex())
```

```
    sleep(60)
except:  # noqa: E722
  pass

print('Cleaning up')
lora.close()
exit(0)
```

Vous pouvez également récupérer le programme sous forme d'archive .zip en [8]. Ce programme est disponible sur le Github d'Amédée Bulle [9]. Le programme ci-dessus est un test, il comporte encore des éléments inutiles, reliquats du programme original. On y trouve également la réception d'un downlink (message descendant vers le nœud) qui pourrait être utilisé pour déclencher une action (allumer la chaudière, déclencher une alarme, allumer une lampe…).
En ligne 39 on resette le circuit LoRa (à faire au moins une fois après la connexion de la carte à l'alimentation).
En ligne 42 on entre les identifiants de l'appli et la clé correspondante. (Il faudra rentrer vos propres identifiants).
En ligne 46 on se connecte à la passerelle en OTAA. Over The Air : les clefs de chiffrement sont obtenues par un échange avec le réseau.
Des lignes 56 à 70 on récupère les données climatiques et on les met en forme pour les inclure dans la payload.
En ligne 75 on crée la trame au format CayenneLPP et on l'envoie. Le programme transmet la payload (voir ci-dessous) actualisée toutes les minutes pour les tests (à modifier selon vos besoins et en fonction de la réglementation).
Lorsque vous lancez le programme en lignes de commandes puis que vous fermez la fenêtre, il sera arrêté si vous étiez en SSH (puTTY). Pour continuer l'exécution après fermeture de la fenêtre ajoutez un **&** à la fin de la ligne.

Payload Cayenne LPP

Pour diminuer le nombre d'octets transmis, il est intéressant d'utiliser Cayenne LPP. C'est un codage open source créé par Cayenne disponible sous forme de bibliothèque. Une payload CayenneLPP contient des métadonnées qui permettent de savoir quelles données sont transmises. Pour chaque valeur le premier octet envoyé est l'identificateur, l'identifiant du canal. Le deuxième octet est le type de données, le Data Type. On trouve ensuite la valeur mesurée. Vous trouverez plus d'informations sur Cayenne LPP en [10].
Décortiquons cette charge utile de 11 octets :
01 67 00 D7 02 73 25 C9 03 68 51
01 67 00 D7 :
• **01** = 1d => canal 1
• **67** = 103d => Capteur de température valeur sur 2 octets
• **00 D7** = 215d => **21.5°C** (valeur/10)
02 73 25 C9 :
• **02** = 2d => canal 2
• **73** = 115d => Baromètre valeur sur 2 octets
• **25 C9** = 9673d => **967.3 hPa** (valeur/10)
03 68 51 :
• **03** = 3 => canal 3
• **68** = 104d => Capteur d'humidité valeur sur 1 octet



10

• **51** = 81d => **40.5 %** (valeur/2 car bit de poids faible =0,5% HR)
En 11 octets, CayenneLPP a donc permis de passer les valeurs de 3 capteurs et le type des informations transmises. Ce format est très utilisé dans le monde de l'IoT car il facilite la transmission des données, tout en conservant une occupation réduite de la bande passante.

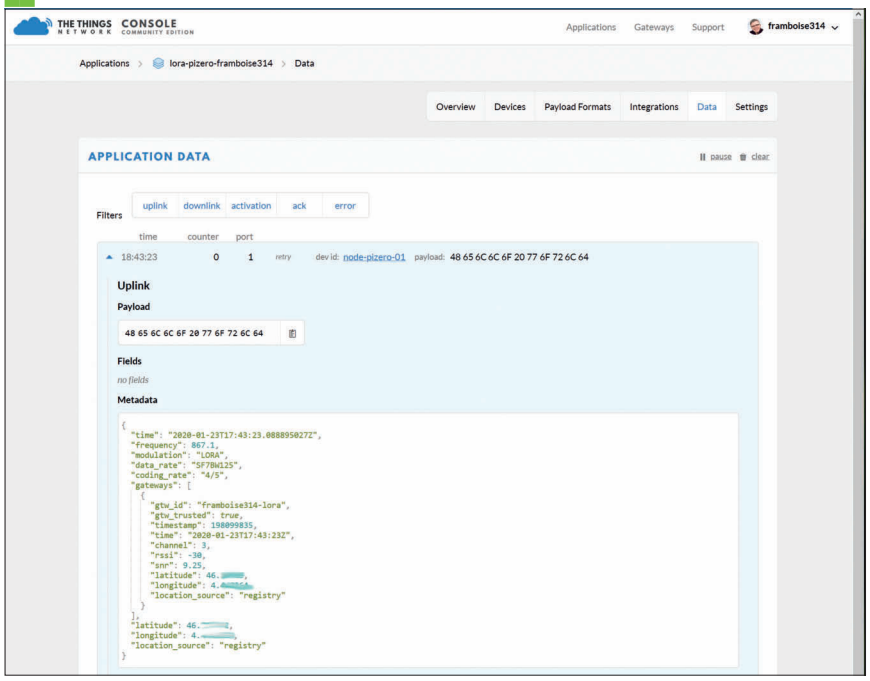
Inscription du nœud sur The Things Network

Nous avions précédemment configuré la Gateway sur TTN, maintenant on passe à la configuration du nœud. Connectez-vous à votre compte The Thing Networks. Sur l'écran d'accueil de TTN cliquez sur l'icône **APPLICATIONS** (image 6) puis sur **Add Application**.
Donnez un nom à votre application (attention tous les caractères ne sont pas autorisés) et ajoutez une description. Sélectionnez le serveur TTN qui sera sollicité, ici le serveur européen **ttn-handler-eu**. Validez vos choix avec le bouton **Add Application**. Vous récupérez un numéro d'application **EUI** unique, qu'on va associer avec votre carte LoRa.
Dans l'onglet **Devices**, cliquez sur **Register Device**. Donnez un nom à votre matériel, ce nom ne pourra pas être modifié. Collez le **Device EUI** récupéré plus haut en ligne de commande sur le Raspberry Pi Zero. **App Key** sera générée par la suite. Cliquez sur **Register** pour enregistrer vos modifications.
Après un moment vous avez accès à la fenêtre **Device Overview** (10). L'**App Key** existe et vous pouvez la voir (clic sur l'icône œil) et la copier (clic sur l'icône feuille à droite du champ). En haut de la fenêtre (dans le fil d'Ariane) cliquez sur le nom de votre applica-

tion puis dans le menu sur **Payload Format**. Dans la rubrique **Payload Format**, choisissez CayenneLPP qui est le format utilisé dans ce tutoriel. L'utilisation de Cayenne LPP n'est bien entendu pas obligatoire, et vous pourrez créer votre propre format par la suite. Validez avec **Save** en bas de la fenêtre.

A partir de là 11, vous devriez voir apparaître des données dans la rubrique **Data**. La payload brute est affichée. En dépliant les données vous aurez une vue sur l'ensemble des informations reçues par The Things Network.

11



Conclusion

Avec ce kit, PiSupply propose un ensemble qu'on peut rapidement mettre en œuvre pour découvrir le monde LoRaWAN. La disponibilité d'une distribution « clé en main » pour la passerelle est un plus incontestable. Il suffira ensuite de quelques lignes de Python pour développer un nœud opérationnel rapidement. Les transmissions proposées ici sont des tests. En exploitation il faudra respecter un certain nombre de contraintes [11] : 30 secondes d'émission par dispositif et par jour. Cela correspond pour une payload de 10 octets à 2 messages par jour en SF12 ou 500 messages par jour en SF7

Il est également possible d'envoyer des données vers le nœud (downlink). Les conditions sont encore plus restrictives que pour la remontée d'informations.

Les données ne sont pas conservées indéfiniment sur The Things Network [11]. Elles sont supprimées au bout de 7 jours. Si vous avez besoin de les conserver vous pouvez les récupérer avec l'API ou utiliser le service Cayenne pour créer des tableaux de bord (dashboard) et afficher les données.

Sources

[1] https://uk.pi-supply.com/search?type=product&q=lora
[2] https://uk.pi-supply.com/products/iot-lora-gateway-hat-for-raspberry-pi?lang=fr
[3] https://uk.pi-supply.com/products/iot-lora-node-phat-for-raspberry-pi?lang=fr
[4] https://www.thethingsnetwork.org/
[5] https://uk.pi-supply.com/products/iot-lora-gateway-hat-for-raspberry-pi?lang=fr
[6] https://github.com/PiSupply/iot-lora-image/releases
[7] https://learn.pi-supply.com/make/getting-started-with-the-raspberry-pi-lora-node-phat/
[8] https://www.framboise314.fr/wp-content/uploads/2020/01/lora_node_progpy.zip
[9] https://github.com/AmedeeBulle/pyrak811/blob/master/examples/otaa.py
[10] https://community.mydevices.com/t/cayenne-lpp-2-0/7510
[11] https://www.thethingsnetwork.org/docs/applications/storage/
[12] https://developers.mydevices.com/cayenne/signin/
[13] https://www.thethingsnetwork.org/article/lorawan-distance-world-record
[14] https://www.raspberrypi.org/downloads/

François Mocq
Electronicien, radioamateur (F1GYT), Technicien de maintenance puis formateur d'adultes en maintenance informatique, réseaux et télécom. Créateur du blog www.framboise314.fr *et auteur des livres "Raspberry Pi 4" et "Scratch et Raspberry Pi" parus aux Editions ENI*

Programmer un module Sonoff Basic avec un Raspberry Pi

La marque itead recouvre de nombreux produits de domotique. Parmi ceux-ci, la gamme Sonoff dans laquelle on trouve des prises connectées ou des commutateurs comme le Sonoff Basic. On trouve ce commutateur WiFi pour moins de 10 euros sur les sites en ligne. Pourquoi s'intéresser à ce commutateur me direz-vous ? Il est à base d'ESP8266 et il est tout à fait possible de le programmer avec un Raspberry Pi. Pas besoin de dégainer la grosse artillerie et d'allumer votre super PC, un Raspberry Pi suffira largement pour vous amuser avec ce module sur un coin d'établi !

Avertissement : ***Ce module est alimenté par la tension du secteur 240v. Le contact avec les fils soumis à cette tension est mortel ! Il vous appartient de prendre toutes les précautions pour éviter tout contact même accidentel avec le secteur. Les pistes de circuit imprimé du module transportent le 240v. Faites toujours comme si le module était connecté au secteur. On oublie vite que la prise est branchée ! Remettez systématiquement le module dans son boîtier avant de le connecter au secteur. L'auteur de l'article et Programmez! ne pourront être tenus pour responsable en cas d'accident ou de dommage.***

Le module Sonoff Basic

Le module Sonoff Basic WiFi Wireless est un **interrupteur piloté en WiFI**, protégé dans un boîtier **IP66** (protégé contre les poussières et les jets d'eau). Enfin ça, c'est la doc qui le dit. Pour les projections d'eau je serais plutôt réservé.

Caractéristiques

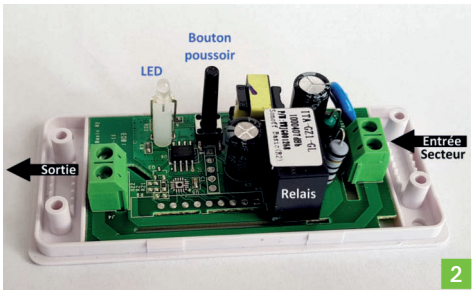
- Allumage et extinction à distance ;
- Application gratuite pour iOS et Android mobile eWeLink ;
- Contrôle sur réseau local – Allumez ou éteignez l'appareil même sans accès à Internet ;
- État du dispositif affiché en temps réel sur l'application ;
- Réglage de minuteries programmées/ compte à rebours/boucles pour activer/désactiver à une heure précise via l'application ;
- Intégration à la smarthome (maison intelligente) ;
- Allumage/Extinction d'un groupe d'appareils d'un seul coup ;
- Déclenchement possible par la température, le bruit … en fonction des capteurs installés ;
- Fonctionne avec Amazon Alexa, Google Assistant, IFTTT, Google Nest et de nombreux logiciels domotiques.

Sympa tout ça, non ? Moi ce qui me gêne un peu c'est de passer par l'appli eWelink pour allumer ma lampe de salon. Pas parce que ça me gêne qu'un serveur chinois enregistre cette action (quoique), mais parce qu'il semble inconcevable de mettre en œuvre toute cette technologie pour allumer… une lampe.

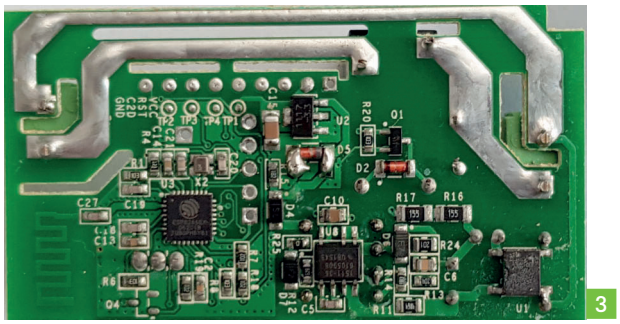
Le matériel

Le module reçu était présenté dans une boîte en carton fin (1). Les vis de fixation des plaques de blocage des fils ne sont pas montées, elles sont à part, dans un petit sachet en plastique. Ces deux plaques s'enlèvent donc très facilement. Le boîtier est juste clipsé à la base et s'ouvre très facilement avec l'ongle ou avec un tournevis fin.

1



2


3

Dans le boîtier du Sonoff Basic

A l'intérieur (2) on découvre le module monté sur son circuit imprimé. Le secteur 240v entre par la droite et ressort par la gauche après avoir transité via le relais. Un bouton poussoir accessible via l'extérieur du boîtier et une LED nous permettront d'interagir avec le module par la suite.

Sous la carte (3) on repère les pistes transportant la tension secteur (les grosses pistes étamées), séparées par des vides sans circuit pour éviter les amorçages, étincelles… Sur la gauche le circuit U3 qui est notre ESP8266. Le schéma de la carte est disponible en ligne [1].

Comment programmer ce module ?

Facile… En attaquant les entrées Tx et Rx de l'ESP8266. Sauf que…le fabricant a juste oublié de monter le connecteur pour y

accéder. On voit sur 4 les pastilles marquées 1, 2, 3, 4 et 5 qui n'attendent que l'arrivée d'un connecteur au pas de 2,54mm (1/10 pouce) pour servir. Sur votre module, il y a de fortes chances que les pastilles soient obstruées par de la soudure. Il faudra d'abord les en débarrasser avant de monter le connecteur et de le souder avec précaution.

Voilà ce que vous devriez obtenir. A présent notre ESP8266 est accessible via son port série (5) et nous allons pouvoir le programmer et recevoir les messages qu'il renvoie sur le port série. Vous avez accès à l'alimentation 3,3v et GND, au port série Rx et Tx, ainsi qu'à un GPIO, le N° 14. Notez bien que :

Le circuit doit être alimenté en 3,3v, **pas en 5v**

Il ne faut **pas connecter** l'alimentation 3,3v quand le module est relié au secteur. Sous une alimentation de 3,3v seule la LED s'allume, **le relais ne colle pas**

PROGRAMMER L'ESP8266 AVEC LE RASPBERRY PI

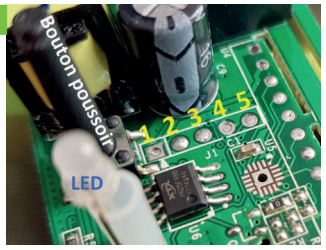
Si vous faites de l'Arduino, vous avez déjà tout ce qu'il faut pour programmer l'ESP8266, il suffit d'adapter l'IDE Arduino pour qu'il soit compatible avec les ESP. Si vous ne l'utilisez pas encore, voilà une occasion d'installer l'IDE sur votre Raspberry Pi.

L'IDE Arduino sur Raspbian

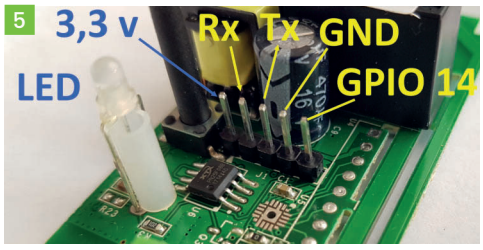
L'IDE Arduino s'installe aisément sur Raspbian. Naviguez sur la page arduino.cc et téléchargez depuis la page Download [2] la version arduino-1.8.12-linuxarm.tar.xz. (ou une version plus récente en fonction de la date à laquelle vous suivrez ce tutoriel). Installez l'IDE Arduino sur Raspbian. Dans un terminal tapez les lignes suivantes :

```
tar xvf arduino-1.8.12-linuxarm.tar.xz
cd arduino-1.8.12/
sudo ./install.sh
```

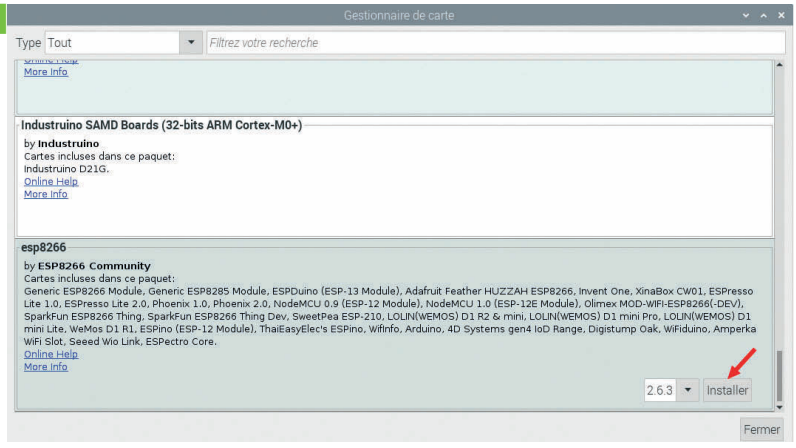
4



5



6



Vous trouverez une nouvelle icône dans le menu Programmation : **Arduino IDE**. Dans les types de cartes (Outils => Type de carte) l'ESP8266 ne figure pas, nous allons l'ajouter.

Nota : *Lors de l'installation j'ai eu un message d'erreur, je l'ai contourné en créant le fichier manquant manuellement.*

Le message :

```
pi@raspberrypi:~/Downloads/arduino-1.8.12 $ sudo ./install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE...
touch: impossible de faire un touch '/root/.config/mimeapps.list': Aucun fichier ou dossier de ce type
/usr/bin/xdg-mime: 848: /usr/bin/xdg-mime: cannot create /root/.config/mimeapps.list.new: Directory nonexistent
done!
```

Le contournement :

```
pi@raspberrypi:~/Downloads/arduino-1.8.12 $ sudo mkdir /root/.config
pi@raspberrypi:~/Downloads/arduino-1.8.12 $ sudo touch /root/.config/mimeapps.list
```

Puis relancez l'installation et il n'y a plus d'erreur.

Ajouter l'ESP8266 à l'IDE Arduino

Rendez-vous sur la page web [3] et récupérez le lien .json (https://arduino.esp8266.com/stable/package_esp8266com_index.json).

Dans l'IDE Arduino, ouvrez la fenêtre **Fichier** => **Préférences** et collez le lien json dans la zone de texte **URL de gestionnaire de cartes supplémentaires**.

Ouvrez **Outils** => **Type de Carte** => **Gestionnaire de carte**. Cherchez la bibliothèque **ESP8266 by ESP8266 Community** (6) et Cliquez sur le bouton **Installer**. Vous pouvez suivre l'installation sur une barre de défilement. Attendez la fin de l'installation, fermez la fenêtre et redémarrez l'IDE Arduino.

Vous avez maintenant accès aux ESP8266 dans la liste des cartes (**Outils** => **Type de carte**). Choisissez celui qui correspond à votre modèle. Ici ce sera **Generic ESP8266 Module**. Vérifiez que le port USB **/dev/ttyUSB0** est bien indiqué. Si vous ne voyez pas d'USB0, laissez l'ESP connecté au port USB et redémarrez le Raspberry Pi. Sélectionnez le port **/dev/ttyUSB0** pour communiquer avec l'ESP8266.

MODIFIER LE PROGRAMME DU MODULE SONOFF

Attention : pour cette partie, NE CONNECTEZ PAS LE MODULE AU SECTEUR.

Utilisez un adaptateur USB-Série (7) qui fonctionnera aussi bien sur un PC que sur un Raspberry Pi. Connecter les fils à l'adaptateur USB. On a besoin des fils **TxD** et **RxD** (émission et réception de données du port série) et de l'alimentation +3,3v et GND. Connectez ces fils sur les broches correspondantes (Image 05) du module Sonoff. Attention de ne pas faire d'inversion, surtout au niveau de l'alimentation, ce qui serait fatal au module. La pin du GPIO14, à droite, reste libre.

Connectez l'adaptateur USB-Série à un port USB du Raspberry Pi. Sur ce modèle d'adaptateur, une LED indique que l'adaptateur est sous tension. La LED du module Sonoff devrait se mettre à clignoter au rythme d'une fois par seconde. On peut maintenant passer le module en mode programmation : débranchez l'adaptateur USB du Raspberry Pi. Appuyez sur la tige du bouton poussoir. Maintenez le bouton appuyé et rebranchez l'adaptateur USB. **Maintenez encore**

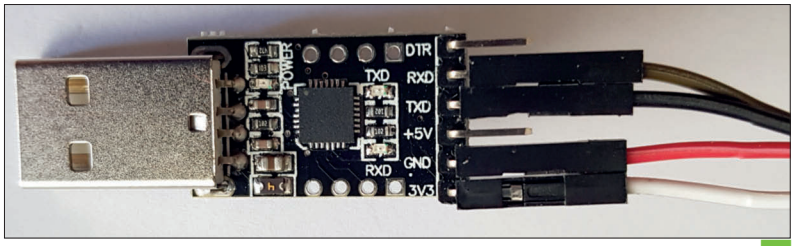
le bouton appuyé plus de 3 secondes. Vous pouvez relâcher le bouton. La LED ne doit pas clignoter. Si elle clignote c'est que vous n'avez pas appuyé assez longtemps ! Recommencez.
Votre module Sonoff Basic est maintenant passé en mode programmation, on peut charger un programme dans l'ESP8266 du module Sonoff Basic. Lancez l'IDE Arduino sur le Raspberry Pi (**Menu principal** > **Programmation** > **Arduino IDE**). Saisissez ce programme, disponible également en téléchargement [4].

```
/*
SONOFF Blink par Michel Deslierres
Modifié par François Mocq
Basé sur le programme Blink pour l'ESP8266 de Simon Peter
Fait clignoter la LED verte du Sonoff : deux flash rapides, un temps de pause
Ce code exemple est dans le domaine public
La LED verte du module Sonoff Basic est connectée au GPIO 13
*/
const int LED_PIN = 13;

void setup() {
pinMode(LED_PIN, OUTPUT); // Initialiser la pin LED_BUILTIN en sortie
}

// cette fonction loop tourne indéfiniment
void loop() {
digitalWrite(LED_PIN, LOW);  // Allume la LED (Noter que LOW est le niveau de la tension
                             // mais ici la LED est allumée car
                             // elle est active à l'état bas sur l'ESP-01)
delay(10);                   // Attendre 10ms
digitalWrite(LED_PIN, HIGH); // Eteindre la LED en mettant la sortie à 1 (3,3v)
delay (100);                 // Attendre 100ms
digitalWrite(LED_PIN, LOW);  // Allumer la LED
delay(10);                   // Attendre 10ms
digitalWrite(LED_PIN, HIGH); // Eteindre la LED
delay(500);                  // Attendre 500ms
}
```

Cliquez sur le bouton de téléversement de l'IDE. A la fin de la programmation, vous devriez avoir un message « **Hard resetting via RTS pin…** » dans le bas de la fenêtre de l'IDE Arduino.
Pour lancer le programme il faut redémarrer le module Sonoff. Débranchez et rebranchez l'adaptateur USB-Série sur la prise USB du Raspberry Pi. Après ce reset la LED doit clignoter en envoyant deux flashs brefs suivis d'une pause d'une demi seconde. J'ai modifié le programme blink dans ce sens car si on a un blink qui fait clignoter la LED exactement comme avant qu'on reprogramme le module… on se demande si ça a fonctionné ou pas.

Piloter le module Sonoff depuis une page web

On a fait la première partie du chemin en transférant avec succès un nouveau programme dans le module Sonoff. Au passage vous noterez qu'il ne pourra plus être utilisé comme le constructeur le prévoyait, c'est à vous de gérer tout ça maintenant. Dans l'IDE Arduino, saisissez le programme présenté en 8, que vous pouvez télécharger en ligne [5]. Modifiez les identifiants de connexion en fonction de vos identifiants. Déconnectez l'adaptateur USB pour mettre le



7

Sonoff hors tension. Appuyez sur le bouton du Sonoff, tenez appuyé. Rebranchez la prise USB. Attendez au moins 3 secondes que l'ESP8266 passe en mode programmation. Cliquez sur le bouton de téléversement de l'IDE Arduino. L'IDE compile votre programme (croquis) puis le transfère dans l'ESP8266.
Si tout se passe bien, à la fin du transfert vous obtenez un message «**Leaving**…Hard resetting via RTS pin… » Par contre le reset c'est vous qui allez le faire en débranchant et en rebranchant l'adaptateur USB du Raspberry Pi. L'ESP8266 redémarre et la LED doit clignoter plusieurs fois (ligne 52 à 64) en attendant que l'ESP8266 se voie attribuer une adresse IP par le DHCP de la box. Lorsque la LED s'arrête de clignoter (normalement ça clignote juste 3 ou 4 fois maxi) votre module Sonoff est prêt à recevoir vos ordres.

Utiliser le moniteur série

Si vous regardez le programme, il comporte de nombreuses lignes commençant par **Serial.println**. Ces lignes sont inutiles en utilisation normale du module Sonoff car il n'a pas d'écran. Par contre, pour la mise au point c'est un outil très intéressant. Les lignes Serial.println envoient les informations que vous décidez vers le port série. Il suffit d'ouvrir le terminal via le menu de l'IDE (**Outils** > **Moniteur Série**), de vérifier que la vitesse du moniteur (115200) correspond à celle que l'ESP8266 a reçue du programme (ligne 45) et vous verrez apparaître des indications sur ce qu'il se passe. Vous pouvez bien entendu ajouter d'autres infos comme l'état de la LED et du relais par exemple.

```
Connection au WiFi en cours...
.(IP unset)
6
.(IP unset)
6
.(IP unset)
6
.192.168.1.26
3
Connecté à Livebox-57A9
Accédez au serveur à l'adresse : 192.168.1.26
Serveur web prêt..
```

Avec ces informations et l'arrêt des clignotements de la LED vous savez que le module Sonoff est connecté à la box. Il n'y a plus qu'à commander le commutateur à distance.

Commande du module Sonoff Basic sur une page web

Ouvrez une page web sur l'adresse du module. C'est facile si vous avez ouvert le moniteur série, il affiche l'adresse IP du module

8

```
/*
  D'après Simple Wifi Switch de Jeffrey Roshan
  https://gist.github.com/jeffreyroshan/

  Adaptation Française : François Mocq
  www.framboise314.fr
  */

// Inclure les bibliothèques utilisées dans le script

// Gestion du WifI pour l'ESP8266
#include <ESP8266WiFi.h>

// Client capable de se connecter à une adresse IP
#include <WiFiClient.h>

// Serveur web capable de gérer les requêtes HTTP POST et GET d'un client unique
#include <ESP8266WebServer.h>


// Identifiants de connexion à la box
const char* ssid = "Livebox-57A9";
const char* password = "2585                279";


ESP8266WebServer server(80);

String web_on_html = "<h2>Etat actuel : ON</h2><p><a
href=\"on\"><button>ON</button></a> <a
href=\"off\"><button>OFF</button></a></p>";
String web_off_html = "<h2>Etat actuel : OFF</h2><p><a
href=\"on\"><button>ON</button></a> <a
href=\"off\"><button>OFF</button></a></p>";

int gpio_13_led = 13;
int gpio_12_relais = 12;

void setup(void){
  //  Initialisation des sorties LED et Relais
  // Eteindre la LED
  pinMode(gpio_13_led, OUTPUT);
  digitalWrite(gpio_13_led, HIGH);

  // Décoller le Relais
  pinMode(gpio_12_relais, OUTPUT);
  digitalWrite(gpio_12_relais, LOW);

  // Configurer la vitesse du port série
  Serial.begin(115200);
  delay(5000);

  // Connecter l'ESP8266 au WiFi
  WiFi.begin(ssid, password);
  Serial.println("Connection au WiFi en cours...");

  // Attendre la connexion en faisant clignoter la LED
  while (WiFi.status() != WL_CONNECTED) {
    // Allumer la LED
    digitalWrite(gpio_13_led, LOW);
    delay(500);
    Serial.print(".");
    // Attente de l'adresse IP
    Serial.println(WiFi.localIP());
    Serial.println(WiFi.status());
    // Eteindre la LED
    digitalWrite(gpio_13_led, HIGH);
    delay(500);
  }

  Serial.println("");
  Serial.print("Connecté à ");
  Serial.println(ssid);
  Serial.print("Accédez au serveur à l'adresse : ");
  Serial.println(WiFi.localIP());

  // Réponse du serveur
  server.on("/", [](){
    if(digitalRead(gpio_12_relais)==HIGH) {
      server.send(200, "text/html", web_on_html);
    } else {
      server.send(200, "text/html", web_off_html);
    }
  });

  // Réponse lors de l'appui sur ON
  server.on("/on", [](){
    server.send(200, "text/html", web_on_html);
    // Allumer la LED, coller le Relais
    digitalWrite(gpio_13_led, LOW);
    digitalWrite(gpio_12_relais, HIGH);
    delay(1000);
  });

  // Réponse lors de l'appui sur OFF
  server.on("/off", [](){
    server.send(200, "text/html", web_off_html);
    // Eteindre la LED, décoller le Relais
    digitalWrite(gpio_13_led, HIGH);
    digitalWrite(gpio_12_relais, LOW);
    delay(1000);
  });

// Démarrage du serveur web
  server.begin();
  Serial.println("Serveur web prêt..");
}

// Boucle principale du programme
// Gestion du client web
void loop(void){
  server.handleClient();
}
```

Sonoff. Sinon allez fouiner dans le DHCP de votre box. La page web peut être ouverte sur le Raspberry Pi (on en profite puisqu'il est à côté), sur un PC, un MAC, un smartphone ou une tablette.

Normalement vous obtenez une page web comme celle qui est présentée sur 9. La LED est éteinte sur le module Sonoff. Cliquez sur le bouton **ON** : La LED s'allume. Attention, comme le module n'est pas encore relié au secteur, le relais ne colle pas et vous n'entendez rien.

La page web change et vous obtenez ce qui est présenté sur 10. La LED est allumée sur le module Sonoff. Cliquez sur le bouton **OFF** : La LED s'éteint et le relais ne fait toujours pas de bruit. Mais cette manipulation nous permet de vérifier que tout se passe bien et que le programme fonctionne. Vous pouvez voir cet essai en vidéo [6].

Utilisation en réel du module Sonoff Basic

NE CONNECTEZ RIEN AU SECTEUR POUR LE MOMENT

Débranchez tous les fils provenant de l'adaptateur USB qui sont connectés sur le module Sonoff. Branchez les fils secteur sur l'entrée du module. *Remontez le module dans son boîtier* **avant de le connecter au secteur**. Prenez toutes les précautions pour ne pas entrer en contact avec la tension 240 v.

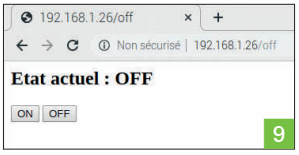
Branchez la prise de courant. La LED va clignoter plusieurs fois en attendant l'adresse IP. Quand la LED ne clignote plus, allez sur la page web et actionnez les boutons. Cette fois la LED s'allume et s'éteint, mais vous entendez aussi le relais qui claque à chaque commande et si vous avez connecté une lampe 240v à la sortie du module, elle va s'allumer et s'éteindre en même temps que la LED.

Conclusion

Nous voici à la fin de ce tutoriel. Vous aurez vu comment modifier le programme d'un module Sonoff Basic peu cher pour l'adapter à vos besoins. Ce n'est qu'une mise en bouche, car avec ces infos, vous pouvez maintenant automatiser la commande. Il suffit de piloter depuis un programme en Python pour pouvoir gérer par programme l'allumage et l'extinction de ce qui est branché à la sortie du module. Vous avez sans doute remarqué que le GPIO14 qui était lui aussi accessible n'a pas été utilisé ici, et puis pourquoi ne pas récupérer les broches Tx et Rx comme E/S ? On pourrait y brancher d'autres choses : un autre relais… une sonde de température DS18B20 pour faire une espèce de thermostat… ? Mais ça, c'est une autre histoire.

Sources

[1] https://www.framboise314.fr/wp-content/uploads/2020/04/Sonoff-Schema.pdf
[2] https://www.arduino.cc/en/Main/Software
[3] https://github.com/esp8266/Arduino#installing-with-boards-manager
[4] https://www.framboise314.fr/wp-content/uploads/2020/04/blink_sonoff.zip
[5] https://www.framboise314.fr/wp-content/uploads/2020/04/test_web_FR.zip
[6] https://bit.ly/2RxECUe



François Mocq
Electronicien, radioamateur (F1GYT), Technicien de maintenance puis formateur d'adultes en maintenance informatique, réseaux et télécom. Créateur du blog www.framboise314.fr *et auteur des livres "Raspberry Pi 4" et "Scratch et Raspberry Pi" parus aux Editions ENI*

GPS USB avec le Raspberry Pi

Aujourd'hui le GPS en tant qu'appareil autonome a plus ou moins disparu, distancé par les GPS embarqués et par les applications sur les smartphones. Monter son propre système GPS est un moyen de découvrir ce système et les informations qu'il transmet. Le Raspberry Pi est tout à fait capable de gérer le GPS et l'affichage de la carte, c'est lui que j'ai retenu pour cette présentation.

Le GPS, comment ça marche ?

Principe

Le système GPS repose sur une constellation de satellites en orbite autour de la Terre (1). Chaque satellite embarque plusieurs horloges atomiques (4 pour Galileo). Chaque satellite transmet des informations permettant de le localiser précisément dans l'espace ainsi que l'heure exacte de l'émission du message. Avec ces informations, le récepteur mesure le temps que met le signal pour lui parvenir. Il peut donc calculer la distance précise qui le sépare du satellite.

Avec un seul signal reçu d'un satellite, le récepteur sait uniquement qu'il est positionné sur un cercle centré sur le satellite.

Avec 2 satellites, le récepteur se situe sur un des deux points d'intersection.

Avec 3 satellites, le récepteur peut déterminer le point unique où il se situe. Ici on a visualisé la position sur des cercles. Dans la réalité, le récepteur se positionne sur une sphère. L'intersection de 3 sphères correspond à 2 points. Il faudra donc un quatrième satellite pour connaître la position réelle. Le deuxième point d'intersection des sphères ne se trouve pas sur la sphère terrestre et peut aussi être éliminé par calcul. Plus le nombre de satellites reçu est élevé, plus la précision est grande. Avec plus d'informations il devient également possible de déterminer l'altitude par rapport au niveau de la mer.

Pour sécuriser son accès aux données de positionnement, chaque pays ou communauté développe son propre système : **GPS** pour les américains, **GLONASS** pour les russes, **Beidou** pour les chinois, **Galileo** pour l'Europe, etc. Des accords entre certains acteurs permettent une interopérabilité des systèmes.

Trame NMEA

La norme NMEA 0183 définit une communication série permettant de transmettre une trame à de multiples récepteurs. Le format utilisé est l'ASCII. Cette norme propriétaire est payante, mais elle a été étudiée par rétro-ingénierie et à partir de sources publiques. Le récepteur reçoit les informations des satellites, les traite, et génère des trames NMEA contenant les informations qu'il a extraites/calculées. Ce sont ces trames qui sont envoyées en série via le port USB par le récepteur.

Il existe différents types de trames (une trentaine) chaque type est défini par les premiers caractères émis. Par exemple **$GPGGA**, … indique une trame GPS de type GGA. Les autres acteurs possèdent leurs propres identifiants : GA = Galileo, BD ou GB = Beidou, GL = GLONASS ; GN = Signaux mixtes GPS+GLONASS.

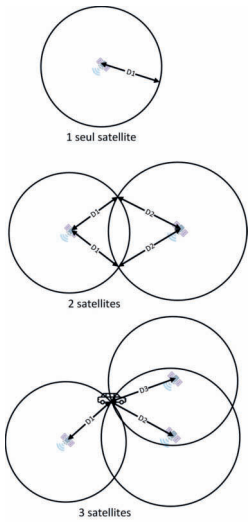
Exemple de trame NMEA

Voici un exemple de trame NMEA reçue avec ce GPS :

$GNGGA,145144.00,4648.64138,N,00426.90498,E,1,10,1.20,389.8,M,47.1,M,,*43

- GNGGA : Type de trame : GGA (GPS Fix et Date)
- 145144.00 : Trame envoyée à 14 h 51 min 44 s (heure UTC)
- 4648.64138, N : Latitude 46,4864138° Nord
- 00426.90498, E : Longitude 4.2690498° Est
- 1 : Type de positionnement 1 signifie GPS
- 10 : Nombre de satellites utilisés pour calculer les coordonnées
- 1.20 : Précision horizontale (où 1 est la valeur optimale, de 2 à 3 précision excellente, de 5 à 6 bonne, au-delà de 8 mesures non acceptable)
- 389.8, M : Altitude du récepteur en mètres
- …,…,… : D'autres informations peuvent figurer ici
- *43 : Somme de contrôle de parité

D'autres trames contiennent l'heure, la latitude et la longitude, ainsi que la vitesse et la direction de déplacement. C'est l'ensemble des informations récupérées dans ces trames qui est utilisé par les logiciels GPS.



1

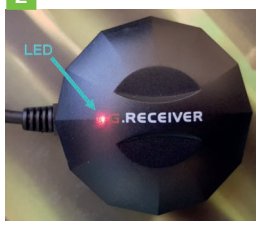
Récepteur GPS

Choix du récepteur

J'ai opté pour un GPS miniature GN803-G à moins de 15€, trouvé en ligne [1] mais d'autres modèles conviendront. On trouve des GPS USB de 10€ à plusieurs centaines d'euros, en fonction de la puissance de calcul, de la précision, de la résistance aux intempéries… 2

Le récepteur contient l'ensemble du matériel, antenne et récepteur, dans un boîtier quasiment circulaire. Une LED clignote toutes les secondes, elle indique le verrouillage du récepteur sur les satellites. Le boîtier a une épaisseur de 20mm 3 . Il est équipé à la partie inférieure d'un aimant puissant permettant de coller le récepteur sur la carrosserie d'une voiture. Une protection souple évite les rayures. Le câble USB d'un mètre cinquante est suffisant pour amener le signal dans l'habitacle, ou pour poser le récepteur sur un rebord de fenêtre si vous êtes chez vous. La sensibilité du récepteur est suffisante pour faire les tests à l'intérieur, avec la possibilité d'obtenir une précision dégradée.

2



3

Mise en service du GPS

Installez la dernière version Desktop complète de Raspbian et mettez-la à jour. Démarrez le Raspberry Pi. Quand le bureau est affi-

ché, Connectez le récepteur GPS sur un port USB. Ouvrez un terminal et tapez **dmesg** :

```
[ 705.861298] usb 1-1.1: new full-speed USB device number 6 using xhci_hcd
[ 705.995790] usb 1-1.1: New USB device found, idVendor=1546, idProduct=01a8, bcdDevice= 3.01
[ 705.995807] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 705.995820] usb 1-1.1: Product: u-blox GNSS receiver
[ 705.995833] usb 1-1.1: Manufacturer: u-blox AG - www.u-blox.com
[ 706.000716] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
```

Vous verrez apparaître les lignes ci-dessus prouvant que le GPS a bien été détecté (u-blox GNSS receiver) et identifié. La commande lsusb vous affiche cette ligne :

```
Bus 001 Device 006: ID 1546:01a8 U-Blox AG [u-blox 8]
```

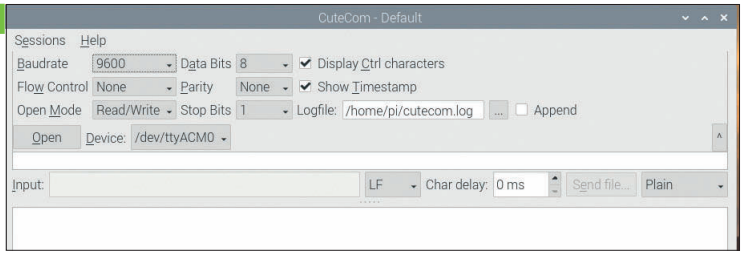
Qui confirme l'identification de l'appareil par le système.

Installer un terminal série

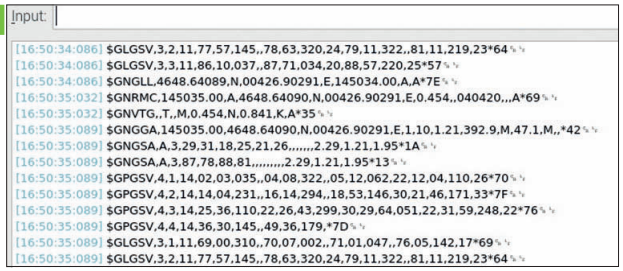
Un terminal série peut afficher les trames NMEA mais Raspbian n'en propose pas par défaut, il faut en installer un. En mode graphique **CuteCom** est une solution légère et assez conviviale. Dans un terminal tapez

```
sudo apt-get install cutecom
```

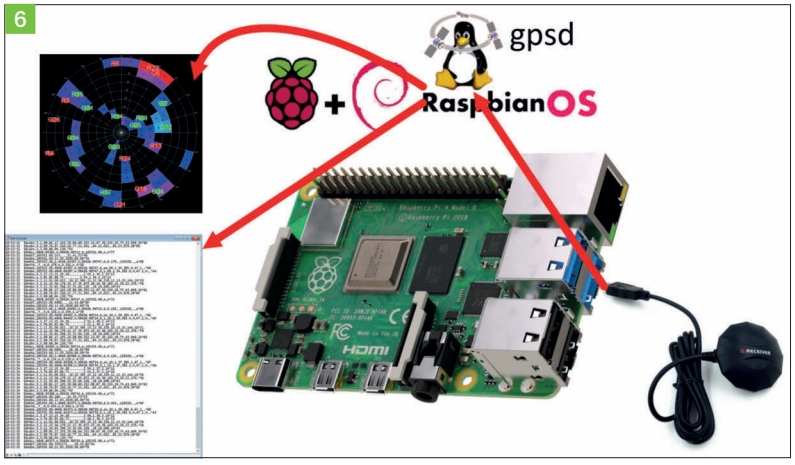
4



5



6



A la fin de l'installation, vous pouvez lancer CuteCom en tapant cutecom dans une console ou via le menu principal dans lequel une rubrique **Outils système** est apparue. Vous y trouverez une icône **CuteCom**. La fenêtre de CuteCom s'ouvre, cliquez sur **Settings** pour configurer la liaison série.

4 Dans la fenêtre **Settings** réglez la vitesse à 9600 bits/s et choisissez le **Device** correspondant à la connexion de votre GPS, ici */dev/ttyACM0*. Cliquez sur le bouton **Open** pour commencer à recevoir les trames NMEA.

La fenêtre du terminal va faire défiler les trames reçues. On peut voir (5) que les différents types de trames sont présents. Si les données n'apparaissent pas, n'allez pas plus loin. Il faut impérativement recevoir les trames NMEA avant de passer à la suite.

Installation de gpsd

Gpsd, c'est quoi ?

gpsd est un démon (ou daemon : un programme lancé automatiquement au démarrage du système et qui s'exécute en arrière-plan). gpsd reçoit les données d'un ou plusieurs récepteurs GPS ou AIS connectés à un ordinateur hôte par des ports série ou USB, fournissant des informations sur la position, le parcours et la vitesse des récepteurs connectés. Il peut être interrogé sur le **port TCP 2947** de l'ordinateur hôte. On voit sur 6 le GPS à droite. Les informations sont transmises à gpsd via le port USB. Deux consommateurs viennent lire les données GPS pour les afficher sous forme graphique (répartition des satellites dans le ciel) ou en mode texte (trames NMEA).

Installer gpsd

Dans un terminal saisissez la commande suivante :

```
sudo apt-get install gpsd gpsd-clients
```

Une fois le logiciel installé, on passe à la configuration. Vous pouvez faire une copie de sauvegarde du fichier avant de le modifier, par sécurité.

```
sudo nano /etc/default/gpsd
```

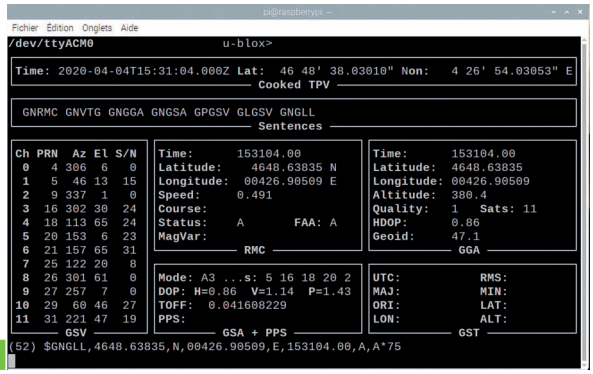
Modifiez les lignes du fichier suivant ces indications :

- **START_DAEMON="true"** signifie que vous voulez que **gpsd** se lance au démarrage comme daemon (service du système) ;
- **USBAUTO="true"** permet de connecter automatiquement les récepteurs GPS à **gpsd** quand on les connecte sur un port ;
- **GPSD_OPTIONS="-n"** si vous souhaitez que le GPS recherche les satellites à la mise en route pour synchroniser l'heure ;
- **DEVICES="/dev/ttyACM0** dépend de votre système. C'est le même nom que celui que vous avez trouvé avec **CuteCom** à l'étape précédente.

Sauvegardez les modifications avec CTRL X, puis rebootez Raspbian.

Tester gpsd

Parmi les clients de ***gpsd,*** nous avons installé ***gpsmon*** qui est un moniteur permettant de se connecter à ***gpsd*** et de vérifier que tout fonctionne bien, en mode texte. Tapez ***gpsmon*** dans un terminal en ligne de commande. Vous devriez obtenir une fenêtre présentant les informations GPS sous forme de texte (7).



7 Si vous avez un écran comme celui-ci tout va bien, vous pouvez continuer. Si l'écran est vide ou si vous avez des erreurs qui s'affichent il va falloir mettre tout ça d'aplomb avant de continuer.

Logiciel de navigation Navit

Navit, c'est quoi ?

Navit est un **logiciel libre** de navigation routière (ou de guidage routier) sous Linux et autres systèmes d'exploitation. Il est capable d'utiliser les données OSM (**OpenStreetMap**). Il est non seulement **compatible Linux** mais il est aussi utilisable sur un certain nombre d'appareils mobiles sous Android ou Windows. Ce projet utilise un fichier ".bin" généré à partir des données cartographiques d'OSM. Il peut ainsi effectuer un rendu de la carte en temps réel. Navit est capable d'effectuer des recherches par adresse, de vous guider en temps réel, de recalculer l'itinéraire en cas d'erreur. Consultez le site du projet [3] pour plus d'informations. (d'après OpenStreetMap [4]).

Navit n'est pas le logiciel de navigation le plus évolué, il a l'avantage d'exister sur le Raspberry Pi et d'être facile à installer. Des pages d'aide en ligne existent en français [7].

Installer Navit

Le logiciel Navit est disponible dans le dépôt Raspbian. Pour l'installer il suffira de saisir la commande suivante en ligne de commande :

```
sudo apt-get install navit
```

Pour exécuter Navit vous pouvez simplement taper navit en ligne de commande, vous retrouverez également une icône Navit dans la rubrique **Accessoires** du menu principal. Une fois lancé, déception ! On obtient un écran vide avec un point rouge qui indique la position du récepteur GPS… Et la carte n'apparaît pas. En fait, il va falloir télécharger la carte qui nous intéresse puis indiquer à Navit de l'utiliser. Attention, plus la zone que vous sélectionnerez sera grande, plus le fichier sera encombrant !

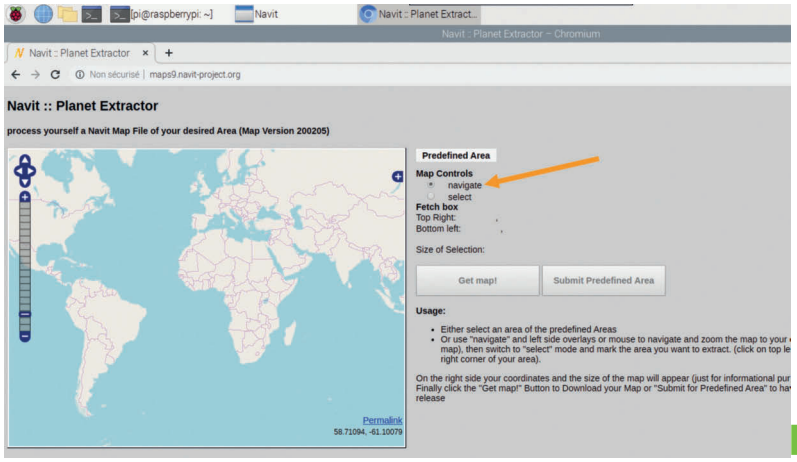
Télécharger la carte

Dans le navigateur web de Raspbian, rendez-vous sur le site de téléchargement de Navit [5].

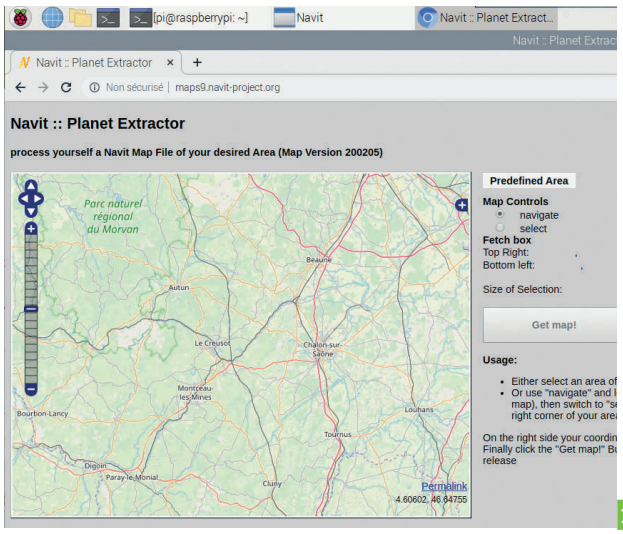
Le bouton **Navigate** est coché (9), il va permettre de choisir la zone qui vous intéresse. Utilisez **la molette de la souris** (Zoom) et le **clic gauche** (déplacement) pour choisir une zone.

Si vous souhaitez récupérer une zone déjà sélectionnée, passez par le bouton **Predifined Area**.

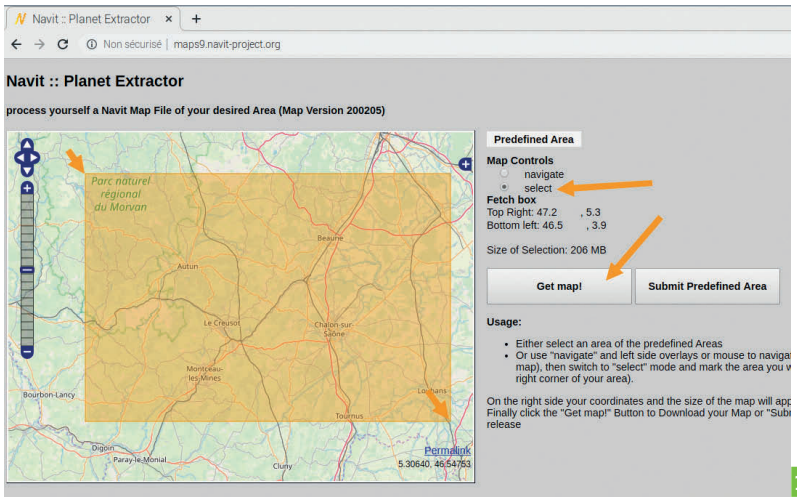
Lorsque vous êtes satisfait(e) de la zone affichée, cliquez sur le bouton **select**, situé sous le bouton **navigate** pour "couper" la partie de la carte que vous voulez utiliser. (10)



9



10



11

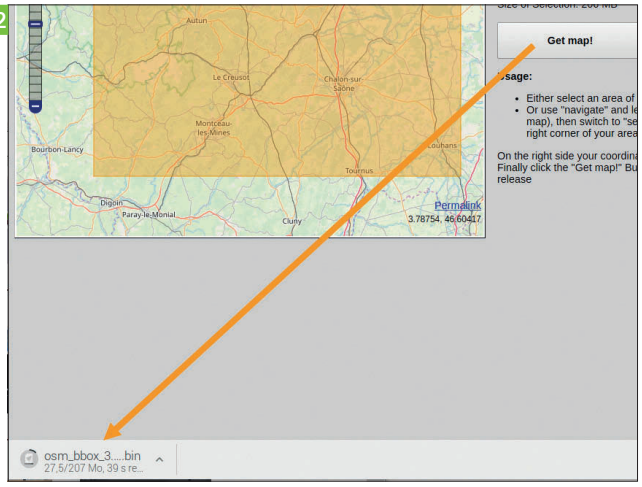
Avec la souris sélectionnez la zone à récupérer (clic gauche et tracer le rectangle). Quand la zone est définie, cliquez sur le bouton **Get map!** (11)

Le fichier ***osm_bbox_3….bin*** est téléchargé (voir en bas à gauche de la fenêtre). Attendez la fin du téléchargement. (12)

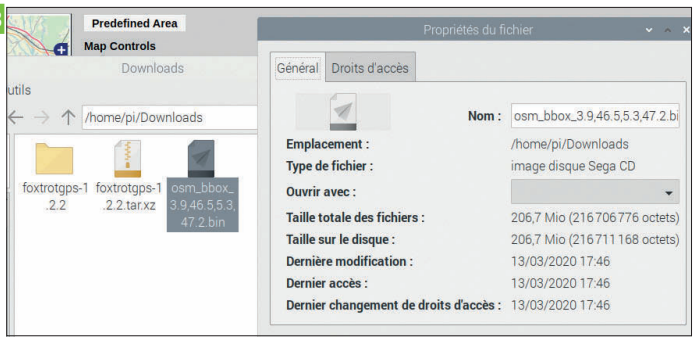
Le fichier est disponible dans le dossier **Downloads** (13).

Pour la zone que j'ai sélectionnée, il représente 207Mo environ. Il reste à indiquer à Navit que c'est cette carte qu'il doit utiliser. Avant

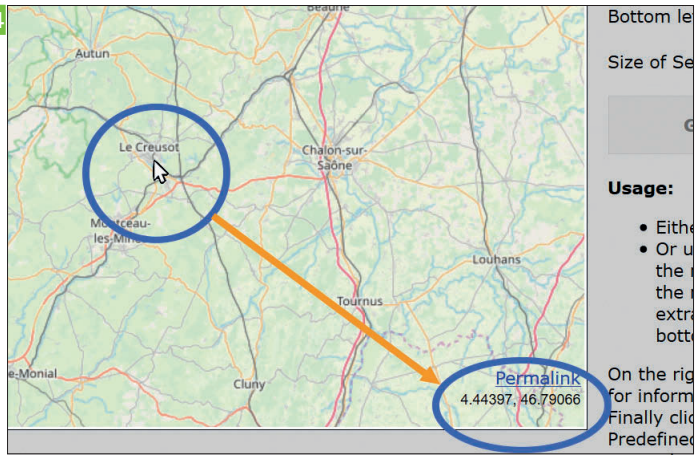
12



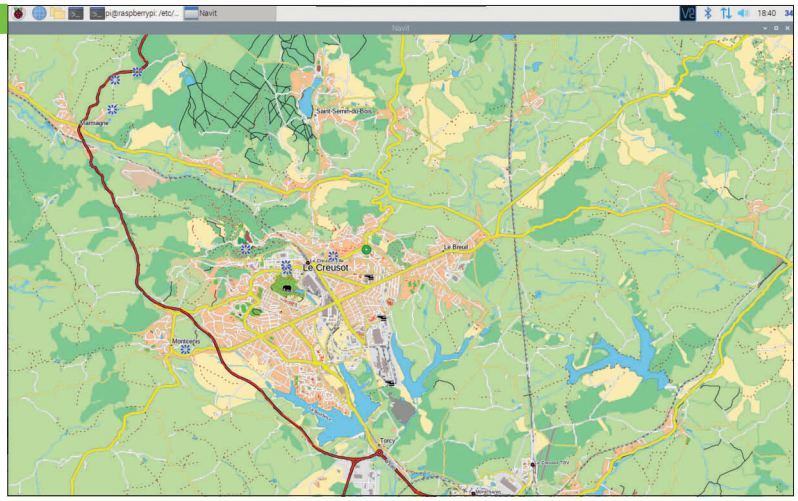
13



14



8



de continuer j'ai renommé la carte **creusot.bin** pour faciliter les choses.

Pour le centrage de la carte, placez la souris sur l'endroit que vous voulez voir apparaître au centre de la fenêtre GPS, et relevez les coordonnées en bas à droite. (14)

Sauvegardez le fichier ***/etc/navit.xml*** avant de le modifier avec sudo nano /etc/navit.xml

Recherchez la ligne **navit center** (CTRL W dans nano) Dans **navit.xml**, reportez les coordonnées du centre de la carte dans cette ligne :

```
<navit center="4.44397 46.79066" zoom="256" tracking="1" orientation="-1"
 recent_dest="250" drag_bitmap="0">
```

Ensuite recherchez la partie qui permet d'utiliser la carte openStreetMap et modifiez comme suit

```
<mapset enabled="no">
   <xi:include href="$NAVIT_SHAREDIR/maps/*.xml"/>
</mapset>
<!-- Mapset template for openstreetmaps -->
<mapset enabled="yes">
   <map type="binfile" enabled="yes" data="/home/pi/Downloads/creusot.bin"/>
</mapset>
```

Il n'y a plus qu'à relancer **Navit** et il affichera la carte que vous avez choisie. (8)

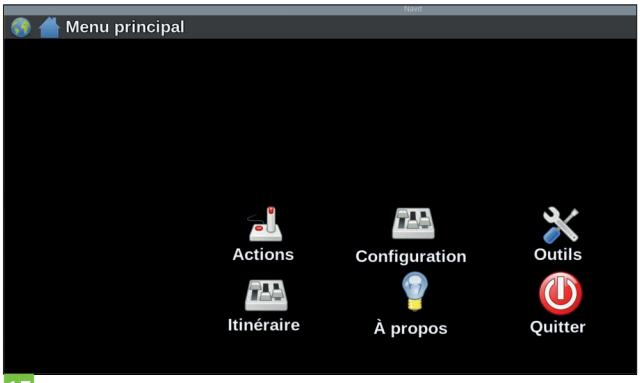
Comme prévu, la carte est centrée sur Le Creusot. Vous pouvez aussi changer d'autres paramètres, mais je vous laisse découvrir tout ça ainsi que l'utilisation de **Navit** avec **OpenStreetMap**.

Utiliser Navit 15

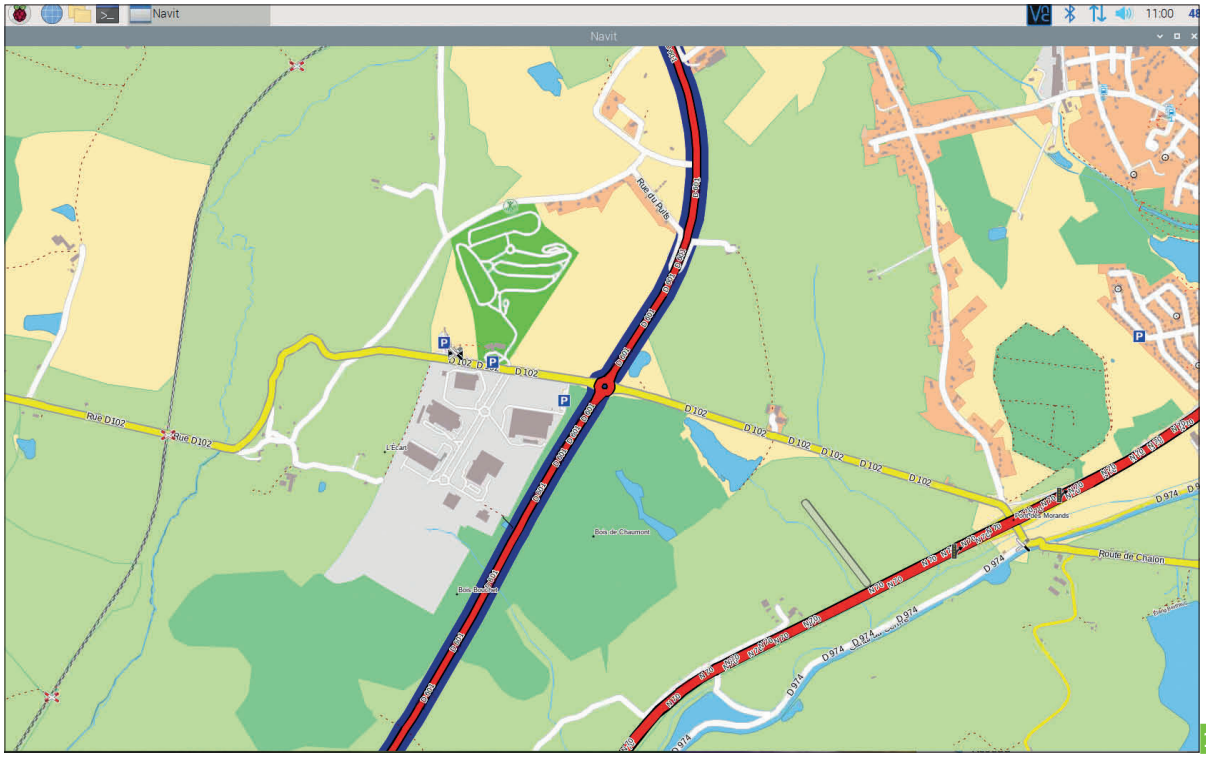
Dans **Navit** tapez sur la touche **Entrée**, vous accédez au menu principal de l'application. Vous obtiendrez des informations détaillées sur l'utilisation de Navit en consultant le wiki [en anglais] de l'application [6].

Actions vous permet de définir le point de départ (**Définir comme position**). En cliquant sur l'icône **Ville** vous pouvez saisir le nom de la ville au clavier ou sur un écran tactile. Le tiret (obligatoire pour les noms de villes composés) est disponible dans la partie **123** du clavier virtuel.

Saisissez ensuite l'adresse d'arrivée et entrez-la avec **Définir comme destination**. Navit calcule l'itinéraire (de quelques secondes à quelques dizaines de secondes), et l'affiche sous forme d'un trait bleu épais (16). Il vous appartiendra de suivre cet itinéraire. Si vous prenez une route différente, l'itinéraire sera recalculé.



15



16

L'icône **Itinéraire** permet de revenir à l'affichage complet du trajet, ou d'afficher une description de l'itinéraire en mode texte.
Navit utilise les cartes openStreetMap (OSM) et donne accès aux POI (points d'intérêts) figurant dans OSM, comme les parkings, stations essence, pharmacies etc.

Conclusion

La mise en œuvre d'un logiciel GPS sur le Raspberry Pi est relativement facile et les paquets sont disponibles sans avoir à compiler quoi que ce soit. Sans être aussi performant qu'un GPS du commerce, Navit est une solution GPS maker. On privilégiera un Raspberry Pi 4 si la vitesse de réaction est un critère important. Mais cette solution fonctionne également sur des Raspberry Pi plus anciens… avec un peu de patience.
Pour les Raspberry Pi isolés, non connectés à un réseau, le GPS peut se révéler intéressant pour fournir une heure précise (à quelques restrictions près) au système [9].
D'autres solutions existent, en particulier pour les marins. Vous en aurez un aperçu dans l'article [8] sur le blog framboise314.fr.

Sources

[1] https://fr.aliexpress.com/item/32951942245.html
[2] https://fr.wikipedia.org/wiki/NMEA_0183
[3] https://www.navit-project.org/
[4] https://wiki.openstreetmap.org/wiki/FR:Navit
[5] http://maps9.navit-project.org/
[6] https://navit.readthedocs.io/en/latest/
[7] https://sebsauvage.net/wiki/doku.php?id=navit
[8] https://www.framboise314.fr/un-gps-usb-pour-le-raspberry-pi/
[9] https://www.framboise314.fr/lheure-gps-sur-votre-raspberry-pi/

Jennifer Proust
Xebia

Un Mölkky connecté avec Node JS et du Bluetooth Low Energy

Si vous avez déjà joué au Mölkky, vous avez surement déjà prononcé ou entendu cette phrase ou un équivalent « On en est à combien ? ». Pour ceux d'entre vous qui ne savent pas ce qu'est le Mölkky, il s'agit d'un jeu de quilles finlandais dont le but est de marquer des points pour atteindre 50. Ceci en renversant les quilles avec un bâton lanceur. Généralement on est nombreux, c'est l'heure de l'apéritif et très rapidement personne ne sait où en est la partie.

« Et si quelque chose comptait les points pour nous ? »

J'ai donc fait un prototype avec des beacons dotés d'un accéléromètre, et on l'a mis à l'épreuve en novembre dernier pendant la Xebicon 19. On vous explique tout dans cet article.

Définir le proto

Les règles du jeu

Le principe du jeu est de faire tomber des quilles en bois à l'aide d'un bâton lanceur. Les quilles sont marquées de 1 à 12. La première équipe arrivant à totaliser exactement 50 points gagne la partie. Au début du jeu, les quilles sont positionnées comme sur le schéma suivant : 1

1

Pour renverser les quilles, le joueur de la première équipe jette le bâton lanceur sur les quilles.

Trois situations sont alors possibles :

- plusieurs quilles sont tombées : l'équipe gagne autant de points que de quilles renversées ;
- une seule quille est tombée : l'équipe gagne le nombre de points inscrits sur la quille ;
- aucune quille n'est tombée : le joueur réessaie dans la limite de 3 lancers ratés (cette règle diffère souvent).

On redresse ensuite les quilles à l'endroit où elles étaient tombées. Ainsi elles s'éparpillent au cours du jeu. C'est ensuite à l'équipe suivante de jouer. Si une équipe dépasse 50 points, son score redescend à 25. Il faut en effet atteindre exactement un total de 50, pas plus.

Gameplay cible versus objectif du proto

Pour notre proto, on acceptera que pour arriver au résultat idéal, il nous faudra itérer. On part donc sur un scénario, qui pourrait se dérouler ainsi, avec des étapes supplémentaires au scénario idéal :

Scenario idéal	Scenario proto
Le joueur lance le bâton sur les quilles	
	Un écran lui indique les quilles renversées et les points à marquer
	Le joueur valide ses points sur un device
Un écran met à jour son score	
C'est à l'équipe suivante de jouer	

Le Hardware

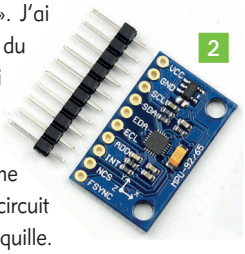
Quelle « thing » et quelle techno pour notre solution IOT

Gyroscope, accéléromètre : de quels capteurs avons-nous besoin ?

La première chose qui vient à l'esprit pour connaître la position d'une quille c'est « il nous faut un gyroscope dans la quille comme dans les smartphones ! ».

Après quelques recherches on comprend qu'en réalité un smartphone combine les données de plusieurs capteurs pour estimer sa position. Contrairement à ce qu'on imagine, notre smartphone n'a pas de gyroscope. Il a un gyromètre, un accéléromètre et un magnétomètre. Le smartphone les utilise pour obtenir la valeur absolue du tangage (mouvement de l'avant vers l'arrière) et du roulis (mouvement de droite à gauche). [https://sites.google.com/site/wikismartphone/techno/acclromtresgyroscopesmagntomtresgps]

Alors, j'ai cherché un objet doté de ces trois capteurs. En gros j'ai recherché sur Google « gyroscope sensors ». J'ai exploré un bon nombre de pistes. J'ai regardé du côté d'Arduino et de Raspberry. Je suis aussi souvent tombée sur des capteurs qui semblaient parfaits comme celui-ci : 2



2

Je voulais plus simple : hors de question de me mettre à faire des soudures pour brancher un circuit électronique à une batterie à l'intérieur d'une quille.

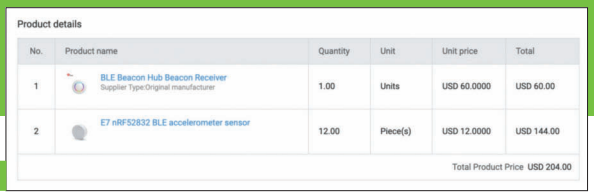
Des capteurs BLE et leurs contraintes

De fil en aiguille, j'ai découvert Espruino qui commençait à beaucoup me plaire. Ça parlait JavaScript et applications mobiles, pour la développeuse front que je suis, c'était plutôt séduisant. Alors j'ai cherché des sensors Bluetooth. J'ai découvert des frameworks qui donnaient envie : Evothings et Johnny-Five par exemple.

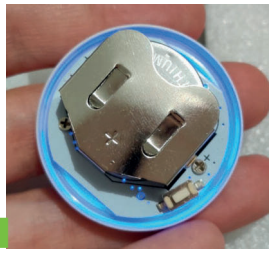
Ma recherche s'est de plus en plus affinée et puis, je me suis dit que des sensors BLE (Bluetooth Low Energy) étaient effectivement la clé : il nous fallait des balises avec un accéléromètre ! Comme c'est basse consommation, une pile dans notre petit objet serait parfaite. Plusieurs solutions s'offraient à nous : Kontact.io, Blue Net Beacon, Sensoro et surtout Estimote.

Et puis en discutant avec les collègues, un jour l'un d'entre eux me dit « tu vas avoir une limite : les spécs de Bluetooth, c'est pas plus de 7 devices connectés simultanément à un device ! »

Effectivement :



3



5

9

"A master BR/EDR Bluetooth device can communicate with a maximum of seven devices in a piconet (an ad-hoc computer network using Bluetooth technology), though not all devices reach this maximum." [https://en.wikipedia.org/wiki/Bluetooth]. Qu'à cela ne tienne, j'utiliserai une passerelle !

Une gateway et 12 beacons : c'est Minew !

En faisant toutes ces recherches, je finissais toujours par tomber sur des beacons très peu documentés, mais aussi bien moins chers, vendus sur Alibaba par l'entreprise Minew. J'ai donc décidé de commander et d'essayer : 3

Les beacons transmettent les données de l'accéléromètre à la gateway par Bluetooth. Celle-ci communique ensuite ces données à l'adresse d'un serveur (cloud, local ou hébergé) en HTTP ou en MQTT. Elle a donc besoin d'être connectée par WiFi (ou par Ethernet). 4

Des beacons aux positions

Configurer les beacons

Pour allumer un beacon, il faut l'ouvrir et presser 3 secondes le bouton on. Il se met alors à clignoter quelques secondes puis à émettre des données. À partir de ce moment, il est actif et joignable. Nous allons voir quelles sont ces données et comment les intercepter. 5

Pour le configurer, Minew a mis à disposition une application mobile téléchargeable (BeaconSET sur Google Play / BeaconSET). L'entreprise nous a également envoyé les sources de l'application native Android, qui peut servir de base pour un usage qui serait différent du nôtre.

Changer le mot de passe

Par défaut le mot de passe est minew123, il est vivement conseillé d'en choisir un nouveau. 6

Émettre les données de l'accéléromètre

Un beacon est conçu pour de multiples usages, notamment dans le marketing. Plusieurs « slots » sont alors configurables dans l'interface de configuration du beacon. Ici un slot en particulier nous intéresse c'est l'accéléromètre (ACC), que l'on souhaite actif en continu sans avoir besoin de le déclencher manuellement avec le bouton. Pour ce faire on positionne le trigger à motion. On supprime donc les autres, à l'exception de INFO, qui est utile au beacon en lui-même et on le configure. 7

La home de l'application nous permet alors de vérifier que le beacon transmet les valeurs de mouvement du beacon selon 3 axes orthogonaux : X (abscisse), Y (ordonnée) et surtout Z (cote). 8

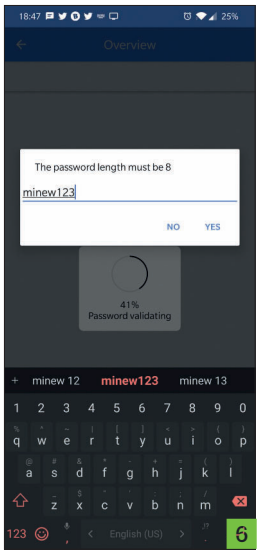
On effectue ces opérations sur chacun des 11 beacons restants.

Configurer la gateway 9

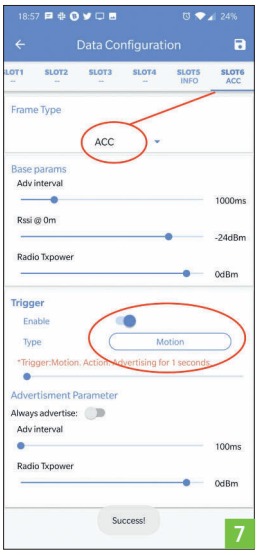
Réseau local et wifi

Une fois allumée, la gateway diffuse un réseau WiFi

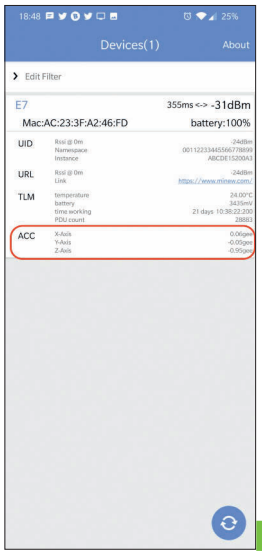
4



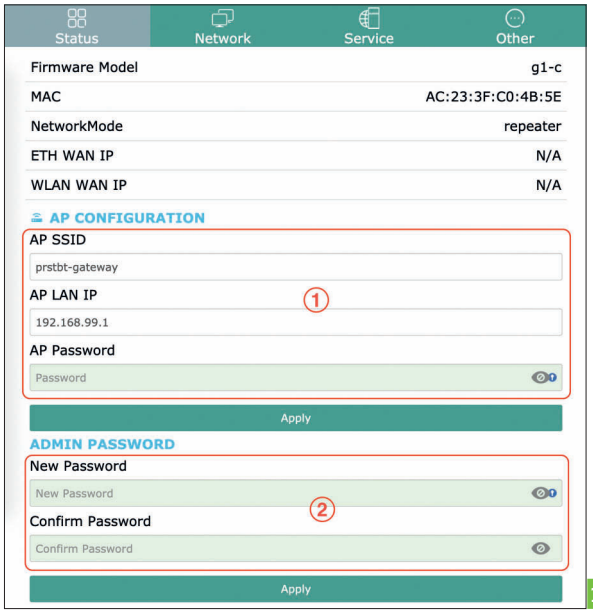
6



7



8



10

« GW-XXXXXXXXXXXX », il n'y a pas de mot de passe par défaut. Lorsqu'on est connecté à ce WiFi, l'interface se trouve à l'adresse http://192.168.99.1

La première chose à faire est de changer et sécuriser le SSID que diffuse la gateway. Il faut aussi mettre un mot de passe sur l'interface. 10

Ensuite notre gateway a besoin d'être elle-même reliée à Internet pour pouvoir communiquer à des services cloud par exemple. Pour cela, on peut la connecter avec une prise Ethernet ou configurer un réseau Internet domestique dans l'onglet Network : 11

Après redémarrage, on sait si le routeur est bien connecté à Internet lorsqu'une adresse IP lui a été attribuée : 12

Le fait d'avoir sélectionné le mode répétiteur sur la gateway, nous

permet d'utiliser la connexion Internet à laquelle elle est elle-même connectée, lorsqu'on se connecte à son propre réseau WiFi.

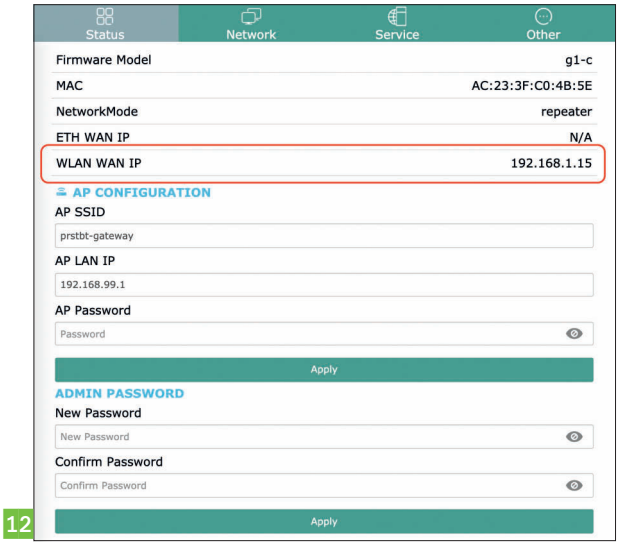
Réception et émission des données

A priori, notre gateway reçoit les données des beacons allumés et doit pouvoir les transmettre à n'importe quel service. Comment vérifier ?

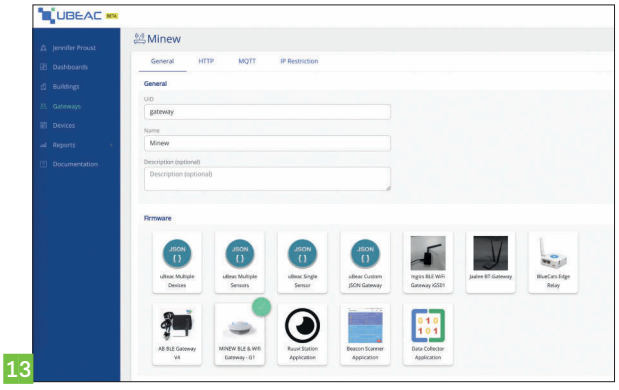
Dans un premier temps, utilisons la plateforme IOT uBeac, un service d'agrégation et de visualisation de données. Celle-ci nous permettra de vérifier que notre gateway transmet correctement ce qu'elle reçoit. Après avoir créé un compte, nous configurons une gateway et sélectionnons la nôtre : Minew. 13

L'onglet HTTP nous donne le endpoint à copier qui va nous servir à configurer le service sur l'interface de notre gateway.

11



12



13

Nous pouvons maintenant nous rendre sur l'interface de la gateway dans l'onglet Services et lui fournir le endpoint d'uBeac, sélectionnons également le format JSON : 14

Si on bouge nos beacons, on peut alors voir apparaître les requêtes en temps réel : 15

Et voir à quoi ressemble la data reçue pour un beacon : 16

Nous verrons plus tard comment interpréter ce JSON.

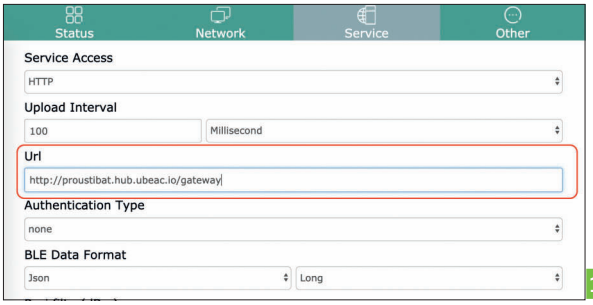
Le Software

Le matériel fonctionne. Il ne nous reste plus qu'à développer la partie software de notre Mölkky.

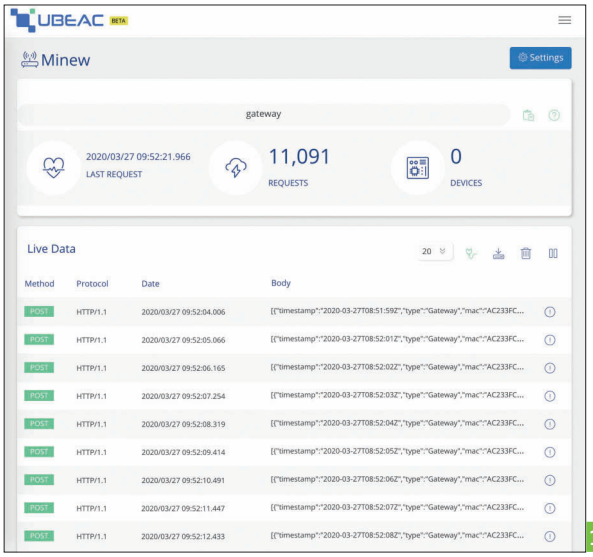
Créer notre propre endpoint pour recevoir les données

Mise en place d'un serveur Node

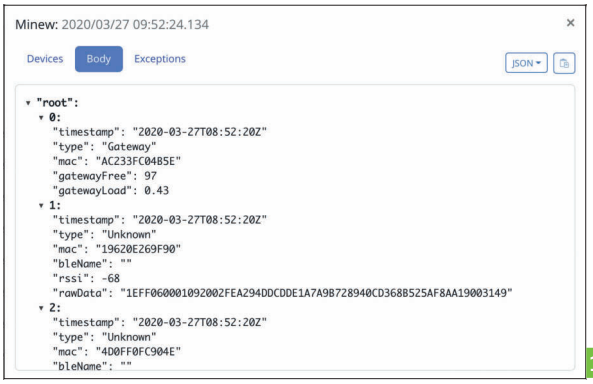
À la manière d'uBeac, nous créons un serveur qui, sur un endpoint en POST reçoit les données en continu pour les traiter. Pour cela



14



15



16

nous utilisons le framework Express. On crée une route /api/minew :

```
import express from 'express';

const router = express.Router();

router.post('/', async (req, res) => {
 const { body } = req;
 console.log({ body });
 res.end();
});

module.exports = router;
```

On peut déployer notre serveur sur un Raspberry ou chez un hébergeur. Notre endpoint sera donc http://192.168.99.103:8888/api/minew en local et nous pouvons le fournir à la gateway : 17

C'est parti, notre serveur reçoit alors beaucoup d'informations, on note qu'il reçoit bien les informations de la gateway en elle-même, mais aussi des informations de potentiels devices BLE autour :

```
POST /api/minew 200 1.563 ms - -
{ body:
   [ { timestamp: '2020-03-27T11:38:20Z',
       type: 'Gateway',
       mac: 'AC233FC04B5E',
       gatewayFree: 90,
       gatewayLoad: 1.25 },
     { timestamp: '2020-03-27T11:38:20Z',
       type: 'Unknown',
       mac: '02B9CFFD1023',
       bleName: '',
       rssi: -74,
       rawData:
        '1EFF060001092002243A028AD99690810B40E49E4719AC78626873A60F94C8' },
     { timestamp: '2020-03-27T11:38:20Z',
       type: 'Unknown',
       mac: '55E0D11D5561',
       bleName: '',
       rssi: -14,
       rawData: '09FFE0000125CA5D1F04' } ] }
```

Améliorons la configuration de notre gateway pour filtrer et recevoir uniquement les informations de nos sensors. Désactivons l'envoi des données de la gateway et filtrons par adresse MAC. Ici l'exemple avec un seul beacon : 18

À présent notre serveur ne reçoit qu'un objet vide si on ne bouge pas les beacons en question et sinon il reçoit les données des beacons qu'on a indiqués à la gateway :

```
POST /api/minew 200 0.427 ms - -
{ body:
   [ { timestamp: '2020-03-27T11:50:12Z',
       type: 'Unknown',
       mac: 'AC233FA246A2',
       bleName: '',
       rssi: -68,
       rawData: '0201060303E1FF1216E1FFA10364000000500EBA246A23F23AC' } ] }
```

Interprétation des données reçues

Pour l'instant ce qu'on reçoit ne sont que des données brutes dans la clé rawData du body. Il s'agit d'un format de diffusion de paquet. Pour en savoir plus, je vous conseille l'article « Understanding the different types of BLE Beacons ».

ReelyActive nous fournit par exemple un outil en ligne pour déchiffrer ces données à partir de la valeur de rawData et en sélectionnant le bon sensor : 19

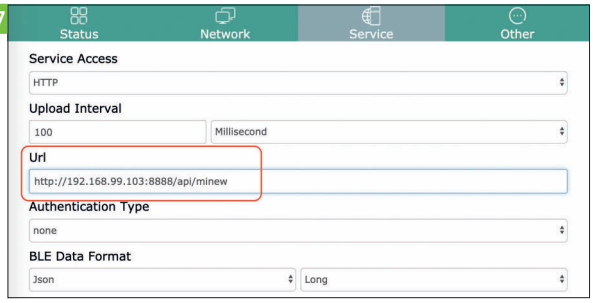
Et voici, nous voyons enfin apparaître les valeurs de nos accélérations : 20

Et en plus ReelyActive propose en open source une librairie qu'on va pouvoir utiliser : Advlib.

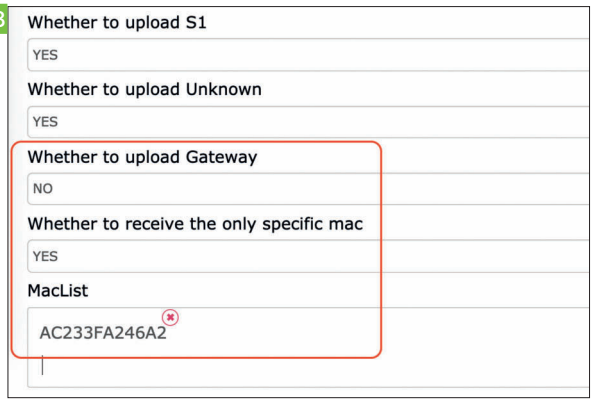
Transformons le log de notre route :

```
import advlib from 'advlib';
import express from 'express';
import isArray from 'lodash/isArray';

const router = express.Router();
```

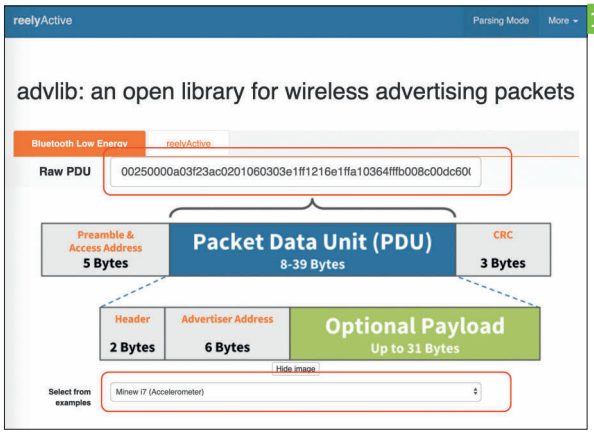
17



18



19



20

```
router.post('/', async (req, res) => {
 const { body } = req;
 isArray(body) && body.length > 0 && body.forEach(({ rawData }) => {
   rawData && console.log(advlib.ble.data.process(rawData));
 });
 res.end();
});

module.exports = router;
```

Résultat :

```
POST /api/minew 200 0.650 ms - -
{ flags: [ 'LE General Discoverable Mode', 'BR/EDR Not Supported' ],
  complete16BitUUIDs: 'ffe1',
  serviceData:
   { uuid: 'ffe1',
     data: 'a10364000a000000f0a246a23f23ac',
     minew:
      { frameType: 'a1',
        productModel: 3,
        batteryPercent: 100,
        accelerationX: 0.0390625,
        accelerationY: 0,
        accelerationZ: 0.9375,
        macAddress: 'ac:23:3f:a2:46:a2' } } }
```

Nettoyons les données inutiles en filtrant ce qu'on reçoit de façon à ne garder que batteryPercent et surtout mac et accelerationZ. Ici SKITTLES est une constante importée qui associe chaque adresse MAC à un numéro de quilles :

```
router.post('/', async (req, res) => {
 const { body } = req;
 // Look for data concerning skittles sensors
 if (body && isArray(body) && body.length > 0) {
   const skittles = body
     .filter((sensor) => Object
       .keys(SKITTLES)
       .includes(sensor.mac));

   // Look for skittles with raw data and transform raw data
   if (skittles.length > 0) {
     const skittlesInfo = skittles
       .filter((skittle) => !isUndefined(skittle.rawData)
           && !isNull(skittle.rawData)
           && skittle.rawData.length > 0)
       .map(({ mac, rawData }) => ({
         mac,
         ...advlib.ble.data.process(rawData),
       }))
       .filter((skittle) => hasIn(skittle, 'serviceData.uuid')
           && hasIn(skittle, 'serviceData.data')
           && hasIn(skittle, 'serviceData.minew'))
       .map(({ serviceData: { uuid, data, minew }, ...rest }) => ({
         uuid,
         data,
         ...minew,
         ...rest,
       }))
       .map((data) => omit(data, [
         'data',
         'frameType',
         'productModel',
         'accelerationX',
         'accelerationY',
         'macAddress',
         'flags',
         'complete16BitUUIDs',
         'uuid',
       ]));

     console.log({ skittlesInfo });
   } else {
     debug('No skittle sensors', skittles);
   }
 } else {
   debug('No Body or Empty body');
 }
 res.end();
});
```

Nous obtenons donc un tableau d'objets pour chaque beacon reçu :

```
{ skittlesInfo:
   [ { batteryPercent: 100,
       accelerationZ: 0.9296875,
       mac: 'AC233FA246A2' } ] }
```

De l'accélération à la position

L'accéléromètre permet de savoir dans quelle direction le beacon se déplace, il ne détecte pas une position mais une accélération sur l'un des trois axes X, Y, Z.

Donc ce que chaque beacon va nous envoyer, ce sont ses changements de vitesses et ses mouvements de translation.

Ce n'est pas suffisant pour ce que nous voulons : savoir si le beacon est horizontal ou vertical ! Heureusement, l'accéléromètre permet aussi de détecter la force de gravité générée par la Terre. C'est en fait ce détail qui va nous être très utile, beaucoup plus que tout le reste. En effet, par chance, la valeur de l'accélération sur l'axe Z est toujours proche de 1 lorsque le beacon ne bouge pas sur cet axe, c'est sa force de gravité. Ainsi nous pouvons déduire qu'il est en position debout. Nous voulons obtenir un objet avec la structure suivante pour chaque réception :

```
{
    XXXXXXXXXXX: { // adresse mac
        battery: 100,
        position: 'KNOCKED_OVER', // 'KNOCKED_OVER' ou 'UPRIGHT'
        z: 0.9296875,
        value: 12 // numero de la quille
    },
    ...
}
```

Si on stocke sur le serveur la dernière position de chaque quille (quasiment en temps réel selon l'intervalle d'envoi réglé sur la

gateway), alors nous avons tout ce qu'il faut pour créer une partie de Mölkky.

Considérons donc un objet global sur notre serveur qu'on appelle lastState. Donnons une marge d'erreur via POSITION_LEVEL d'une valeur de 0.8 et ajoutons à la suite de notre code :

```
skittlesInfo.forEach(({ mac, accelerationZ: z, batteryPercent: battery }) => {
  lastState[mac] = {
    ...lastState[mac],
    ...(battery && { battery }),
    ...(z && {
      position: z <= POSITION_LEVEL ? 'UPRIGHT' : 'KNOCKED_OVER',
      z,
    }),
    value: SKITTLES[mac],
  };
});
```

Du socket pour le temps réel

Pour qu'un client puisse se connecter à notre serveur et avoir les données en temps réel, nous utilisons Socket.io pour émettre notre set de quilles avec leurs positions :

```
const io = req.app.get('socketio');
io.emit('UPDATE', lastState);
```

Développer l'application

Le serveur

L'API pour la gestion du jeu

Elle fournit 4 endpoints accessibles en POST :

- */start* : pour démarrer une partie ;
- */score* : pour envoyer le score de l'équipe qui vient de jouer ;
- */miss* : si une équipe a joué mais n'a pas touché de quilles ;
- */reset* : pour recommencer la partie ou pour réinitialiser la partie quand elle est terminée.

Endpoint	Body	Réponse
/api/molkky/start	`{ "teams": [ "cat", "dog" ], "playingTeam": "dog" }`	`{ "scores": { "cat": { "score": 0, "left": 50 }, "dog": { "score": 0, "left": 50 } }, "currentTurn": { "isPlaying": "dog", "remain": 3 } }`
/api/molkky/score	`{ "team": "cat", "points": 10 }`	`{ "scores": { "cat": { "score": 0, "left": 50 }, "dog": { "score": 10, "left": 40 } }, "currentTurn": { "isPlaying": "cat", "remain": 3, "wining": "dog" } }`
/api/molkky/miss	`{ "team": "dog" }`	`{ "scores": { "cat": { "score": 0, "left": 50 }, "dog": { "score": 6, "left": 44 } }, "currentTurn": { "isPlaying": "cat", "remain": 2 } }`
/api/molkky/reset	-	`{ "scores": null, "currentTurn": null }`

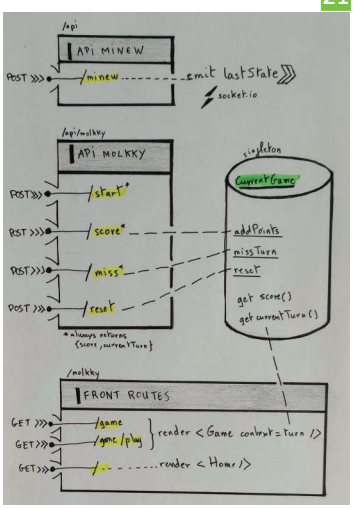
Nous avons documenté ces endpoints et leurs réponses potentielles sur https://documenter.getpostman.com/view/1117131/SW11Vxdm?version=latest.

Un singleton pour la partie en cours

Un singleton CurrentGame est instancié lorsque le endpoint /start est appelé. C'est cette instance qui contient toutes les infos de la partie en cours.

Les routes pour servir le front

Enfin les routes */molkky/game* et */molkky/play* rendent le composant front Game en lui fournissant le currentGame en contexte. 21



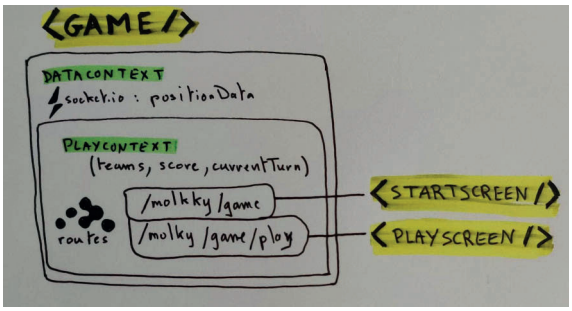
21

Le client pour les joueurs

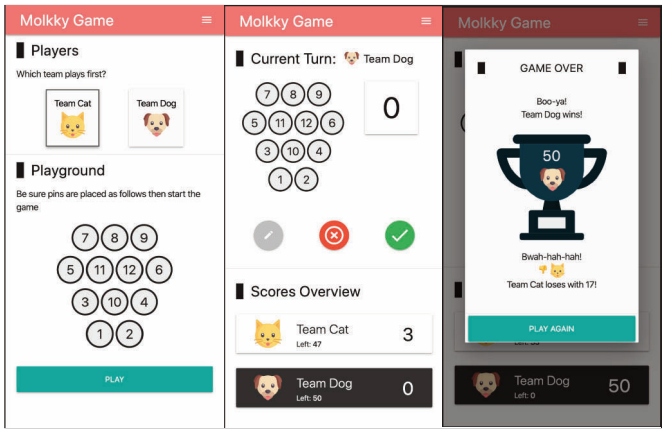
Ici nous avons réalisé l'application front avec React. Le composant Game (que nous avons vu plus tôt, en server side rendering) :

- Un DataContext qui contient les données des quilles, récupérées via Socket.io ;
- Un PlayContext qui contient les données propres à la partie ;
- Selon la route :

- Un composant StartScreen qui permet à l'utilisateur de sélectionner la team qui va jouer et vérifier que toutes les quilles soient détectées et en position debout avant de pouvoir commencer la partie.
- sUn composant PlayScreen qui contient les composants du jeu : affichage des quilles, score, bouton de validation du score, modale pour modifier éventuellement le score.



Screenshots d'une partie :



Le Mölkky en situation réelle

Intégrer nos beacons à nos quilles

Les essais

Avant d'installer nos beacons dans des quilles en bois, nous avons fait des tests. Les images se passent de commentaires. 22

Les quilles en bois

Un collègue doué en bricolage + un set de Mölkky en bois = le tour est joué ! 23

Bilan

Il faut améliorer la fiabilité

À l'usage, sur un stand de la Xebicon, nous avons observé que parfois les beacons ne captaient pas le fait d'avoir été renversés ou relevés. Est-ce dû au choc ? Faut-il améliorer les réglages des beacons ?

De plus, le fait d'avoir un système où l'envoi de POST est multiple en permanence, n'est pas des plus performants. On aurait préféré une solution qui va chercher l'information sur le device bluetooth, plutôt que celui-ci n'émette en permanence. D'autres technologies sont peut-être plus appropriées. Aussi faudrait-il approfondir et surveiller ce que donne l'API Web Bluetooth.

En conclusion, on peut dire que pour des beacons, la fiabilité est :

- raisonnable car nous avons pu faire plusieurs parties ;
- insuffisante car nous avons dû plusieurs fois corriger la détection.

Il faut itérer pour développer des features

Nous avons implémenté un leaderboard pour la journée sur le stand de la Xebicon pour enregistrer les scores via Firebase.

Beaucoup d'améliorations et de nouvelles features sont envisageables :

- settings des règles de jeu ;
- prendre en compte des joueurs individuels et plus de 2 teams ;
- jouer avec les statistiques de jeu ;
- etc.

Si l'envie vous prend de vouloir tester, les sources sont disponibles sur le projet https://github.com/proustibat/connected-molkky, avec un menu qui permet de simuler des beacons si vous n'en avez pas.

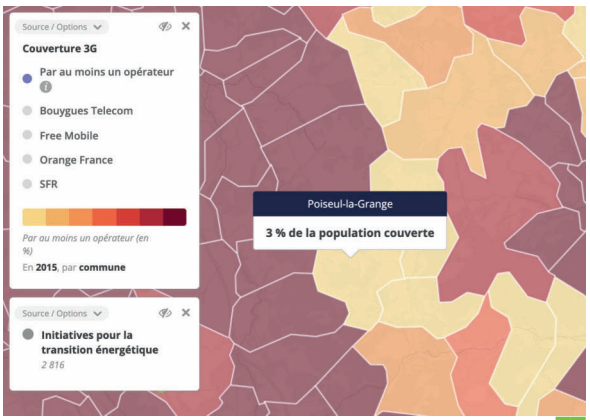
Le mot de la fin

On ne va pas se mentir : notre Mölkky est loin d'être prêt à être commercialisé. Se balader sur les terrains de jeu avec une gateway branchée qui a besoin d'Internet c'est pas l'idéal. Ce n'est pas facilement portable et si vous êtes à Poiseul-la-Grange c'est même un peu difficile de trouver de la 3G. 24

Chez Publicis Sapient Engineering, on aime les projets impossibles et surtout on aime chercher des solutions. La Xebicon, c'est l'occasion de partager la recherche qu'on fait grâce à ces projets insensés qui nous passionnent. Et qui sait, peut-être vous donner envie de participer pour les améliorer ?


22


23



24

Sébastien Colas
Je suis formateur en informatique depuis bientôt 20 ans. Au cours de ma carrière j'ai pu dispenser des cours sur de nombreuses technologies : Serveur d'applications JavaEE, SOA, Services Web, Linux, Virtualisation, API led Connectivty et Application Network. Sans oublier les langages : Java, PHP, Python, JavaScript... Je suis aussi auteur pour « Linux Pratique » et « Hackable » autour des sujets Raspberry Pi, Arduino, IoT, électronique digitale et bien sûr Linux. http://colas.sebastien.free.fr/

Découverte du M5Stack

Dans cet article, nous allons prendre en main un M5Stack ainsi qu'un M5StickC. Après avoir décrit pourquoi ces deux modèles permettent d'entrer en douceur dans le monde du développement sur IoT (Internet of Thing, en français l'Internet des objets), nous aborderons les outils de développement : UIFlow l'outil de développement graphique ainsi que l'Arduino IDE nous permettant de développer en C.

Avertissement : un bug peut apparaître pour piloter la batmobile depuis M5StickC. L'auteur a remonté le problème au développeur de la librairie. En cours de résolution.

1. Le matériel

Avant toute chose, tout ce qu'il faut savoir à propos du M5Stack se trouve sur le site officiel : https://m5stack.com/

Un des gros avantages des matériels M5Stack est qu'il n'y a pas besoin de sortir sont fer à souder pour commencer à faire du prototypage. En effet de base le M5Stack possède un bon nombre de composants embarqués. 1

Regardons ce que nous propose le M5Stack (version Basic) :

- Un haut-parleur 1 Watt ainsi que sa puce d'amplification
- Un ESP32 comprenant un processeur 240MHz dual core, 520Ko de SRAM, Wi-Fi et Bluetooth
- Un bouton (rouge) pour réinitialisation/marche/arrêt
- Un lecteur Micro SD (attention, limité à 16 Go)
- Une batterie LiPo ainsi qu'une puce IP5306 pour surveiller le niveau de charge
- Un connecteur USB type C permettant de charger les programmes ainsi que la recharge la batterie
- Un écran LCD couleur 320x240
- Trois boutons en façade

La première chose à savoir c'est comment allumer et éteindre les différents modèles :

- Pour allumer le M5Stack il faut appuyer une fois sur le bouton rouge. Pour l'arrêt il faudra appuyez deux fois sur le bouton.
- Pour allumer le M5StickC il faut appuyer une fois sur le bouton situé à côté du connecteur USB. Pour l'arrêt il faudra appuyer 6 secondes sur le bouton.

Les prix

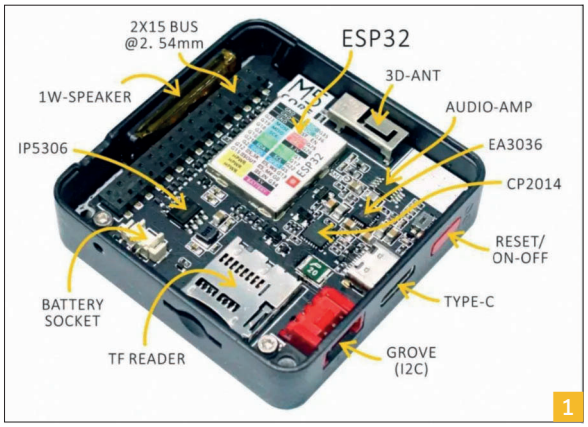
- M5Stack basic : 35€
- M5Stack Fire : 65€
- M5StickC : 25€

Où acheter votre M5Stack ?

- Sur certains sites d'électronique : https://www.gotronic.fr/
- Sur le site officiel : https://m5stack.com

2. Idéal pour l'IoT

Le M5Stack nous permet très facilement de rajouter des modules électroniques en fonction de nos besoins. Ses modules s'enfichent par le dessous du composant, on dit qu'il est « stackable ».



1

Voici une sélection de modules pour M5Stack :

Module M5Stack	Description
USB	Le M5Stack devient Hôte USB, on peut donc y connecter n'importe quel périphérique USB
GSM	Ajouter au M5Stack les fonctions d'un téléphone mobile
GPS	Ajouter un GPS au M5Stack
Battery	Ajouter une batterie supplémentaire pour gagner en autonomie. Plusieurs modules Battery peuvent être utilisés en même temps pour augmenter encore plus l'autonomie du M5Stack
SERVO	Ce module permet au M5Stack de piloter jusqu'à 12 servo moteurs
Stepmotor	Piloter 3 moteurs pas à pas à l'aide de votre M5Stack
LEGO+	Piloter 4 moteurs Lego Mindstorm à l'aide de votre M5Stack

Le M5StickC quant à lui n'est pas « stackable » mais on peut aussi étendre ses fonctionnalités en rajoutant un module appelé « Hat ». De nombreux modules « Hat » existent. Un connecteur 8 broches est prévu pour connecter ces modules « Hat ».

Voici une sélection de Hat pour M5StickC :

M5StickC Hat	Description
PowerC	Le M5StickC gagne en autonomie grâce à ce module permettant d'ajouter 2 batteries rechargeables 16340
18650C	Le M5StickC gagne en autonomie grâce à ce module contenant une batterie rechargeable 18650
JoyC	Transformer le M5StickC en Joystick Wireless grâce à ce module embarquant une batterie rechargeable 16340

M5StickC Hat	Description
RoverC	Transformer le M5StickC en robot roulant grâce à ce module 18350 embarquant une batterie rechargeable 18350
8Servos	Piloter 8 servo moteurs à l'aide de ce module, il faudra rajouter une batterie 16340 pour alimenter les servo moteurs.
Joystick	Ajouter un joystick à votre M5StickC.
Yun	Hat en forme de nuage idéal pour transformer votre M5StickC en station météo. Capteurs inclus : température, humidité, pression.
Thermal Camera	Votre M5StickC devient une caméra thermique de faible résolution 32x24

Comme on peut le constater, le M5StickC seul ne possède pas une grande autonomie (Quelques heures seulement en fonction des utilisations), c'est pourquoi de nombreux Hat proposent l'ajout de batteries. Les modèles M5Stack et M5StickC possèdent au moins un port d'extension compatible Grove permettant de connecter des composants électroniques via I2C, Uart et GPIO.

3. Programmation avec UIFlow

3.1 Installation de UIFlow via M5Burner

A partir du site de téléchargement https://m5stack.com/pages/download téléchargeons le M5Burner. Attention sur certains OS, il faut installer un driver USB avant de pouvoir lancer le M5Burner. (CP2104 Driver sur la page de téléchargement). Une fois lancé, le Burner nous permet de sélectionner le programme à installer. Sélectionnons la version 1.4.5.1 de UIFlow qui est compatible M5Stack et M5StickC. Les versions plus récentes supportent uniquement le M5Stack. Une fois téléchargé nous pouvons envoyer notre programme à notre M5Stack en cliquant sur Burn. N'oubliez pas auparavant de spécifier vos information Wifi. 3

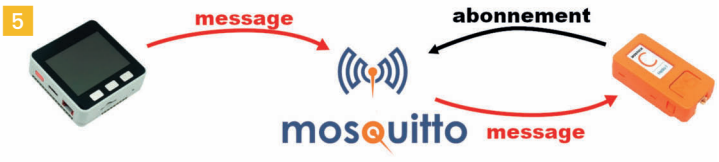
Une fois installé le programme se lance automatiquement. On nous propose un rapide mode d'emploi ainsi que la découverte des fonctionnalités du M5Stack (haut-parleur, microphone, gyroscope, barre RGB…)

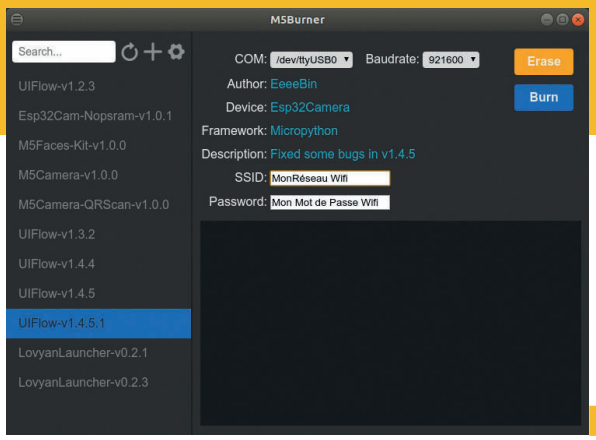
3.2 Utilisation de UIFlow 4

L'étape suivante consiste à connecter l'interface de développement UIFlow à notre M5Stack. Il y a deux possibilités pour accéder à UIFlow : en ligne grâce à L'URL https://flow.m5stack.com/ ou bien grâce à l'IDE disponible sur le site de téléchargement. L'outil nous demande « l'Api key » pour pouvoir déployer notre application sur notre M5Stack. Pour obtenir cette information, il faut redémarrer le M5Stack en appuyant sur le bouton rouge puis rapidement appuyer sur le premier bouton « upload ».

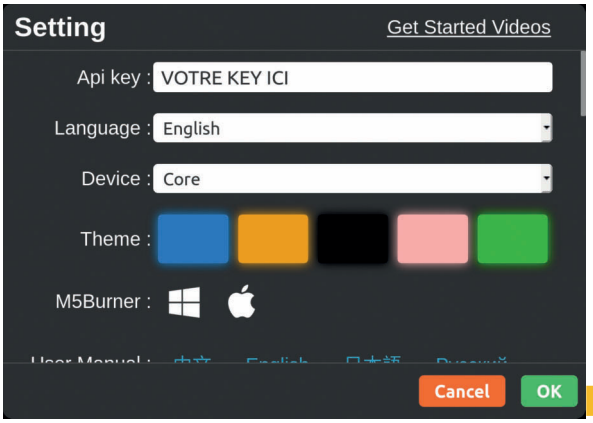
L'étape suivante consiste à connecter l'interface de développement UIFlow à notre M5Stack. Il y a deux possibilités pour accéder à UIFlow : en ligne grâce à L'URL https://flow.m5stack.com/ ou bien grâce à l'IDE disponible sur le site de téléchargement. L'outil nous demande « l'Api key » pour pouvoir déployer notre application sur notre M5Stack. Si vous avez testé les applications préinstallées pour retourner sur le menu affichant « l'API key », il faut redémarrer le

5





3



4

M5Stack en appuyant sur le bouton rouge puis rapidement appuyer sur le bouton du milieu (celui situé sous les deux points).

Pour le moment UIFlow ne permet pas de développer en Français, nous allons donc utiliser l'interface en anglais.

3.3 Programmation « Communication en MQTT »

Je vous propose 2 programmes appelés Flows. Un pour le M5Stack et un pour le M5StickC. Le but principal de ces Flows est de communiquer en asynchrone avec un gestionnaire de messages via le protocole MQTT (Message Queuing Telemetry Transport).

Ce protocole, fréquemment utilisé dans l'IoT, permet la diffusion d'information, on parle de communication en mode éditeur/abonné. (En anglais : publish/subscribe ou encore Pub/Sub). Nous pouvons utiliser l'implémentation du groupe Apache nommée MosQuiTTo. Apache met à notre disposition un serveur à l'adresse suivante : mqtt.eclipse.org:1883. Sur notre schéma le M5StickC s'est abonné au message. Le M5Stack envoie un message qui est diffusé par le serveur MQTT aux abonnés, en l'occurrence notre M5StickC. Il s'agit d'un schéma simplifié. Comme vous ne disposez peut-être pas de plusieurs M5Stack, l'émetteur du message est aussi un abonné. Il recevra donc son propre message. 5

Voici le code pour notre M5Stack :

```
Setup
Set screen brightness 20
set          client id "Programmez_2"
server       "mqtt.eclipse.org"
port         1883
user         ""
password     ""
keepalive    300
mqtt start
Speaker.volume 0.1
```

6

```
mqtt subscribe "/topic/M5Stack" with topic_data
 Label        message  show  get topic_data
 Label        batterie  show  get Battery Level
 play tone    Hight A   for 1/2 beat
 Http Request
 Method       GET
 URL          "http://worldtimeapi.org/api/timezone/cet"
 Headers
 Data
 Success      Label     time  show  get key "datetime" in map loads json Get Data
 Fail         Label     time  show  "Erreur"

Button        A  wasPressed
 publish topic  "/topic/M5Stack"   msg   "M5Stack Button A"

Button        B  wasPressed
 publish topic  "/topic/M5Stack"   msg   "M5Stack Button B"

Button        C  wasPressed
 publish topic  "/topic/M5Stack"   msg   "M5Stack Button C"
```

Voici le code pour notre M5StickC :

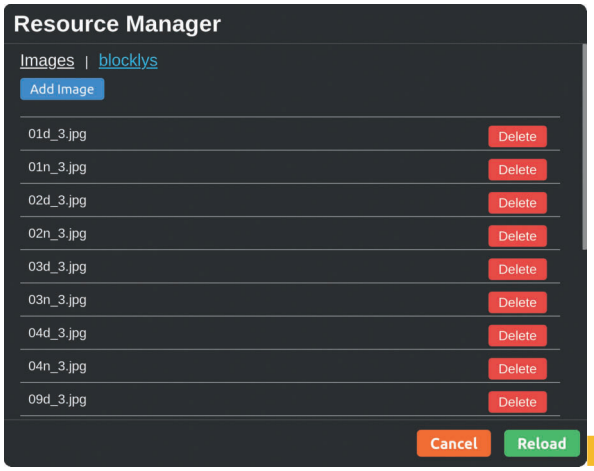
```
Setup
Set screen brightness 20
set client id   "Programmez_1"
server          "mqtt.eclipse.org"
port            1883
user            ""
password        ""
keepalive       300
mqtt start

mqtt subscribe "/topic/M5Stack" with topic_data
 Label        message  show  get topic_data

Button        A  wasPressed
 publish topic "/topic/M5Stack" msg   "M5StickC Button A"

Button        B  wasPressed
 publish topic "/topic/M5Stack" msg   "M5StickC Button B"
```

3.4 Installer un serveur Mosquitto

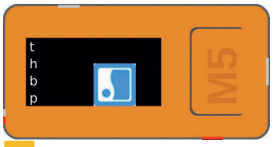
Dans les Flows précédents, nous avons utilisé le serveur MQTT proposé par Apache (MosQiTTo). Nous pouvons nous aussi installer notre propre serveur MosQiTTo sur un serveur Raspberry Pi en tapant la commande :

```
sudo apt-get install mosquitto
```

Ensuite il nous suffira de remplacer dans notre code : mqtt.eclipse.org par l'adresse IP de notre Raspberry Pi. Attention UIFlow ne reconnaît pas le protocole de découverte Bonjour. Donc impossible de prendre pour adresse : raspberry.local



7



8

3.5 Programmation « Météo avec Hat YUN »

Dans ce deuxième programme UIFlow nous allons utiliser le M5StickC et le Hat YUN. Ce composant est clairement adapté au développement d'une application météorologique. Nous allons donc développer le Flow suivant :

- Connection au Wifi ;
- Récupération toutes les 5 minutes des prévisions météo grâce à OpenWeatherMap ;
- Affichage d'une image réprésentant la prévision météo dans les heures à venir ;
- Eclairage du Hat pour renforcer l'image (par exemple si un grand soleil est annoncé nous éclairerons notre hat en jaune) ;
- Affichage toutes les secondes des informations fournies par le Hat : température, humidité, luminosité, pression atmosphérique.

Pour pouvoir utiliser le service OpenWeatherMap, nous devons obtenir une clef d'accès APPID à l'adresse suivante : https://home.openweathermap.org/users/sign_up (dans le code suivant, il faudra remplacer XXX par votre clef). Nous devons aussi récupérer le code de notre ville grâce à la liste que l'on peut télécharger ici : http://bulk.openweathermap.org/sample/city.list.json.gz (dans le code suivant il faudra remplacer YYY par le code ville).

Il faut ensuite charger les images météo dans notre M5StickC par l'intermédiaire du « Resource Manager ». Le bouton se trouve en haut à droite de l'interface 6. Ensuite, nous pouvons envoyer notre image au format jpg ou bpm 7. Ici nous allons choisir une image de soleil de 80x80 au format jpg. Toutes les options JPEG ne sont pas supportées, désactivons donc toutes les options lors de la création du fichier.

Nous pouvons placer notre image : image0 et nos labels : t,h,b,p (resp : température, humidité, brighness : luminosité , pression). 8

Il ne nous reste plus qu'à écrire notre programme :

```
Setup
wifi connect (log in lcd true)
Set Screen backgroundColor « black »
repeat while true
do
 Http Request
 Method GET
```

```
URL http://api.openweathermap.org/data/2.5/forecast?id=XXX&APPID=YYY&units=metric

Headers
Data
Success
 set data to Get Data
 set list to make list from text data with delimiter « , »
 set icon in list get #36
 set list to make list from text icon with delimiter « " »
 set icon in list list get #4
 if icon = « 01d »
 do
  set image0 image 01d.jpg
  set hat all RGB color « yellow »
 if icon = « 01n »
 do
  set image0 image 01n.jpg
  set hat all RGB color « black »
…
Fail
repeat 300 times
do
 Label p show « P » + Get hat yun Pressure
 Label h show « H » + Get hat yun humidity
 Label t show « P » + Get hat yun temperature
 Label b show « P » + Get hat yun Brightness
 Wait 1s
```

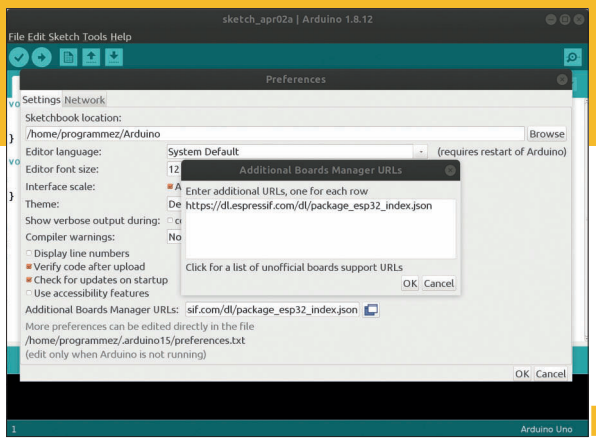
Le code a été tronqué pour ne montrer que les conditions météo 01d et 01n (respectivement beau temps de jour, beau temps de nuit). Il faudra bien sûr que nous codions tous les cas. La liste des valeurs possibles se trouve ici : https://openweathermap.org/weather-conditions.
OpenWeatherMap nous retourne des information JSON, malheureusement faute de documentation le block de lecture JSON est inutilisable pour le moment. Nous avons donc dû biaisé en créant une liste avec comme délimiteur la virgule pour récupérer le code de condition météo.

4. Programmation avec Arduino IDE

La majeure partie des composants IoT supportent le langage C. Un gros avantage des M5Stack c'est qu'ils sont compatibles avec Arduino. Avant de pouvoir programmer nous allons donc devoir configurer notre Arduino IDE.

4.1 Installation de Arduino IDE

L'outil de développement est disponible en téléchargement à l'URL suivante : https://www.arduino.cc/en/Main/Software .



9

9 Les M5Stack intègre un ESP32. Il nous faut donc ajouter le support de l'ESP32 à notre IDE. Cliquons sur File / Preferences. Dans le champ « Additional Boards Manager URLs » il faut saisir : https://dl.espressif.com/dl/package_esp32_index.json (voir capture d'écran)
Il faut maintenant activer le support de l'ESP32. Cliquons sur Tools / Board / Board Manager... Puis saisir ESP32 et installer le composant.
Il ne nous reste plus qu'à faire les derniers réglages :
- déclarer le « Board » à utiliser (sur la capture d'écran M5Stack-Core-ESP32) ;
- la vitesse de communication « Upload speed » (1500000)
- spécifier le « Port » de communication (par exemple /dev/ttyUSB0).

4.2 Installation des Bibliothèques

Nous allons maintenant télécharger les bibliothèques dont nous avons besoin. Tout d'abord les bibliothèques M5Stack et M5StickC (voir capture d'écran).
Cliquons sur Sktech / Include library / Manage Librairies… Sélectionnons M5Stack et M5StickC.

4.3 Programmation « cliqueur bluetooth »

Je vous propose dans ce premier projet en C de transformer notre M5StickC en « cliqueur bluetooth » à utiliser lors de nos futures présentations.
Le principe de fonctionnement est donc le suivant :
- Connexion Bluetooth, notre « cliqueur » sera vu comme un clavier ;
- Lorsque l'on clique sur le bouton A, envoyer le fait qu'on appuie sur la flèche du bas (pour faire défiler notre présentation) et afficher un triangle vert sur l'écran LCD ;
- Lorsque l'on clique sur le bouton B, envoyer le fait qu'on appuie sur la flèche du haut (pour faire défiler notre présentation) et afficher un triangle rouge sur l'écran LCD.

Pour commencer téléchargeons une bibliothèque pour simuler notre clavier BlueTooth : https://github.com/T-vK/ESP32-BLE-Keyboard
Ajoutons la bibliothèque: sktech / Include library / Add .zip Librairy…
Ci-dessous le code de notre programme :

```
include <BleKeyboard.h>
#include <M5StickC.h>
/* #include <M5Stack.h> */

BleKeyboard bleKeyboard("Clicker Bluetooth","Programmez",100);

void setup() {
 M5.begin();
 Serial.begin(115200);
 Serial.println("Starting BLE work!");
 bleKeyboard.begin();
 M5.Axp.ScreenBreath(10);
/*  M5.Lcd.setBrightness(100); */
}

boolean redrawA = true;
boolean redrawB = true;
const int triangles[2][6] = {{20,70,60,70,40,30},{20,80,60,80,40,120}};

void draw_triangle(boolean pressed, int t,int color, boolean *redraw)
{
 if (pressed)
 {
  M5.Lcd.fillTriangle(triangles[t][0],triangles[t][1],triangles[t][2],
                      triangles[t][3],triangles[t][4],triangles[t][5],color);
```

```
 }
 else
 {
  if (*redraw)
  {
   M5.Lcd.fillTriangle(triangles[t][0],triangles[t][1],triangles[t][2],
               triangles[t][3],triangles[t][4],triangles[t][5],BLACK);
   M5.Lcd.drawTriangle(triangles[t][0],triangles[t][1],triangles[t][2],
               triangles[t][3],triangles[t][4],triangles[t][5],color);
   *redraw=false;
  }
 }
}
void loop() {
 draw_triangle(M5.BtnA.read(),0,GREEN,&redrawA);
 draw_triangle(M5.BtnB.read(),1,RED ,&redrawB);

 if(bleKeyboard.isConnected())
 {
  if (M5.BtnA.wasPressed())
  {
   Serial.println("Sending KEY_DOWN_ARROW");
   bleKeyboard.write(KEY_DOWN_ARROW);
   redrawA = true;
  }
  if (M5.BtnB.wasPressed())
  {
   Serial.println("Sending KEY_UP_ARROW");
   bleKeyboard.write(KEY_UP_ARROW);
   redrawB = true;
  }
 }
 delay(100);
}
```

Intéressons-nous aux fonctionnalités relatives à notre M5Stack :

- #include <M5StickC.h> ; nous déclarons l'utilisation de la bibliothèque ;
- M5.begin(); initialisation de notre M5StickC ;
- M5.Axp.ScreenBreath(10); spécifions la luminosité de notre afficheur LCD. Attention cette commande est spécifique au M5StickC, pour un M5Stack il faudra utiliser la commande M5.Lcd.setBrightness(100);
- M5.Lcd.fillTriangle(...); M5.Lcd.drawTriangle(...) ; fonctions permettant d'afficher des triangles sur l'écran LCD.
- M5.BtnA.wasPressed(); détectons que le bouton a été pressé.

> Note de l'auteur : Après une recherche sur Internet, il semble que je ne sois pas le premier à avoir pensé à transformer le M5Stack en « cliqueur bluetooth »
> https://programresource.net/2020/04/09/3244.html

4.4 Programmation « Pilotage de la Batmobile Lego »

Je vous propose dans ce second projet en C le pilotage de la Batmobile Lego à l'aide de notre M5StickC auquel on aura ajouté le Hat Joystick.
Le principe de fonctionnement est donc le suivant :

- Connexion Bluetooth, notre M5StickC va piloter le Hub Lego ;
- Récupération de l'état du joystick : x,y,bouton ;
- Pilotage des moteurs attachés sur les ports A et B ;
- Si le bouton est appuyé changement de la couleur de la led Lego.

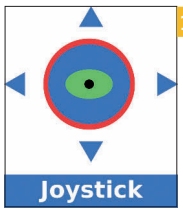
10 Nous allons maintenant télécharger une bibliothèque supplémentaire : Legoino (voir capture d'écran).
Cliquons sur Sktech / Include library / Manage Librairies… Sélectionnons Legoino.

11 Premier problème : le Joystick Hat ne semble pas être pas de très bonne qualité. Comme on peut le voir sur le schéma nous allons éviter de prendre les valeurs lorsque le Joystick est en butée (zone rouge) car la valeur retournée est systématiquement -128. De même lorsque le Joystick est au repos les valeurs en x et y ne sont pas toujours à zéro. Nous allons donc définir une zone morte (zone où il ne se passera rien) avec comme intervale $-40 < x < +40$ et $-20 < y < 20$. Sur le schéma, il s'agit de la zone en vert. Il nous restera donc toute une zone de travail, en bleu sur le schéma. Dans cette zone nous activerons les moteurs du Lego en mode tout ou rien avec une vitesse de 80 sur un maximum de 100.
Second problème : les bibliothèques Legoino et M5StickC ne sont pas compatibles. En effet les couleurs sont définies dans ces deux bibliothèques avec les mêmes noms, ce qui génère un problème de compilation. Nous allons donc modifier la déclaration des couleurs dans la bibliothèque Legoino. Pour cela, modifions la définition des couleurs dans le fichier Lpf2Hub.h se trouvant dans le répertoire Arduino/libraries/Legoino/src/ .

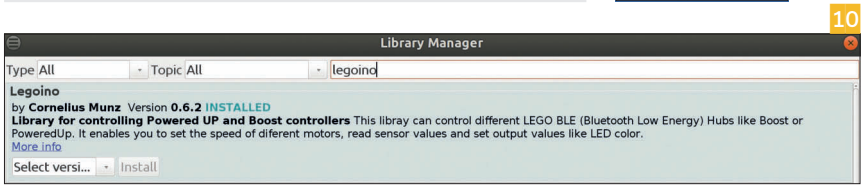
```
typedef enum Color
{
 LEGO_BLACK = 0,
 LEGO_PINK = 1,
 LEGO_PURPLE = 2,
 LEGO_BLUE = 3,
 LEGO_LIGHTBLUE = 4,
 LEGO_CYAN = 5,
 LEGO_GREEN = 6,
 LEGO_YELLOW = 7,
 LEGO_ORANGE = 8,
 LEGO_RED = 9,
 LEGO_WHITE = 10,
 LEGO_NONE = 255
};
```

Voici maintenant le code de notre programme :

```
#include <M5StickC.h>
#include <PoweredUpHub.h>
#include <Wire.h>
#define JOY_ADDR 0x38

int8_t x_data,y_data,button_data;

PoweredUpHub myHub;
PoweredUpHub::Port _portA = PoweredUpHub::Port::A;
PoweredUpHub::Port _portB = PoweredUpHub::Port::B;

boolean powerup_ok  = false;
int   motor_speed  = 80;
```



11



10

```
int   dead_zone_x  = 50;
int   dead_zone_y  = 20;
int   bad_value    = -128;

void setup() {
 M5.begin();
 Wire.begin(0, 26, 100000);
 myHub.init();
}

void loop() {
 powerup_ok = myHub.isConnected();
 Wire.beginTransmission(JOY_ADDR);
 Wire.write(0x02);
 Wire.endTransmission();
 Wire.requestFrom(JOY_ADDR, 3);

 if (!powerup_ok)
 {
  if (myHub.isConnecting()) {
   myHub.connectHub();
  }
 }
 else
  if (Wire.available()) {
   x_data = Wire.read();
   y_data = Wire.read();
   button_data = !Wire.read();

   if (button_data) {
     myHub.setLedColor(LEGO_RED);

   }
   else {
     myHub.setLedColor(LEGO_GREEN);
   }

   if (x_data != bad_value)
    if (x_data > dead_zone_x)
    {
     myHub.setMotorSpeed(_portA, -motor_speed);
     myHub.setMotorSpeed(_portB, -motor_speed);
    }
    else
    if (x_data < -dead_zone_x)
    {
     myHub.setMotorSpeed(_portA, motor_speed);
     myHub.setMotorSpeed(_portB, motor_speed);
    }
    else
     if (y_data != bad_value)
      if (y_data > dead_zone_y)
      {
       myHub.setMotorSpeed(_portA, motor_speed);
       myHub.setMotorSpeed(_portB, -motor_speed);
      }
      else
      if (y_data < -dead_zone_y)
      {
       myHub.setMotorSpeed(_portA, -motor_speed);
       myHub.setMotorSpeed(_portB, motor_speed);
      }
      else
      {
       myHub.setMotorSpeed(_portA, 0);
       myHub.setMotorSpeed(_portB, 0);
      }
  }
 delay(50);
}
```

Regardons de plus près les fonctionnalités intéressantes :

- #define JOY_ADDR 0x38 ; notre Hat Joystick communique en I2C dont l'adresse est 0x38 ;
- PoweredUpHub myHub; notre BatMobile est pilotée par un Hub PoweredUp que nous déclarons ici :
- PoweredUpHub::Port _portA = PoweredUpHub::Port::A; spécifions les port PoweredUp que nous voulons utiliser. Nous utiliserons les ports A et B pour piloter les moteurs.
- myHub.init(); initialisons notre Hub ;
- Wire.beginTransmission(JOY_ADDR); Wire.write(0x02); Wire.endTransmission(); ces 3 fonctions ont pour but d'acquérir l'état du JoyStick en I2C.
- Wire.requestFrom(JOY_ADDR, 3); demandons l'état sur JoyStick codé sur 3 octets : la position en x, la position en y, l'état du bouton ;
- x_data = Wire.read(); y_data = Wire.read(); button_data = !Wire.read(); récupérons les informations. L'état du bouton est à true ;
- myHub.setMotorSpeed(_portA, motor_speed); cette fonction permet de modifier la vitesse attachée au moteur 1. Attention pour faire avancer notre BatMobile les moteurs doivent tourner dans un sens différent.

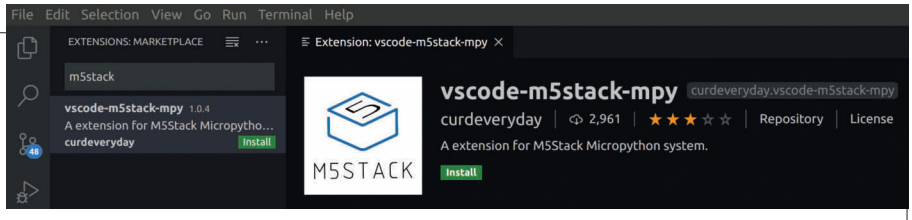
5 Conclusion

J'espère que cet article vous aura donner envie de programmer sur ces formidables composants IoT que sont les M5Stack. N'hésitez pas à créer vos propres projets et à les partager avec la communauté.

Pour aller plus loin

1 Développement MicropPython

Il est aussi possible de développer en MicroPython. Pour cela il existe 2 possibilités. Soit en utilisant UIFlow, soit en utilisant Visual Studio Code. Toute la documentation est disponible ci-dessous :
https://github.com/m5stack/UIFlow-Code



2 Développement avec PlatformIO

PlatformIO propose aussi un environnement de développement basé sur Visual Studio Code. Toute la documentation est disponible sur le site : https://github.com/m5stack/M5Stack-platformio"

Jon Mikel Inza
Référent IoT et Azure chez Exakis-Nelite
De formation technique, mais avec un parcours varié, je travaille aujourd'hui en tant que Référent IoT et Azure. Mon rôle principal est d'aider nos clients à construire des solutions cohérentes et optimisées par rapport à leurs besoins.

IoT : comment créer son IoT, le configurer et l'exploiter

Partie 2

L'IoT est un sujet très à la mode ces dernières années. Il est généralement connu du grand public par le CIoT — Consumer IoT — (domotique, wearables, etc.) ce qui ne représente qu'une petite partie de l'univers IoT. Ce dernier intègre différents domaines avec des latitudes très importantes (business, technologie, fonctionnel) et devient le vecteur d'une vraie transformation digitale (industrie, santé, énergie/environnement, agriculture, automobile, enseignement, smart cities, smart buildings, transports, logistique, etc.).

Réalisation

Maintenant que nous savons sur quel socle technique nous allons travailler, passons à l'étape de réalisation (voir n°241).

Comme mentionné précédemment, la solution proposée dans l'article sera construite autour d'IoT Central. Ce dernier est une solution SaaS qui, d'un point de vue macro, couvre les fonctionnalités suivantes :

- Gestion des devices (activation, statuts, gérer une partie des Twin properties, lancer des Opérations de différents types, etc).
- Gestion de template de devices, qui permet de construire des visuels par défaut sur la base de propriétés prédéfinies ;
- IoT Plug and Play ready ;
- Persistance des données ;
- Export des données ;
- Mise en place des déclencheurs sur la base de messages de certains types ;
- Mise en place des alertes ;
- Gestion des accès.

Cet article n'est pas un tutoriel IoT Central, décrivant chacune de ses fonctionnalités. Nous nous concentrerons uniquement sur les étapes nécessaires pour implémenter notre solution.

Étapes

Ci-dessous, la liste des étapes principales que nous allons devoir couvrir :

1 - Créer une instance d'IoT Central et ajouter une application ;
2 - Configurer le device (Wifi) ;
3 - Connecter le device ;
4 - Envoyer des informations ;
5 - Contrôler la réception des informations ;
6 - Mettre en place les actions déclenchées par la réception des messages (les builds Azure DevOps) ;
7 - Contrôler le déclenchement des actions.

Exécution

Création d'une instance d'IoT Central

URL du portail : https:// aka.ms/iotcentral 15 16

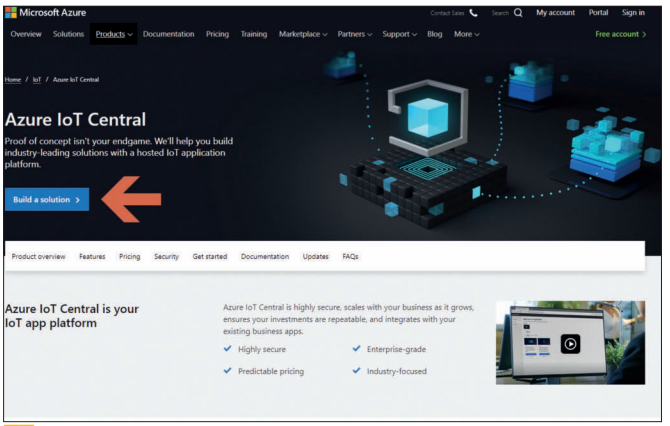
Une fois arrivés au portail, nous pouvons créer une application IoT Central. 17

Comme vous pouvez le constater, la plateforme inclut déjà une série de templates répondant à différents verticaux du marché. Les templates incluent des définitions de devices avec des propriétés prédéfinies et des visuels également prédéfinis.

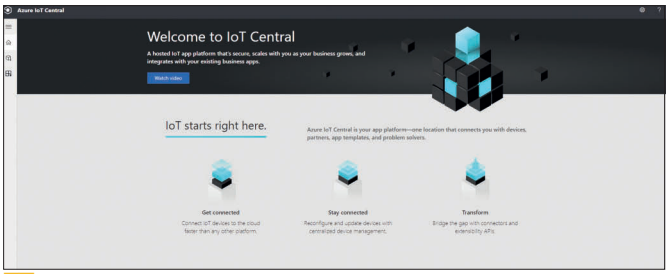
Dans notre cas, nous allons partir sur une solution vide : Custom app. 18

L'écran de création demande quelques informations basiques :

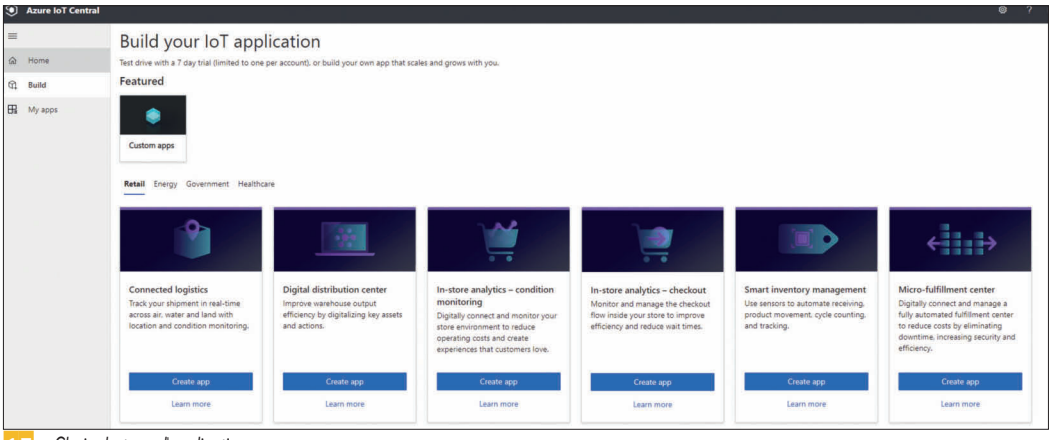
- Le nom de l'application ;
- L'URL que nous souhaitons attribuer à l'application ;
- Type de template ;
- Niveau de facturation.



15 - *IoT Central*



16 - *Nouvelle application IoT Central*



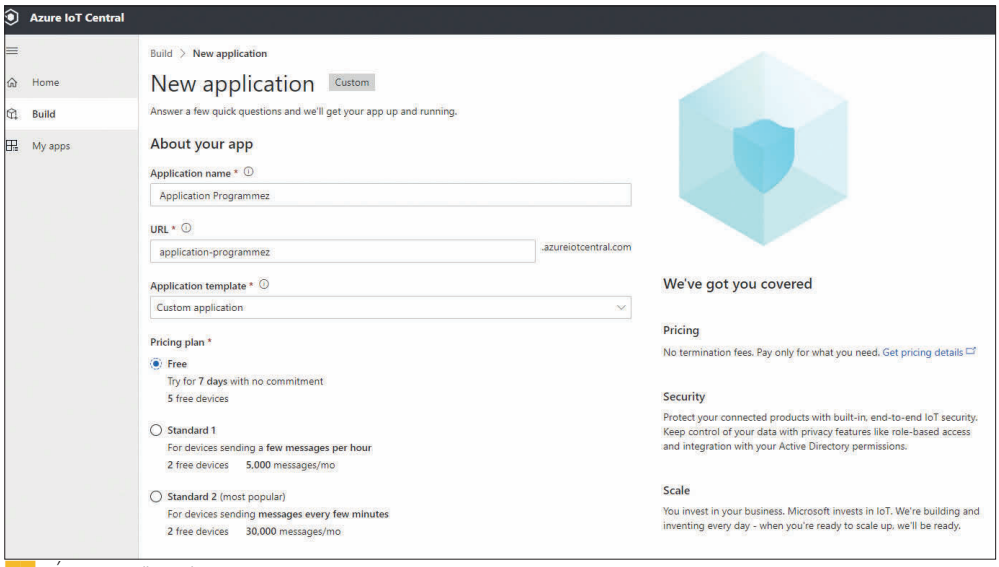
17 - *Choix du type d'application*

Il existe un niveau de facturation totalement gratuit qui permet d'utiliser toutes les fonctionnalités d'IoT Central pendant 7 jours (par application créée).
Si vous souhaitez garder votre application, vous devrez choisir l'une des offres Standard. Pour cela, il sera nécessaire de l'associer à un abonnement Azure.
Chacune des offres Standard permet de bénéficier de 2 devices de manière totalement gratuite.
C'est plutôt pratique pour faire une petite solution ou ses tests dans son coin.
Une fois le formulaire complété, la solution est rapidement créée.
Comme vous pourrez le constater, il s'agit d'une solution réellement opérationnelle et exploitable en production.

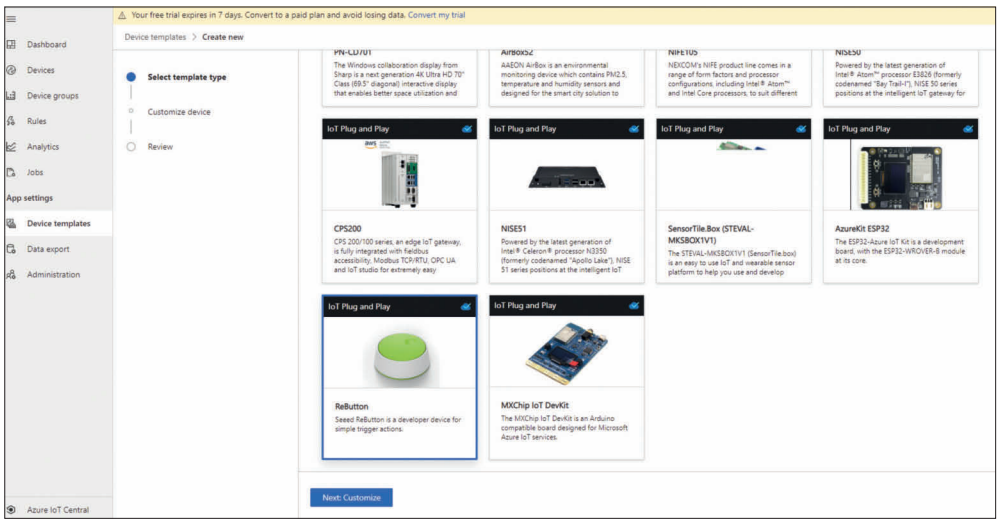
Note importante :
L'application qui vient d'être créée est dimensionnée pour pouvoir gérer plusieurs milliers de devices. En ce qui concerne l'administrateur de l'application, il n'aura rien à faire pour pouvoir gérer cette volumétrie. C'est déjà pris en compte.

Importer le template ReButton

Lors de cette étape, nous allons importer le template IoT Central du device que nous allons utiliser : un ReButton. 19
ReButton est un bouton IoT intégrant une connectivité wifi.
Il permet d'envoyer des messages IoT différents en fonction du type de click (simple, double, triple, long). Sa conception lui permet de fonctionner sur des piles et son coût est abordable.
Le template peut être importé directement depuis le portail IoT Central (cf. screenshot ci-dessous). 20



18 - *Écran nouvelle application*



20 - *ReButton template*

L'utilisation de templates n'est pas obligatoire, mais permet d'industrialiser certaines actions (ex : définir propriétés pour les devices, définir les visuels sur la base de ces propriétés, etc.).

Note : *le device est catalogué comme « IoT Plug and Play ready ».*
Cet article ne décrit pas cette notion, mais si elle vous intéresse et que vous souhaitez l'exploiter dans le cadre du projet créé dans cet article, pensez à mettre à jour la version du firmware du device.
Le configuration sera légèrement différente avec la mise à jour PnP.

Créer un device dans l'application

Nous pouvons maintenant créer un device sur la base du template importé. 21
IoT Central offre la possibilité de simuler des devices. Ces devices se comportent comme un objet réel et permettent d'envoyer des messages sur la base des propriétés définies dans le template.
Dans notre cas, nous allons utiliser un objet réel.
Une fois le device créé, nous devons garder les éléments de configuration suivants :

- ID scope,
- Device ID,
- Primary Key.

Ils sont accessibles en cliquant sur le bouton « Connect » : 22

Configurer le device

Après la création et la configuration de la partie applicative, nous devons passer à la partie device.
Les actions à faire sont :

- Configurer la connexion Wifi du device ;
- Configurer la chaîne de connexion du device.

Configurer la connexion Wifi

Appuyez le bouton jusqu'à ce que la LED devienne blanche (10 secondes environ). Ceci démarre le bouton en mode AP

19 - *ReButton, de Seeed*

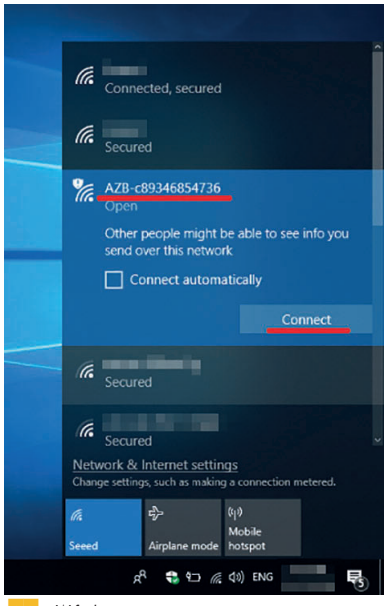
(Access Point) et nous permettra de nous connecter au device.
En relâchant le bouton, la LED se met à clignoter, ce qui confirme que le bouton est en mode AP (attention, ce mode consomme plus d'énergie que le mode de fonctionnement normal. Evitez d'oublier de laisser un bouton en mode AP).
Dès que la LED clignote, vous devriez pouvoir voir le bouton depuis le wifi de votre PC. Le nom du device est du type AZB—[code], le code figurant dans le device. 23
Une fois connecté, ouvrez un navigateur web et saisissez l'adresse suivante :
http://192.168.0.1
Vous devriez arriver vers la page d'accueil de votre device. 24

Note : *les écrans peuvent être légèrement différents en fonction de la version du firmware du device.*

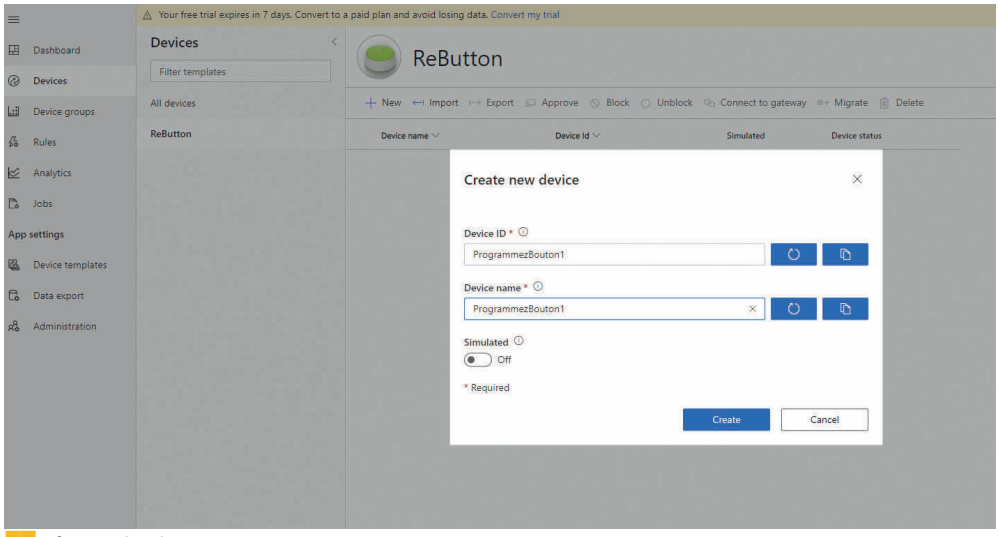
Cliquez sur la configuration Wifi et saisissez le SSID ainsi que le mot de passe de votre wifi. 25

Configuration de la partie IoT Central

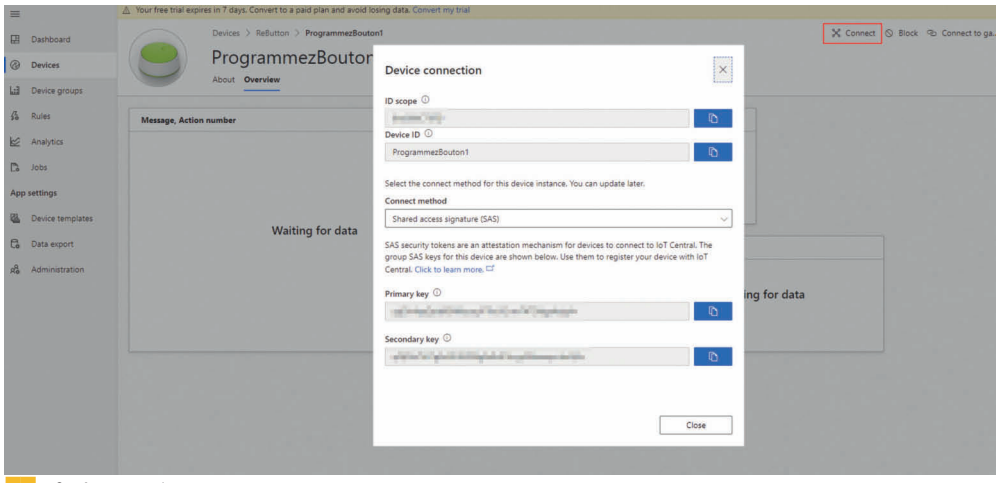
Revenez ensuite vers la Home Page pour accéder à la configuration de IoT Central.
Saisissez les informations du device qui ont été mises de côté lors de la création d'un device dans IoT Central. 26
Vous pouvez désormais éteindre le device (bouton Shutdown sur la page) et déconnecter le PC du wifi du device.
N'oubliez pas de reconnecter votre PC à internet pour la suite.



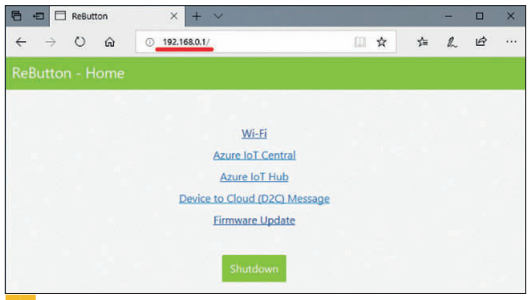
23 - *Wifi device*



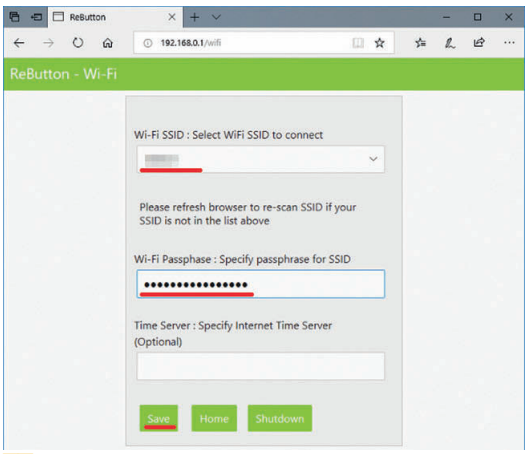
21 - *Création d'un device*



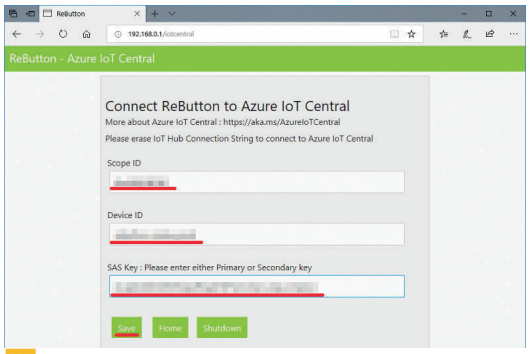
22 - *Configuration device*



24 - *Configuration device-Home page*



25 - *Configuration device-Wifi*



26 - *Configuration device-IoT Central*

Envoyer des informations

Il est temps de tester notre device, la connectivité et l'ensemble de la solution.
Pour cela, il suffit de cliquer sur le bouton.
La première fois, le bouton met un peu de temps à réagir.
Par défaut, chaque fois que nous appuyons le bouton avec un simple clic, la LED s'allume (couleur bleue) et clignote ensuite pendant une trentaine de secondes.

Contrôle de l'envoi des informations

Pour contrôler la réception des messages, nous devons retourner à IoT Central.
Par défaut, le device que nous avons créé intègre un dashboard qui visualise un certain nombre d'informations. Ce visuel est défini dans le template, mais vous pouvez le changer à votre façon ou créer vos propres visuels.
Après rafraîchissement de l'écran de notre device, on peut bien confirmer que le device envoie bien les messages pour chaque click. C'est parfait. 27
À titre d'exemple, vous trouverez ci-dessous un dashboard affichant l'historique des messages reçus et le niveau de charge des piles. 28

Mettre en place les actions déclenchées par la réception des messages (builds Azure DevOps)

À ce stade, nous avons un device IoT qui s'intègre avec une solution IoT et dans laquelle nous recevons bien les messages envoyés par le device.
Nous pouvons maintenant ajouter un peu d'intelligence du côté cloud : déclencher un pipeline de type build sur un Azure DevOps.
Pour rester dans la démarche « simple et low code » de cet article, nous allons utiliser des Azure Logic Apps et intégrer ces Logics Apps avec notre application IoT Central et Azure DevOps. Pour aller plus loin, vous aurez besoin des éléments suivants :

- Un Team Project dans un Azure DevOps (si vous n'avez pas d'Azure DevOps, vous pouvez en créer un gratuitement) ;
- Un pipeline de type build dans ce Team Project ;
- Un abonnement Azure (vous pouvez en créer un gratuitement si vous n'en avez pas).

Logic App

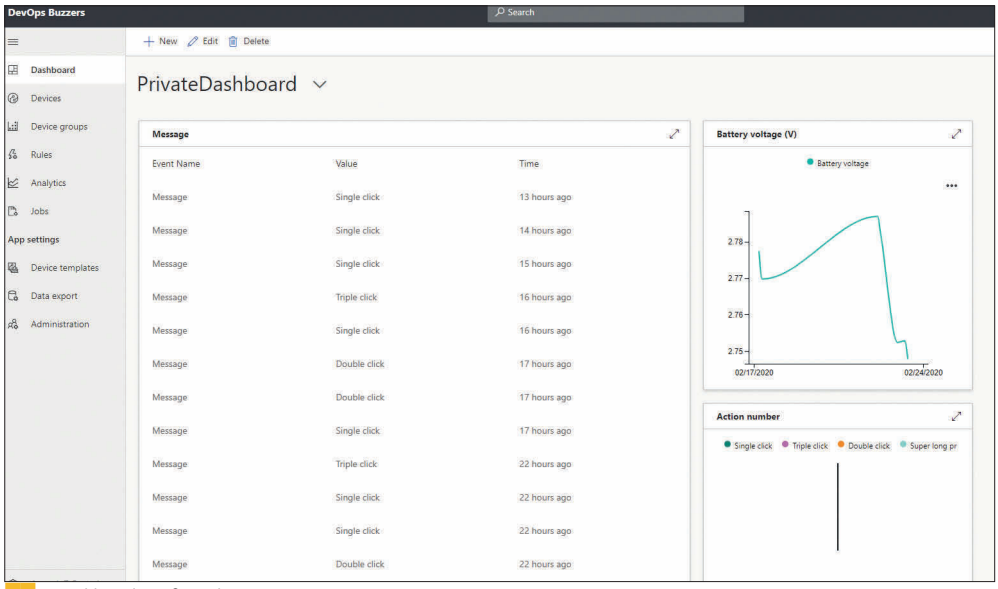
Nous allons créer une Logic App qui contient deux étapes :

- Un handler http, qui va recevoir l'appel du WebHook d'IoT Central.
- Une action qui déclenche un build Azure DevOps.

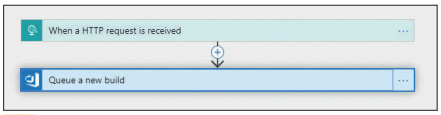
Steps Azure Logic App : 29
Détails du paramétrage : 30
Une fois les settings réalisés, pensez à bien enregistrer et exécuter la Logic App.
Ceci va générer l'instance du service qui va recevoir les appels d'IoT Central et l'URL (encadrée en rouge dans la capture d'écran précédente) que nous devons configurer dans le WebHook d'IoT Central.

WebHook IoT Central

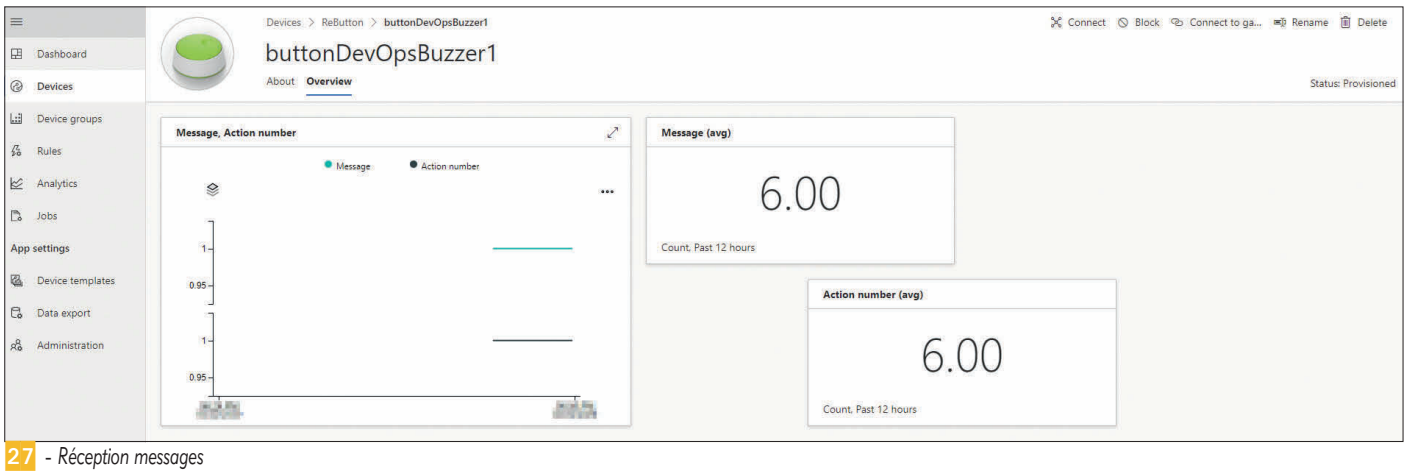
Pour configurer le WebHook IoT Central pour appeler la Logic App, il suffit de créer une règle dans IoT Central associée au template ReButton et de configurer un WebHook avec l'URL de la Logic App créée précédemment. 31
C'est terminé .



28 - *Dashboard IoT Central*



29 - *Azure Logic App, Steps*



27 - *Réception messages*

Contrôler le déclenchement des actions

Vérifions que tout fonctionne correctement avant d'annoncer la fin de la fin.

Pour cela, allons regarder l'Overview de la Logic App que nous avons créée et afficher l'historique des messages reçus. 32

Parfait, la Logic App est bien appelée et fait son travail sans anomalies.

Vérifions désormais si les builds Azure DevOps sont bien lancés.

Ceci se fait au niveau du Team Project que nous avons ciblé avec notre pipeline (portail Azure DevOps). 33

Et hop… les builds sont bien lancés.

Remarque importante :

Imaginez que nous devions désormais gérer des millions de devices, gérer les données qu'ils transmettent, les traiter, les analyser et les exploiter tout en garantissant leur intégrité.

C'est en cela qu'on commence à mettre le pied dans l'univers IoT et ses enjeux importants.
C'est en cela que le catalogue de services Azure IoT offre une grande valeur ajoutée, car, avec IoT Central, nous avons peu de choses à faire (éventuellement, gérer les templates de devices s'ils sont différents).

Par cet article, nous sommes en train de particulièrement mettre en avant les avantages d'une solution SaaS comme IoT Central. Certes, les avantages sont là et ils sont réels. Certains éditeurs ont suivi cette philosophie et proposent des solutions IoT verticalisées, par secteurs métier / business. Cette démarche reste tout à fait cohérente dans la logique de rationalisation des coûts par la possibilité de répliquer les solutions.

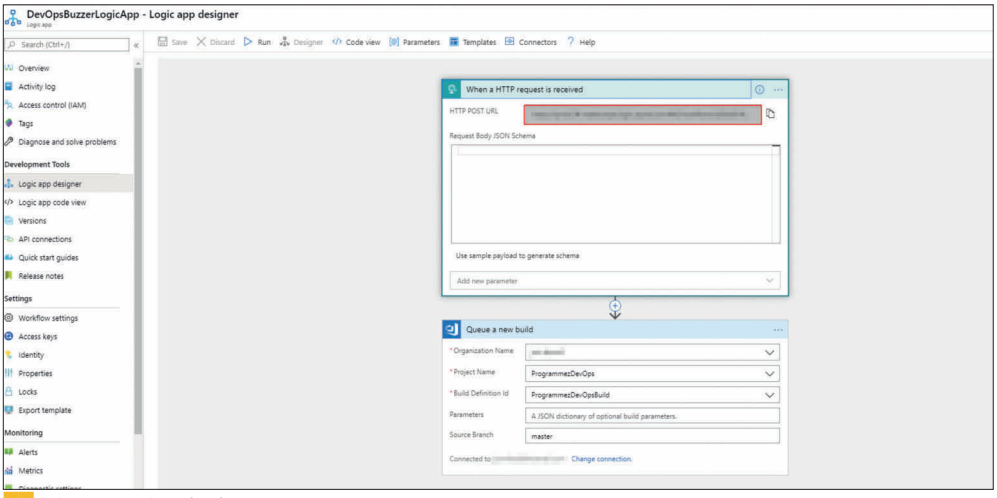
Néanmoins, dans certains projets, les solutions « clé en main » ne seront pas du tout adaptées et il faudra plutôt s'orienter vers les solutions PaaS mentionnées précédemment. La différence en termes de possibilités, performances globales, flux reste importante.

Les avantages de ces solutions seront décrits dans un autre article.

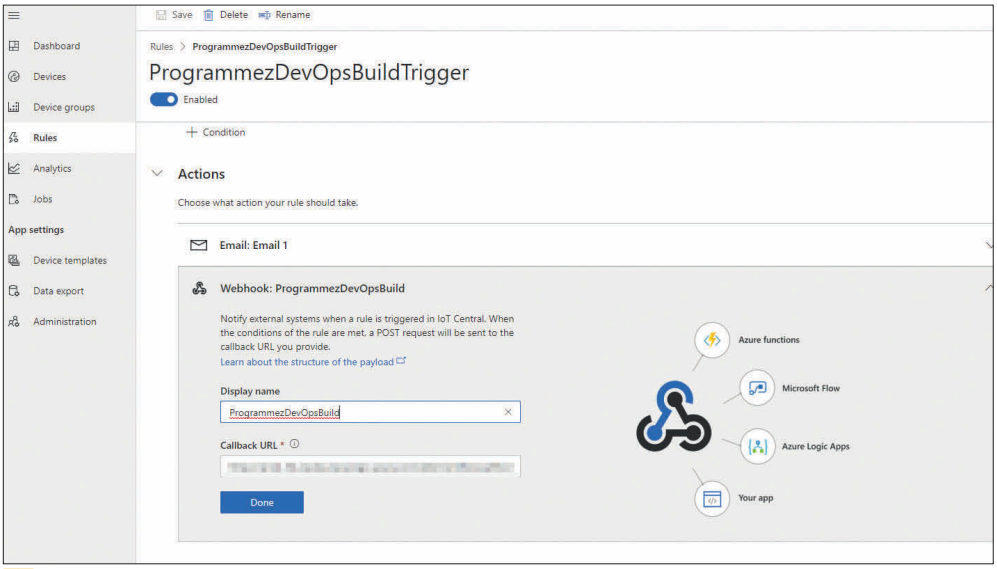
Conclusion

L'effet de mode autour de l'IoT finit par être très réducteur. Le marché IoT est en pleine transformation digitale.

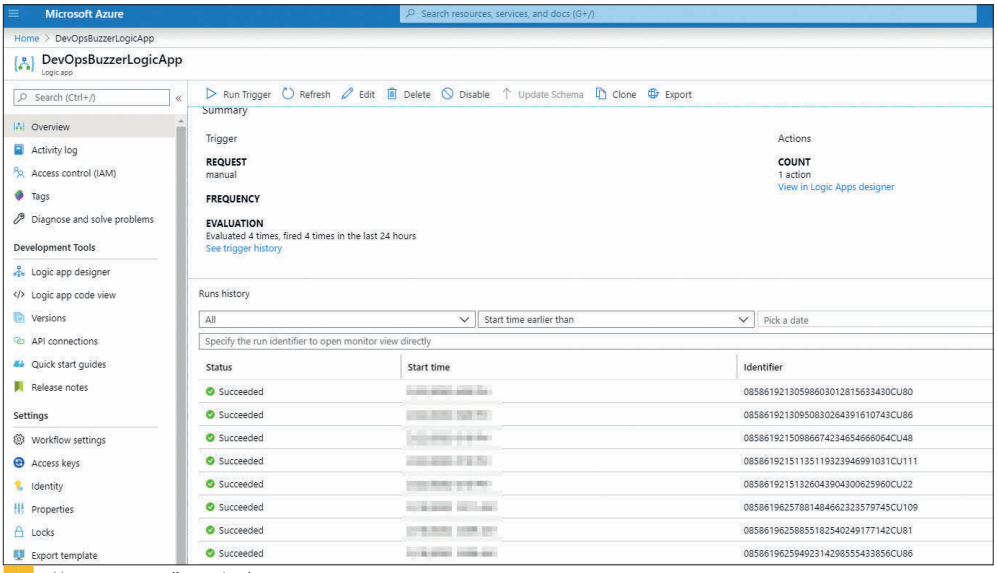
L'IoT est une discipline très riche qui exige une capacité de perspective et transversalité importantes. Les devices, la sécurité, la gestion des objets, les données, l'extraction de valeur et l'intégration de systèmes connexes, le tout étant très lié aux particularités métier de chacun des verticaux, exigent la création d'un nouveau type d'équipe de professionnels (commerciaux, ingénieurs, business analysts, Product Owners, etc.).



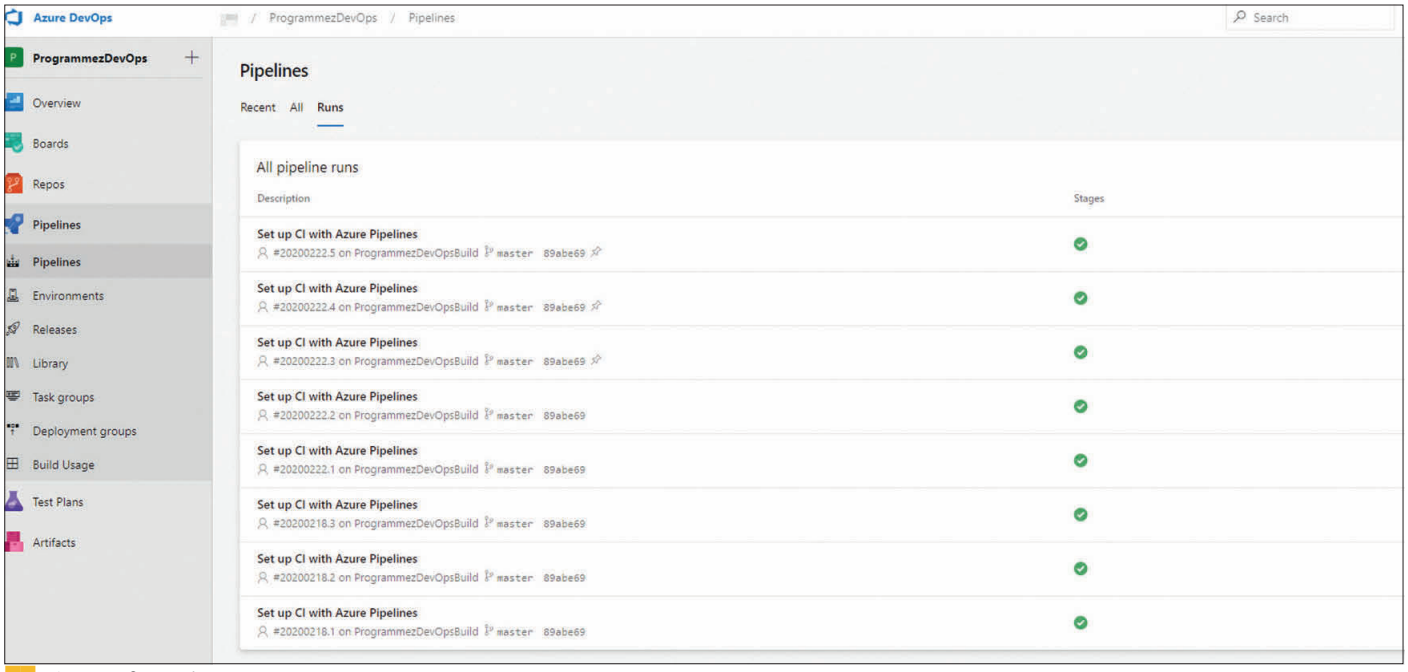
30 - *Azure Logic App, détails*



31 - *Webhook IoT Central*



32 - *Messages reçus (Logic App)*



33 - *Azure DevOps pipelines*

Les dirigeants et autres types de décideurs auront besoin de comprendre cet univers, extrêmement riche en potentiels, pour développer des stratégies et des business models adaptés.

Le travail de certains éditeurs a permis de réduire la complexité d'adoption de ces technologies en mettant la création de valeur comme axe central et en absorbant une partie des complexités techniques. L'IoT est tellement vaste et multidisciplinaire que la coopération et l'approche inclusive sont nécessaires pour construire des solutions cohérentes. Device-makers, éditeurs, partenaires et utilisateurs, tous sont liés et tous doivent vraiment travailler ensemble.

Par ce cas concret, vous avez créé une solution IoT avec un device qui va au-delà de l'action/réaction M2M — Machine To Machine — de deux objets. Votre solution permet d'intégrer un objet et les signaux qu'il envoie avec un système tiers et déclencher des actions.

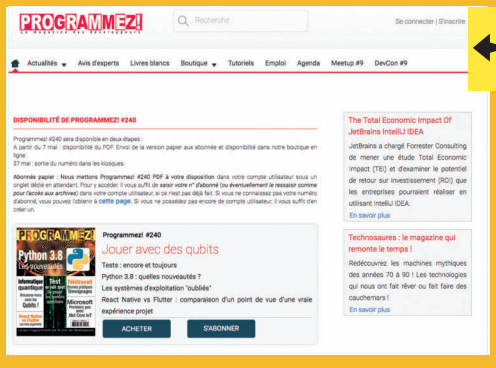
IoT Central vous permet de faire ceci en un temps record tout en intégrant un backoffice de gestion, des dashboards et visuels, des mécanismes d'exportation de données et des gestionnaires d'événements.

De plus, cette même application IoT Central vous permet de gérer 1 device, 10 devices, 10000 devices ou n devices sans aucun changement.

Un prochain article abordera la construction d'une solution IoT avec une approche différente, reposant sur des services PaaS — Platform As A Service —. L'exercice permettra d'illustrer les différences entre les deux approches et les points forts de chacune des stratégies.

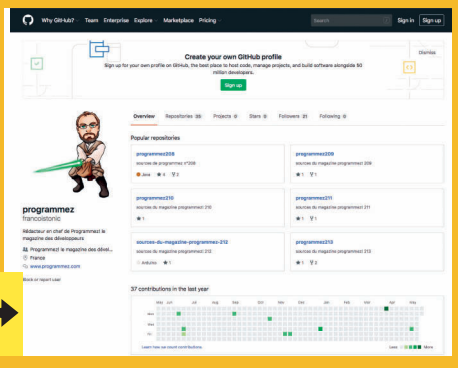
Bienvenue dans la matrice !

OÙ RÉCUPÉRER LES CODES SOURCES DES ARTICLES ?



← programmez.com

github.com/francoistonic →



Laurent Ellerbach
Principal Engineer Manager
Microsoft
laurelle@microsoft.com
https://github.com/Ellerbach

Hacker sa carte de crédit grâce à .NET Core IoT

Dans un de mes précédents articles, j'ai longuement expliqué le concept .NET Core IoT et comment l'utiliser. Nous allons utiliser ce projet Open Source multiplateforme pour lire notre carte de crédit et y récupérer toutes les informations possibles. Je vais en profiter pour vous expliquer les mécanismes de contrôle et d'authentification de l'utilisateur de la carte pour un paiement. En d'autres termes comment hacker sa carte de crédit. Vous verrez qu'il y a de très nombreuses informations disponibles sans avoir à être authentifié ou faire un code pin. Et nous en profiterons pour expliquer les mécanismes de contrôles mis en place par les banques pour sécuriser les transactions.

AVERTISSEMENT : CET ARTICLE EST À BUT PÉDAGOGIQUE. NOUS DÉGAGEONS TOUTE RESPONSABILITÉ SUR LA MAUVAISE UTILISATION, LE DÉTOURNEMENT DE CET ARTICLE À UN USAGE ILLÉGAL.

Matériel nécessaire

Il vous faut aussi soit un lecteur SmartCard USB traditionnel soit un lecteur RFID basé sur les chipsets PN532 ou PN5180. Il est à noter qu'un simple lecteur SmartCard USB coûte une dizaine d'euros et est une bonne façon de commencer. Cela utilisera la puce de la carte de crédit.

Il faut faire attention aux lecteurs RFID, il est important d'avoir un lecteur avec une puissance d'antenne la plus important possible. Les cartes nécessitent en effet pas mal d'énergie pour fonctionner. L'autre point c'est le N de NFC, qui veut dire Near, tout est fait pour pousser à avoir une vraie proximité dans la communication. La puissance de l'antenne est donc le facteur déterminant pour réussir à lire correctement une carte de crédit. Le premier lecteur que j'avais acheté sur la base d'un PN532 ne me permettait de ne lire qu'une seule de mes cartes et ne voyait pas mes passeports non plus, ni les autres composants à base de puce RFID. En général l'information sur la puissance de l'antenne n'est pas ce qu'il est de plus facile à trouver ! La plupart des lecteurs à base de PN5180 que j'ai trouvés fonctionnent relativement bien avec une antenne décente. Comptez aussi une bonne dizaine d'euros pour un lecteur RFID.

Vous pouvez trouver un lecteur SmartCard facilement sur vos sites d'achats en ligne préférés en France, ça peut être un peu plus compliqué pour les lecteurs RFID. Sinon votre site chinois préféré vous fera un plaisir de vous l'expédier, il faudra juste être très patient pour le recevoir !

Installation de .NET Core IoT

Comme présenté dans le n°240, .NET Core IoT est une implémentation complètement multi plateforme qui fonctionne aussi bien sur Windows que Linux. Le support de Mac est moindre car il n'y a que très peu de possibilités de support GPIO, SPI, I2C, etc. Il faut utiliser un périphérique tel que FT4222 pour en profiter, ce qui est supporté avec .NET Core IoT. Vous pourrez faire tourner le code exemple sur toutes les plateformes suivant le lecteur RFID ou SmartCard que vous avez. Il tournera également sans problème sur un Raspeberry Pi par exemple. Une fois cloné, vous aurez tout ce qu'il vous faut. La lecture des cartes à travers le lecteur SmartCard se fait en utilisant le Nuget PCSC qui est aussi multiplateforme qui utilise, suivant la plateforme, les couches natives, et propose une excellente abstraction en .NET.

Il vous faut bien sûr le runtime .NET Core en tant que tel, une version 3.1 fera largement l'affaire. Vous pouvez le faire dans l'environnement que vous voulez. Dans mon cas, j'utilise Windows 10 et Visual Studio 2019. Vous pouvez bien sûr utiliser VS Code mais aussi n'importe quel autre outil sur n'importe quelle plateforme.

Ce qui va nous intéresser dans ce package .NET Core c'est .NET Core IoT pour le support du lecteur RFID (Radio Frequency IDentification) avec le support NFC (Near Field Communication) PN532, PN5180 et surtout le support des cartes de crédit. Je suis d'ailleurs l'auteur du PR ayant permis le support de ces lecteurs RFID et des cartes de crédit. Cela s'est fait dans le contexte d'un projet bancaire où il était nécessaire de pouvoir lire les informations clés d'une carte de crédit en NFC ou à travers un lecteur USB de carte.

Vous trouverez d'ailleurs une maquette de ce projet sur le GitHub de Julien Chomarat, Software Engineer chez Microsoft : https://github.com/jchomarat/Bankiosk. Et vous trouverez aussi les sources de cet article sur mon GitHub : https://github.com/Ellerbach/ReadPayment-CreditCard

Notez que la plupart du code est également disponible avec les différents exemples des lecteurs RFID dans le repository .NET Core IoT : https://github.com/dotnet/iot .

Un peu de théorie

RFID est une technologie permettant la communication sans fil entre un chip et un récepteur. Cette technologie existe depuis de nombreuses années et a beaucoup évolué. Elle dispose de nombreuses versions pas forcément compatibles entre elles. Elle est utilisée pour les contrôles d'accès depuis des années.

Cette couche RFID peut fonctionner avec de nombreuses fréquences. En ce qui nous concernera, ce sera la bande 13,56 MHz. Ensuite, on applique dessus des protocoles de communication tels que ISO 14443-1 à 3 A/B, ISO 15693-3 et ISO 18000-3 pour les couches basses. Et on ajoute encore une communication à niveau plus élevé, dans le cas qui nous concerne pour les cartes de crédit

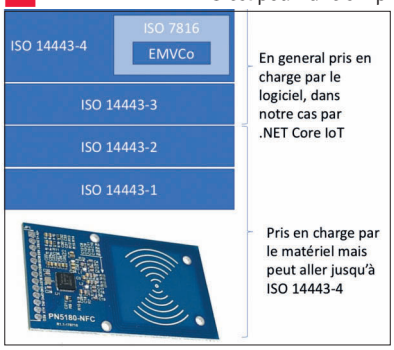
en ISO 14443-4 type B ! C'est d'ailleurs ce qui est également utilisé dans les passeports biométriques par exemple. NFC est juste pour dire proche, cela ne change rien au reste. Avec une antenne bien amplifiée, il est possible d'avoir une communication NFC à plusieurs dizaines de mètres ! Mais ce type d'antenne coûte plusieurs milliers d'euros.

Et il faut ensuite ajouter le protocole de communication spécifique pour les cartes à puces qui est celui qui est aussi utilisé pour la communication en NFC, le ISO 7816. Et puis le monde bancaire a ajouté encore un standard par-dessus, le EMVCo (https://www.emvco.com/) qui est à l'origine un consortium créé par Visa, MasterCard et JCB qui a bien sûr été rejoint par tous les organismes de paiement bancaires, y compris Carte Bleue en France. Nous verrons d'ailleurs plus loin que ces informations sont également accessibles.

Et tous sont des standards. Tous publiés, tous accessibles de façon payante (ISO) ou gratuites (EMVCo). Donc pour accéder aux données présentes dans votre carte de crédit, il vous suffit donc de lire toutes ces spécifications, implémenter ces technologies et voilà. C'est pour faire simple ce que j'ai fait avec en plus l'implémentation de drivers pour les lecteurs RFID. Ce sont plusieurs milliers de page par spécification notamment EMVCo qu'il faut lire !

Et pour mieux comprendre un schéma avec les différentes couches dans notre cas de carte de crédit : 1

1



Ce qui est compliqué dans tout cela c'est la superposition des couches notamment en RFID. Certains matériels vont jusqu'à prendre en charge plus ou moins correctement le protocole de communication 14443-4 Type B et d'autres non. L'autre partie compliquée est l'encapsulation du protocole ISO 7816 dans le protocole 14443-4 et à son tour le protocole EMVCo dans le protocole 7816 !

Pour l'échange de données avec le chipset en tant que tel, on utilise un protocole appelé APDU (Application Protocol Data Unit).

CLA	INS	P1	P2	Lc	Data	Le
Header				Trailer		

Une commande APDU c'est :

- CLA = Classe ;
- INS = Instruction ;
- P1 et P2 les paramètres dépendant de INS ;
- Lc = le nombre d'octets à envoyer à la carte ;
- Data = les données à envoyer en tant que tel ;
- Le = le nombre d'octets maximum attendu dans la réponse, si 0x00 alors 256.

Data	SW 1	SW 2
Body	Trailer	

La réponse APDU :

- Body ce sont les données renvoyées en tant que tel, s'il y en a ;
- SW1 and SW2 sont les informations de fin (trailers), 0x9000 indique un succès, et il existe des centaines de code d'erreur.

Il faut donc connaître les classes et commandes, les paramètres et tout le protocole associé pour récupérer une information.

Et pour vous faciliter la vie, l'implémentation .NET Core IoT fait un maximum d'abstraction, de façon à donner une fenêtre de communication la plus élevée possible. Dans le cas de la carte de crédit, une grande partie des primitives sont implémentées permettant de faire abstraction de presque toutes les contraintes sous-jacentes.

Faisons-nous peur

Le code exemple vous permet d'extraire les données d'une carte soit à travers un lecteur SmartCard soit à travers un des lecteurs RFID disponibles dans .NET Core IoT. Vous avez 3 matériels supportés et voici les options possibles :

- Lecteur USB SmartCard, tout est prêt, vous pouvez sélectionner l'option et insérer votre carte.
- Lecteur PN532, vérifiez que le port série est le bon, dans mon cas COM4 sous Windows. Si vous le lancez sur un Raspberry Pi ou sous Linux/Mac, ce sera certainement « /dev/ttyS0 » ou équivalent. Attention, ce lecteur possède aussi la possibilité d'être utilisé en SPI ou en I2C. Référez-vous dans ce cas à la documentation sur le Github .NET Core IoT. Dans tous les cas, attention à bien brancher les pins même pour la partie port série. Référez-vous à la doc du matériel, les noms des pins et leurs positions changent en fonction de l'implémentation.
- Lecteur PN5180, celui ne fonctionne qu'en SPI :
  - Si vous l'utilisez sur un Raspberry Pi, vous êtes presque prêt. Attention aux branchements des pins et aux numéros de GPIO, il faudra ajuster dans le code,
  - Si vous voulez l'utiliser depuis Windows, Mac ou PC, vous pouvez ajouter un dongle FT4222 de FTDI, un très bon disponible chez bitWizzard pour moins de 15€ : http://bitwizard.nl/shop/FT4222h-Breakout-Board?search=ft4222. Là aussi, faites bien attention au branchement des pins et vous êtes prêts.

Côté code, regardons ce que cela donne pour le lecteur PN5180 avec l'utilisation d'un dongle FT4222. Par exemple, les autres ont une initialisation légèrement différente, mais en ce qui concerne la partie liée à la carte, c'est exactement la même chose :

```
var ftSpi = new Ft4222Spi(new SpiConnectionSettings(0, 1) { ClockFrequency = Pn5180.Maximum
SpiClockFrequency, Mode = Pn5180.DefaultSpiMode, DataFlow = DataFlow.MsbFirst });

var gpioController = new GpioController(PinNumberingScheme.Board, new Ft4222Gpio());

// Reset the device
gpioController.OpenPin(0, PinMode.Output);
gpioController.Write(0, PinValue.Low);
Thread.Sleep(10);
gpioController.Write(0, PinValue.High);
Thread.Sleep(10);
var _pn5180 = Pn5180(ftSpi, 2, 3, gpioController, true);
Data106kbpsTypeB card;
// Poll the data for 20 seconds
var ret = _pn5180.ListenToCardIso14443TypeB(TransmitterRadioFrequencyConfiguration.Iso
14443B_106, ReceiverRadioFrequencyConfiguration.Iso14443B_106, out card, 20000);
Console.WriteLine();
if (!ret)
```

```
{
  Console.WriteLine("Can't read properly the card");
}
else
{
  var creditCard = new CreditCard(_pn5180, card.TargetNumber, 2);
  creditCard.ReadCreditCardInformation();
  DisplayTags(creditCard.Tags, 0);
  // Display Log Entries
  var format = Tag.SearchTag(creditCard.Tags, 0x9F4F).FirstOrDefault();
  if (format != null)
    DisplayLogEntries(creditCard.LogEntries, format.Tags);
}
```

La première partie consiste à créer le support pour le FT4222. On crée le support SPI car le PN5180 utilise SPI et également GPIO car nous en aurons besoin pour réinitialiser le PN5180. De plus, de façon interne, il utilise également GPIO pour le transfert des données.
Ensuite, on crée le PN5180 et on essaie pendant 20 secondes de lire une carte. Si une carte est trouvée alors on crée ensuite une carte de crédit et on lit l'intégralité des données. Cela ne prend que quelques secondes. Sur un vrai terminal, cela prend un peu moins de temps car toutes les informations ne sont pas lues. La dernière partie consiste juste à afficher les données et les données de l'historique.
Allez, considérons que vous êtes prêt, lançons le code et regardons ce que nous arrivons à extraire. Ici avec le lecteur SmartCard en USB (note : pour des questions de sécurité, j'ai bien entendu changé quelques éléments) :

Code complet à récupérer sur programmez.com & github

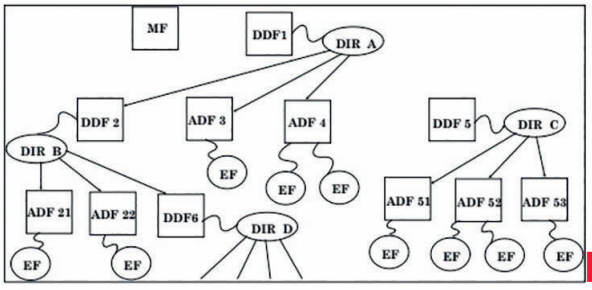
Je pense qu'une aide à la lecture de ces informations s'impose :-)
Tout d'abord, il faut comprendre que l'organisation des informations sur la carte, comme sur tout chipset ISO 7816 se fait à base d'une pseudo structure de fichiers avec des fichiers maîtres, des sortes de répertoires et des sous-fichiers. 2
Ils sont composés de templates avec des numéros. Le tout est encodé en TLV (Tag Length Value). C'est un encodage simple très utilisé dans les communications et le stockage. Il est souvent apparenté au BER (Basic Encoding Rule) et ASN.1 (Abstract Syntax Notation One). Mais évidemment, pour faire simple, tout n'est pas encodé suivant TLV et il est nécessaire d'avoir la liste de tous les tags pour les décoder correctement. Certains contiennent des informations fixes: d'où l'organisation sous forme d'arbre dans les résultats récupérés. 3
D'abord ce qui saute aux yeux, c'est la quantité d'information que l'on peut récupérer. On récupère bien sûr le numéro de la carte, les dates de validité, le nom du porteur (ce n'est pas le cas en NFC) mais aussi des informations telles que des certificats.
Prenons les choses dans l'ordre. On s'aperçoit qu'il y a en fait 2 moyens de paiement dans ma carte à travers les tags « 0061-Application Template », l'un « Visa » et l'autre « CB Comptant ». En France, parce que nous avons des cartes à puces depuis très longtemps, les banques s'étaient mises d'accord sur une Carte Bleue permettant de payer facilement partout. Et c'est toujours le moyen le plus utilisé quand vous payez dans un magasin en France. D'ailleurs on voit le tag « 0087-Application Priority Indicator » à 1 pour CB et 2 pour Visa. En descendant un peu, on trouve plus de détails sur CB avec « 006F-File Control Information (FCI) Template » et notamment la langue préférée « fren » pour French donc français.
Un peu plus bas, nous allons trouver les informations les plus intéressantes à savoir, le numéro de carte mais aussi les dates d'expirations et le pays émetteur (0250 = France)

```
0070-Template, AEF Data
 005A-Application Primary Account Number (PAN): 4974901234567895
 5F24-Application Expiration Date: 2022/03/31
 9F4A-Static Data Authentication Tag List: 82
 9F07-Application Usage Control: FF-00
 5F28-Issuer Country Code: 0250
```

Le numéro a volontairement été transformé et n'est pas valide. Cela étant dit, il donne tout de même pas mal d'informations : 4

Partie 1 : elle s'appelle Major Industry Identifier (MII) et il permet d'identifier le type de carte :
- 1 et 2 pour les compagnies aériennes, 3 pour les voyages ;
- 4 et 5 pour les banques, 6 pour le Merchandising et autres applications bancaires ;
- 7 pour l'industrie pétrolière ;
- 8 pour la santé et les télécommunications ;
- 9 pour une utilisation par les états.

Si vous avez des cartes de fidélité type carte de paiement, c'est le moment de jeter un œil, vous trouverez qu'effectivement, ces codes sont respectés.



2

Tag	Value			Presence
'6F'	FCI Template			M
	'84'	DF Name		M
	'A5'	FCI Proprietary Template		M
		'88'	SFI of the Directory Elementary File	M
		'5F2D'	Language Preference	O
		'9F11'	Issuer Code Table Index	O
		'BF0C'	FCI Issuer Discretionary Data	O
		'XXXX' (Tag constructed according to Book 3, Annex B)	1 or more additional proprietary data elements from an application provider, issuer, or IC card supplier, or EMV-defined tags that are specifically allocated to 'BF0C'	O

3

4

Partie 2 : Issuer Identification Number (IIN), ce sont 6 chiffres qui permettent souvent d'identifier l'émetteur. Par exemple Amex utilise 34xxxx, 37xxxx, Visa – 4xxxxxx et MasterCard – 51xxxx – 55xxxx.
Dans mon cas il s'agit bien d'une carte Visa, nous le savions déjà par l'extraction des données, le numéro le confirme aussi.
Partie 3 : le numéro compte en tant que tel constitué de 9 chiffres. C'est la partie que j'ai masquée.
Partie 4 : le check digit. Il est basé sur l'algorithme de Luhn (Mod 10). On commence par le digit de check, on double une fois sur deux par digit de droite à gauche. Quand on double, si le résultat est supérieur à 10, on ajoute les 2 digits. Et on ajoute le tout. Si c'est un multiple de 10, c'est gagné ! Dans mon cas, vu que j'ai changé les chiffres, ce digit n'est plus le bon.
En descendant encore, un peu plus bas on retrouve le tag « 5F20-Cardholder Name: ELLERBACH/LAURENT.M ». Cette information n'est pas donnée en accès RFID.
En allant encore plus bas, on retrouve presque le même pattern, pour les informations concernant Visa. Dans les 2 cas, le numéro de carte, les dates de validité sont les mêmes. Ma carte pourrait en fait contenir plusieurs comptes avec des informations différentes. J'y reviendrai un peu plus loin.
Et tout à la fin, on trouve le nombre d'essais qu'il me reste, ici 3, et aussi le format de l'historique des achats. A noter qu'ici, l'historique est vide. Il n'est pas accessible quand on n'est pas authentifié. J'ai cependant d'autres cartes où l'historique est accessible totalement. Et je commenterais également certains des autres tags plus loin.

Peur ou pas peur ?

A première vue, tout cela peut faire peur ! Un lecteur de carte physique ou NFC, pas de code pin, et bam, on arrive à extraire toutes ces informations ! Le numéro de la carte, la date de validité, le nom du porteur. Tout ce qu'il me faut pour faire une transaction. Bon, là je sens que vous avez peur.
Rassurons-nous. Tout d'abord ces mêmes informations sont imprimées sur la carte. Oui, elles sont sur la carte. Je le réécris, sur la carte ! Une simple photo et vous avez ces mêmes informations. C'est quand même plus facile comme ça. Oh et puis, elles sont aussi sur la piste magnétique derrière la carte. Très facile à lire !
On retrouve d'ailleurs dans le tag « 0057-Track 2 Equivalent Data: 49-74-90-12-34-56-78-95-D2-20-32-01-34-87-77-47-80-00-0F » ces mêmes informations. C'est l'équivalent de ce qui se trouve sur la bande magnétique. Le numéro de carte va jusqu'à 95, ensuite on a la D2-20-32-01-3 qui est en fait la date 2022/03/31. Oui, l'encodage est un peu particulier. Et on a ensuite quelques autres éléments pour la transaction qui sont les éléments discrets pour la transaction.
Mais alors pourquoi nous dit-on que le code pin est sécurisé pour une transaction ?
En fait, quand on regarde le processus complet de paiement, il y a de nombreuses étapes :

- Validation de l'authenticité de la carte : lors d'une transaction entre un terminal et une carte, la carte est d'abord authentifiée pour vérifier qu'elle est valide. Il existe 2 méthodes qui nécessitent des certificats :
  - Static Data Authentication: vérification statique,
  - Dynamic Data Authentication: vérification dynamique.
- Vérification du porteur de la carte : code pin, signature ou les deux !
- Management des risques du terminal :
  - Limite de paiement pour éviter de couper les transactions en petits morceaux et basée sur l'historique de paiement. C'est là qu'il y a un intérêt à l'avoir stocké et accessible,
  - Sélection aléatoire pour une vérification en ligne,
  - Vérification suivant des règles comme pour toute transaction supérieure à un certain montant.
- Après toutes ces étapes, le terminal décide de procéder à une vérification en ligne, de faire passer la transaction sans vérification ou de rejeter le paiement.

Là, vous commencez à comprendre à quoi vont nous servir tous les autres éléments récupérés et pourquoi ils sont accessibles sans code pin ! Le code pin ne sert qu'à identifier que le porteur de la carte est bien celui qui dit qu'il est. Et possiblement à l'historique de transactions et d'autres informations qui peuvent être stockées secrètement. Mais pour faire une transaction il est inutile. Alors pourquoi nous rabattre les oreilles avec ce fameux code ? Tout simplement parce que la carte est utilisable par une seule personne. Le terminal de la transaction physique (de paiement dans un magasin, de l'automate d'un distributeur) doit identifier la personne. C'est pourquoi le pin est demandé.
On notera que la signature peut être également exigée comme moyen de vérification et que même les deux peuvent l'être. C'est d'ailleurs souvent le cas pour des transactions de montants importants ou lorsque vous êtes en pays étranger. Cela permet juste de limiter les risques que la transaction soit frauduleuse. Voilà pour la réponse sur le code pin.
Pour le reste, avant même qu'on vous demande ce code, la carte est vérifiée. C'est à cela que servent la série de tag « 0090-Issuer Public Key Certificate », « 008F-Certification Authority Public Key Index », « 0092-Issuer Public Key Remainder », « 9F32-Issuer Public Key Exponent », « 9F46-ICC Public Key Certificate », « 9F47-ICC Public Key Exponent », « 9F08-Application Version Number », « 9F48-ICC Public Key Remainder ». Ils sont utilisés avec des certificats présents dans le terminal pour faire des opérations de cryptographie et vérifier que le résultat obtenu est correct. Je ne rentrerai pas dans le détail. Mais pour faire simple, l'authentification statique se base sur des données statiques, qui d'un point de vue sécurité peuvent être copiées dans une carte clone alors que l'authentification dynamique se base sur une génération de nombre plus ou moins aléatoire avec une cryptographie dynamique qui n'est pas réplicable sans avoir le certificat d'origine. Là aussi, l'idée est de limiter les risques au maximum. Tous les terminaux modernes supportent bien entendu l'authentification dynamique qui est le mode par défaut. Les cartes doivent supporter au moins le mode statique.
Si la carte n'est pas vue comme valide, le paiement sera rejeté avant qu'on vous demande votre code pin.
Ensuite, comme indiqué, si les deux premières étapes sont bien passées, une vérification en ligne peut être faite. C'est en général quand le commerçant et vous râlez parce que cela prend du temps, qu'on capte mal et que parfois, il faut réessayer.
Et il faut également imiter un terminal pour obtenir l'ensemble de ces informations à travers les « Card Risk Management Data Object List ». Leur type varie en fonction du paiement Visa, CB, Amex ou autre. D'ailleurs les « CDOL1 » pour le Visa et CB sont différents. Le terminal doit remplir ces informations pour pouvoir accéder à cer-

tains des éléments pour l'authentification de la carte.
Et qu'en est-il du paiement par NFC ? L'idée est en fait la même que précédemment. Le processus est le même. Les différences se font au niveau de des codes de vérification CVC. J'y reviens juste après.
Techniquement, il est possible de faire une copie d'une carte, de générer une très grosse partie des informations dans un chip. Mais la carte échouera dans la vérification dynamique et bien sûr en cas de vérification en ligne notamment à cause des codes CVC.
Et alors, en ce qui concerne les Google Pay, Apple Pay et autres solutions dans le téléphone, comment cela se passe-t-il ? Eh bien, cela fonctionne de la même façon que votre carte, avec les mêmes fonctionnalités: la carte est émulée et tous les moyens de vérification sont appliqués. Vous avez en quelque sorte une carte virtuelle. A noter que des secrets (certificats renouvelables) sont stockés en plus permettant la vérification dynamique de la carte. Quant à l'authentification de l'utilisateur, elle s'effectue par le code pin qui est demandé lorsque vous présentez votre carte pour un paiement.

CVC, CVV1, CVV2, CVV3, WTF ?

Voici la définition de chacun :

- CVC = Card Validation Code ;
- CVV = Card Validation Value.

Ce sont les mêmes, ils étaient juste utilisés dans les pistes magnétiques par différentes banques et elles leur ont donné des noms différents.

- CVV1 était embarqué dans les pistes magnétiques et envoyé à la banque ;
- CVV2 est écrit au dos de la carte et utilisé pour la vérification non présentielle. Donc pour une transaction en ligne ou par téléphone. La longueur du code varie, il est par exemple 3 pour Visa, MasterCard et 4 pour Amex. Et pour Amex, il se trouve sur la face de la carte, juste en dessous du numéro de la carte ! Ce qui rend très facile la copie de cette information qui doit rester pourtant la plus secrète possible ;
- CVV3 est un CVV dynamique, les paiements NFC utilisent cette méthode et le code est différent à chaque transaction.

Penchons-nous sur la vérification CVV2. Il va nous falloir quelques informations :

- Card Verification Key 1 (seul l'émetteur de la carte connaît cette clé), 16 bits ;
- Card Verification Key 2 (seul l'émetteur de la carte connaît cette clé), 16 bits ;
- Le numéro de la carte, le fameux PAN que nous avons récupéré avant ;
- La date d'expiration de la carte au format YYMM ;
- Un code de service (000 for CVV1), (999 pour CVV dynamque tel que pour les paiements NFC), varie pour les paiements CVV2.

Ensuite on applique l'algorithme de vérification suivant :

- Tocheck = append(Pan, expiration, service code, 0) total length 32 ;
- Tocheck1 = Tocheck.Substring(0,16) ;
- Tocheck2 = Tocheck.Substring(16) ;
- First = DES.Encrypt(Tocheck1, Key1, NoPadding) ;
- Second = XOR(HexToByte(First), HexToByte(Tocheck2)) ;
- CVVFull = TipleDEX.Encrypt(Second, Key1 + Key2, NoPadding) ;
- CVV = SelectOnlyDigits(CVVFull, 3 or 4 for length).

Note : *SelectOnlyDigits ne va sélectionner que les digits, pas les alphanumériques. S'il n'y a pas assez de digits, il convertira les premiers A à F en utilisant une table de décimation.*

Ce mécanisme basé sur des secrets très bien gardés, les clés Key1 et Key2 sont celles utilisées lorsque la transaction est vérifiée, soit lors d'un paiement sur un site internet, soit au téléphone, soit lorsque le terminal se connecte au centre de vérification. Et dans tous les cas a posteriori lorsque les transactions effectuées en mode hors connexion sont téléchargées à la banque. Et comme on peut s'en rendre compte, il reste facile de copier le numéro CVV d'une carte et donc de pouvoir effectuer une transaction avec.
L'idée des banques est donc de continuer à limiter la fraude, notamment la copie de ce numéro CVV pour des achats en ligne en imposant un mécanisme de double authentification. Les banques européennes le supportent maintenant, limitant ainsi les risques de fraudes. Cela se fait à travers votre téléphone, recevant un SMS avec un code ou une authentification à effectuer sur l'application de votre banque. Le problème vient majoritairement des sites et fournisseurs de solution de paiement qui ne supportent pas cette double authentification (Three Domain Secure (3-D Secure) ou Tokenisation).

Conclusion

Avec tout ce que nous venons de voir, il est plus facile de prendre une photo d'une carte de paiement, des deux côtés pour récupérer ces informations que n'importe quoi d'autre ! Et cela est suffisant pour effectuer des paiements à distance comme online quand le paiement n'utilise pas une technique de double authentification.
Pour faire une copie de carte, il faut une puce permettant d'écrire les informations que l'on peut extraire avec le code exemple fourni, modifier certains éléments comme le non-support de l'authentification dynamique, ajouter un mécanisme de support de code pin et les mécanismes, comme pour une vraie carte de réponse. Cela est suffisant pour une transaction sur un terminal de paiement, pour des petits montants où il n'y a pas de vérification en ligne, donc des terminaux purement déconnectés. Sinon, le risque d'avoir une connexion pendant la transaction est trop important. Et puis, il ne s'agit là que d'une seule carte qui ne permettra au mieux que quelques transactions. C'est donc beaucoup d'efforts pour pas grand-chose.
L'autre solution est bien sûr de faire la même chose dans votre téléphone en ajoutant une carte de paiement et en payant en NFC. Là aussi, il y a des limites notamment parce que les mêmes processus de vérification s'imposent.
Le plus simple reste encore pour les pirates d'exploiter les données volées sur des serveurs de numéro, date d'expiration, nom du porteur et parfois, alors qu'il ne faut pas les stocker, les CVV. Et puis d'utiliser des sites qui ne supportent pas des solutions de paiement avec double authentification. La raison pour laquelle les banques poussent à la double authentification pour toutes les transactions en ligne, c'est bien sûr pour limiter la fraude. Les plafonds sur les paiements sans fil et la vérification systématique des CVV dans les transactions connectées sont aussi des éléments de réduction de la fraude.
Côté développeur, le mieux est d'utiliser des solutions de paiement éprouvées, qui utilisent la double authentification de façon transparente. Il vaut mieux éviter de faire sa propre solution !

SpaceX : 1 - Développeur : 0

Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication : François Tonic
Rédacteurs en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : la rédaction de ZDnet
Nos experts techniques : L. Ellerbach, J. Bezet-Torres, G. Debray, P. martin, F. Dubois, J-M. Torres, F. Mocq, J. Proust, S. Colas, J. M. Inza, CommitStrip
Couverture : © François Tonic - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : SIB Imprimerie
Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude - Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD Téléphone : 09 67 32 09 34

Contacts : Rédacteur en chef : ftonic@programmez.com - Rédaction : redaction@programmez.com - Webmaster : webmaster@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, juillet 2020

Abonnement :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com*
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.

Tarifs
Abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € - Autres pays : nous consulter.

PDF
35 € (monde entier) souscription sur www.programmez.com

INFORMER
pour transformer l'entreprise

La dématérialisation, le Cloud,
les communications unifiées,
les nécessités de la cybersécurité
transforment le travail et toute l'entreprise,
les services publics.
Le magazine, le site, ses newsletters
vous informent sur cette actualité
mouvante
et vous aident à décoder les tendances

Abonnez-vous

www.solutions-numeriques.com/abonnement/

LIRE

au format digital

- Vous êtes responsable informatique ou bien dirigeant ou cadre d'entreprise ?
  2 sites répondent à votre profil
- La cybersécurité vous concerne ?
  Cliquez sur l'onglet. Vous trouverez les infos, l'annuaire, le lexique, etc
- L'emploi, les salaires, les formations, les offres vous intéressent ?
  Le site sur l'Emploi dans le numérique est à votre disposition

www.solutions-numeriques.com