PROGRAMEZ!
Le magazine des dévs
N°253
07/08
2022

COMMENT CHOISIR SON LANGAGE DE PROGRAMMATION ?

3615 HACKER SON MINITEL

MON 1ER SITE AVEC JAMSTACK

PROJET VALHALLA va-t-il révolutionner JAVA ?

© Diy13

M 04319 - 253 - F: 6,99 €- RD

Printed in EU - Imprimé en UE - BELGIQUE 7,50 € - Canada 10,55 $ CAN - SUISSE 14,10 FS - DOM Surf 8,10 € - TOM 1100 XPF - MAROC 59 DH

#253*

Minitel : retour vers le passé

Le Minitel est le 1er matériel électronique français produit en masse et diffusé à des millions d'exemplaires. Il représente une certaine vision de l'informatique et des télécoms : comment rendre accessible des services divers et variés rapidement et simplement. L'État avait massivement investi dans le réseau téléphonique dans les années 1970 pour rattraper le retard. Il fallait maintenant l'exploiter et rentabiliser le réseau.

Les premiers travaux démarrent en 1973-74 avec la conception TIC-TAC (Terminal Intégré Comportant Téléviseur et Appel au Clavier). Il montre comment exploiter le réseau télécom, comment transporter des informations et comment les diffuser sur un terminal. D'autres projets sont en gestation comme le projet TITAN. Chaque entité télécom française voulait son projet, souvent sans échange avec le projet d'en face. Les chercheurs et les responsables regardent ce qu'il se passe en Angleterre et en Allemagne. Ces pays sont en avance. Le rapport « l'informatisation de la société », publié en 1978, accélère les choses. Et surtout, un mot magique apparaît pour la première fois : télématique. Est-ce que l'association des télécoms et d'informatique peut permettre de rattraper le retard de la France ?
Les premiers prototypes sortent fin années 1970. Le projet de l'annuaire électronique est testé en 1980 puis les expérimentations se développent

dans plusieurs villes. Il faut le matériel, mais surtout des services, bref des applications et des contenus !

Le véritable envol du Minitel se fait courant 1983. L'apparition du concept tarifaire du kiosque, en 1984, va accélérer le déploiement du minitel et attirer de nombreux services.

Le minitel est un paradoxe : il fait découvrir le monde de l'informatique, des services à des millions de Français, mais il va aussi ralentir l'adoption d'Internet. Rappelons que le Minitel n'est pas un micro-ordinateur, il ne sait pas traiter les informations, ni les stocker. Et surtout, par conception : impossible de le mettre à jour. Ce choix va peser sur l'évolution des services et l'ajout de nouvelles fonctions avancées telles que le graphisme, des services plus complexes.

Le déclin s'observe rapidement dès la fin des années 1990 avec la croissance d'Internet en France. Le Minitel finira par être débranché le 30 juin 2012.

Pour ce numéro d'été, Programmez! a voulu rallumer le Minitel. Vous allez découvrir comment utiliser ce vénérable terminal avec Linux, interfacer un ESP32 ou encore utiliser des librairies Python pour développer de nouvelles applications ! Bref, du hack comme nous les aimons à Programmez!.

Nous vous proposons beaucoup de bonnes choses pour cet été : développer des jeux avec le formidable outil Age, l'outil Prettier pour bien formater son code Java, Jamstack par la pratique, des IoT à créer et à coder, comment choisir son langage, un usage de Sudoku un peu particulier et d'autres petites surprises !

Bon été avec Programmez!

**En réalité n°260*

François Tonic
Grand Moff

LES PROCHAINS NUMÉROS

NUMÉRO SPÉCIAL
100% SÉCURITÉ
100% HACK
Disponible dès le 9 septembre

PROGRAMMEZ! N°254
Disponible le 30 septembre

Contenus

Programmez! est une publication bimestrielle de Nefer-IT.
Adresse : 57, rue de Gisors 95300 Pontoise – France. Pour nous contacter : redaction@programmez.com

Disponible

Kiosque / Abonnement - Version papier / Version PDF

Les événements Programmez!

Meetups Programmez!

13 septembre - 18 octobre
8 novembre - 13 décembre

Où : Scaleway Paris
A partir de 18h30

DevCon #15 100 % Kotlin

3 novembre 2022

Où : Campus de l'école ESGI
242 rue du Faubourg Saint-Antoine, Paris
Transport : Nation (RER A, Métro 1, 2, 6, 9)
A partir de 13h30

Informations & inscription : programmez.com

Conférence DevCon .Net & technologies Microsoft

3e édition - décembre 2022

Septembre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
			1	2	3	4
5	6	7	8	9 JUG SummerCamp (La Rochelle)	10	11
12	13 Meetup Programmez!	14	15	16	17	18
19	20	21	22	23	21	25
26	27	28	29 CloudNord (Lille)	30		

Octobre

Lun.	Mar.	Mer.	jeu.	Ven.	Sam.	Dim.
					1	2
3	4	5	6 Paris Web (Paris)	7	8	9
10	11	12	13 Volcamp (Clermond Ferrand) Forum PHP (Paris)	14	15	16
17	18 Meetup Programmez!	19	20 DevFest Nantes	21	22	23
24	25	26	27 Agile Tour (Bordeaux)	28	29	30
31						

A venir

- Open Source Experience : 8-9 novembre / Paris
- ParisTestConf : 15-16 novembre/ Paris
- Agile Tour Toulouse : 15-16 novembre / Toulouse
- Codeurs en Seine : 17 novembre / Rouen
- DevFest Strasbourg : 18 novembre / Strasbourg
- Capitole du libre : 18 & 19 novembre / Toulouse
- DevOps DDay : 1er décembre / Marseille
- BDX I/O : 2 décembre / Bordeaux
- API Days Paris : 14-16 décembre / Paris

2023

- SnowCamp : 25-28 janvier 2023 / Grenoble
- Devoxx : 12-14 avril / Paris
- Best of web : juin / Paris

Merci à Aurélie Vache pour la liste 2022, consultable sur son GitHub : https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

Les partenaires 2022 de

PROGRAMMEZ!

Le magazine des dévs

Niveau maître Jedi

salesforce

softfluent

Scaleway

The cloud that makes sense

Niveau padawan

OSAXiS

Vous voulez soutenir activement Programmez! ?
Devenir partenaires de nos dossiers en ligne et de nos événements ?

Contactez-nous dès maintenant :

ftonic@programmez.com

SQL distribué : l'architecture de base de données de l'ère numérique

Pour leurs modèles économiques numériques, les entreprises ont besoin de bases de données à dimensionnement rapide, dotées d'une forte capacité de stockage et de performances puissantes pour les gros volumes de transactions. Parce qu'elles intègrent ces fonctionnalités, les bases de données SQL distribuées s'imposent dans plusieurs secteurs, comme le e-commerce par exemple.

Sylvain Arbaudie
Senior Customer Engineer EMEA, MariaDB Corporation

Un constat

Le volume de données ne cesse d'augmenter, alimenté par des applications numériques de plus en plus nombreuses. Dans une étude réalisée par le cabinet IDC, le volume total de données créé dans le monde est évalué à 59 zettaoctets, une profusion qui s'explique aussi par l'évolution du mode de travail. Les entreprises collectent toujours plus de données, qui proviennent des clients, des flux de marchandises, des machines, des réseaux sociaux... Parallèlement à cette augmentation constante, l'accentuation de la mondialisation pose d'autres défis en matière de bases de données, comme ceux des entreprises, dont les clients, sont répartis à travers le monde, et qui ont besoin de bases de données capables de traiter des pics de charges en continu, de jour comme de nuit.

Les systèmes classiques de bases de données relationnelles n'ont pas été conçus pour supporter ce type d'évolutivité inhérent aux applications modernes. Concrètement, ils ne peuvent pas assurer le stockage puis le traitement de tels volumes d'information.

Les bases de données traditionnelles ne suffisent plus

Les systèmes de gestion de bases de données relationnelles restent la référence dans le monde des entreprises, car ils sont extrêmement fiables et garantissent l'homogénéité et l'intégrité des données. Or, ce modèle date des années 1970, tout comme le langage SQL d'interrogation de base de données (Structured Query Language). Ces deux concepts sont étroitement liés au point que les gens parlent souvent de bases de données SQL lorsqu'ils se réfèrent aux serveur de gestion de bases de données (SGBDR).

Les bases de données traditionnelles ont atteint leurs limites, faute de pouvoir accompagner la croissance des entreprises numériques où l'évolutivité est le maître-mot. Que faut-il donc choisir ? Faut-il privilégier l'homogénéité et l'intégrité des données au détriment de l'évolutivité ? Ou faut-il privilégier l'évolutivité avec l'homogénéité qui s'ensuit ?

Le meilleur des deux mondes

Cette dichotomie est cependant évitable, car SQL distribué est en réalité une nouvelle génération de bases de données relationnelles qui permet des traitements à une échelle jusqu'alors impossible, tout en maintenant une forte homogénéité et intégrité des données. Une fois déployées, les bases SQL distribuées fédèrent plusieurs nœuds de base de données qui fonctionnent comme un cluster unique. Il suffit d'ajouter un nœud pour les faire évoluer, et une panne éventuelle sur l'un n'aura pas d'incidence sur les autres.

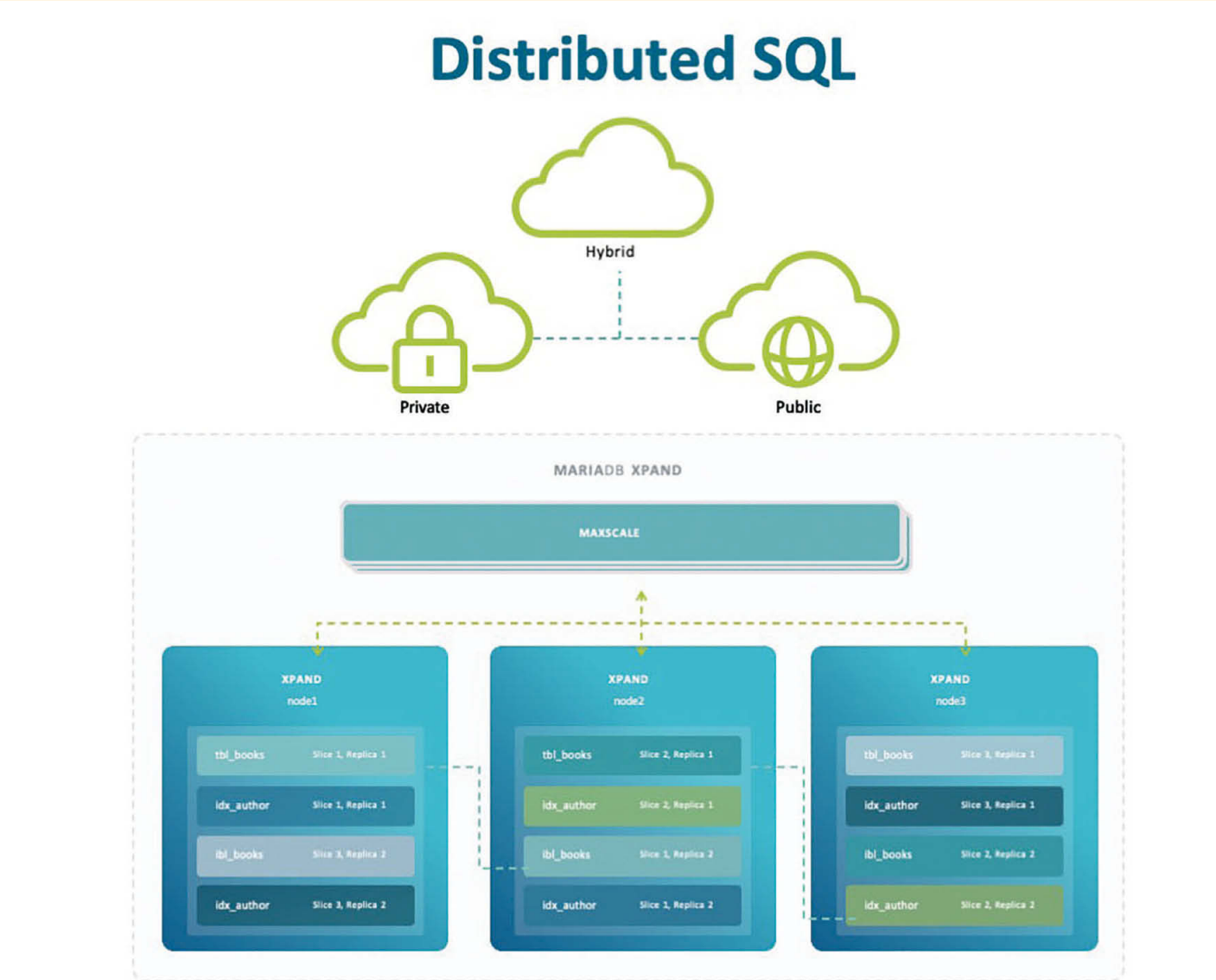
Une base de données SQL distribuée comporte plusieurs instances qui fonctionnent simultanément et indépendamment. Elle s'étend sur plusieurs nœuds d'un cluster qui sont distribués dans un ou plusieurs centres de données. La base de données distribuée tire ensuite parti des conditions physiques pour distribuer des partitions de données en ménageant une bonne redondance. Une panne au niveau d'un centre de données ne rend pas la base de données inopérante ni incohérente dans la mesure où des nœuds supplémentaires peuvent être déployés pour restaurer les performances escomptées. Le système distribue automatiquement à ces instances les données nouvelles et modifiées ainsi que les requêtes, équilibre la charge et synchronise les contenus de base de données.

De ce fait, un nœud unique ne peut pas se transformer en goulot d'étranglement, ce qui garantit l'homogénéité ainsi que de hautes performances et une forte disponibilité. Par ailleurs, le système distribue automatiquement les requêtes vers plusieurs nœuds ce qui, là encore, évite les goulots d'étranglement. Il vérifie en permanence si la distribution des données est optimale. En cas d'évolution du comportement des utilisateurs et de la charge de travail, le système peut transférer des données entre différents nœuds pour améliorer les performances. Parmi les utilisateurs phares de bases de données SQL distribuées pour les applications, nous avons l'un des principaux fabricants d'équipements mobiles. Les applications exigent une évolutivité massive, par exemple pour authentifier chaque utilisateur dans son Cloud. Plusieurs centaines de millions de clients utilisent ces services dans le Cloud. Les bases de données traitent plusieurs milliards d'instructions chaque jour et stockent des centaines de pétaoctets de données.

Sécurité intégrée et autres avantages de SQL distribué

SQL distribué utilise en outre une architecture dite sans partage dans laquelle chaque nœud fonctionne séparément. Ainsi, la panne d'un système n'empêche pas les autres de continuer à traiter comme d'habitude tous les accès en lecture et en écriture. Dès que le nœud redevient disponible, la base de données vérifie automatiquement que les modifications apportées aux données sont bien synchronisées sur l'ensemble des systèmes. Cette redondance rend le système nettement plus sécurisé que d'autres architectures de base de données. Pour l'utilisateur, la base de données est constamment disponible tout en étant facilement évolutive.

Par exemple, certaines entreprises peuvent avoir besoin sur une courte période d'une centaine d'instances de base de données pour faire face à un trafic de pointe. Avec des



bases de données SQL distribuées dans le Cloud, il suffit de cliquer sur un bouton pour ajouter et supprimer des instances dans l'heure. De plus, les données sont toujours distribuées équitablement, notamment lors de l'ajout ou de la suppression de nœuds. Grâce à cette approche de la distribution automatique des données, il n'y a plus lieu de gérer les divers fragments individuellement.

Les bases de données SQL distribuées présentent un double avantage évident pour les opérations informatiques : les systèmes offrent toutes les fonctionnalités importantes du modèle relationnel ainsi qu'un langage standard de base de données connu. Cette familiarité et la possibilité de tirer parti des compétences existantes raccourcissent très nettement la phase de déploiement du système.

SQL distribué dans la pratique d'entreprise

Une architecture de base de données SQL distribuée est adaptée à différentes industries, dont le e-commerce. En effet, une importante campagne de promotions ou une augmentation saisonnière de la demande peut faire exploser la fréquentation d'une boutique en ligne. Les cybermarchands doivent alors allouer assez de capacité pour gérer la ruée pendant des journées promotionnelles telles que les Black Friday ou en période de Noël.

Certains secteurs d'activité enregistrent quant à eux des fluctuations régulières au niveau de l'accès aux bases de données. Les services de livraison de repas sont par exemple très sollicités le soir à l'heure du dîner, tout le monde veut se faire livrer en même temps alors qu'en matinée la charge est inférieure à la moyenne. Les bases de données SQL distribuées dans le Cloud permettent d'ajouter autant de nœuds qu'ils le souhaitent lorsque les transactions sont les plus nombreuses et d'en supprimer pendant les périodes creuses.

Certains de ces services sont actifs partout dans le monde, le pic de performance se déplace sur 24 heures au rythme des fuseaux horaires dans tous les pays desservis. Autrement dit, l'heure de pointe ne s'arrête globalement jamais et génère de forts mouvements d'ondulation.

Les bases de données qui intègrent des fonctions SQL distribuées sont également intéressantes pour des applications dans l'Internet des objets industriels. La surveillance par des capteurs de l'état des machines et des systèmes génère de gros volumes de données, si bien que le stockage des données en local s'avère bien plus efficace. La distribution de la base de données crée en même temps un centre de données à des fins d'analyse, par exemple via l'apprentissage automatique.

Ces exemples démontrent clairement que SQL distribué est une architecture de base de données flexible et adaptée à toutes les exigences de la transformation numérique. Ce type de base de données convient à quantité de secteurs d'activité et de scénarios d'application auxquels il garantit dynamisme et performances.

OvhCloud : une action collective des clients lésés

OvhCloud s'est efforcé de faire de son mieux pour limiter l'impact de l'incendie de son datacenter sur ses clients. Mais pour certains, cela ne suffit pas : 140 entreprises clientes de la société ont ainsi annoncé leur intention de former un recours collectif afin d'exiger des dommages et intérêts suite à la perte de leurs données, emportées par les flammes. Les avocats évaluent le préjudice total à environ 10 millions d'euros et annoncent avoir commencé à envoyer les mises en demeure.

La numéro 2 de Meta quitte son poste

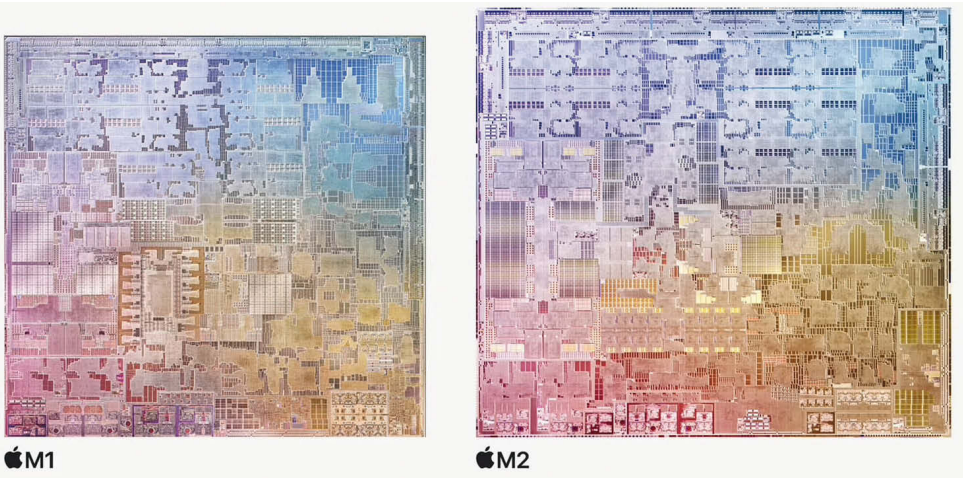
Sheryl Sandberg ne sera plus la directrice d'exploitation du groupe de Mark Zuckerberg à compter de cet automne. La dirigeante a annoncé son départ dans un post Facebook, et sera remplacée par l'actuel directeur de la croissance de l'éditeur, Javier Olivan. Elle explique vouloir recentrer son activité sur « sa fondation et son travail philanthropiques » dans les années à venir. Mais selon le Wall Street Journal, la raison serait tout autre : une enquête interne aurait été ouverte sur son cas depuis l'automne 2021, la directrice est soupçonnée d'avoir profité des ressources de l'entreprise pour ses projets personnels.

Binance dans le viseur de plusieurs enquêtes

La bourse d'échange de cryptomonnaie Binance s'est taillée une place de premier plan sur le secteur et n'hésite pas à manifester ses ambitions de s'imposer comme un leader du marché. Mais cette success-story attire également l'attention : on a ainsi appris que plusieurs enquêtes avaient été ouvertes par l'autorité américaine des marchés financiers, la SEC, sur les activités du groupe hongkongais. Un malheur n'arrivant jamais seul, Reuters a également publié ce mois-ci une enquête montrant comment plusieurs groupes cybercriminels ont utilisé la plateforme pour blanchir des sommes obtenues grâce à des ransomware ou des places de marché clandestines.

Apple présente sa nouvelle génération de puces ARM

La conférence développeur d'Apple a eu lieu au début du mois de juin et le constructeur a profité de l'occasion pour officialiser sa nouvelle génération de SoC ARM, la ligne M2. Celle-ci fait suite à la première tentative d'Apple en la matière, le processeur M1. Cette nouvelle version sera intégrée sur les nouvelles machines annoncées par la société de Tim Cook : le MacBook Air et le MacBook pro 13 pouces. Sans surprise, Apple promet des performances accrues et une consommation énergétique moindre que la génération précédente et confirme par la même que sa rupture avec l'écosystème d'Intel n'était pas qu'une passade.



M1 M2

Fr-Alert, pour faire oublier la débâcle de SAIP

À compter de la fin du mois de juin, les autorités françaises expérimenteront un nouveau dispositif d'alerte des populations via la technique dite du « cell broadcast ». Cette technologie permet d'envoyer un message d'alerte à l'ensemble des téléphones connectés sur une zone géographique définie, et devrait être utilisée pour avertir les citoyens d'un potentiel danger ou d'une menace imminente. La dernière tentative en ce sens, l'application SAIP, avait connu un échec retentissant : celle-ci prenait la forme d'une application à télécharger et n'était pas aussi réactive qu'escomptée. Le cell broadcast permettra au contraire de toucher l'ensemble de la population disposant d'un smartphone, sans qu'il leur soit nécessaire de télécharger l'application.

Internet Explorer tire sa révérence, pour de bon cette fois

Microsoft Internet Explorer

Après environ 27 ans de services, le navigateur web historique de Microsoft raccroche pour de bon et a cessé de fonctionner à compter du 15 juin 2022 sur les systèmes d'exploitation Windows. Cela fait bien longtemps que Microsoft a programmé la fin de vie de son navigateur et l'éditeur propose son navigateur Edge en remplacement, mais la popularité passée d'Internet Explorer l'a rendu essentiel pour de nombreux équipements et applications difficiles à remplacer. Ces questions de compatibilité ont d'ailleurs poussé Microsoft à développer un mode « Internet Explorer » au sein de son navigateur Edge, dont le support sera assuré jusqu'en 2029.

Atos remercie son DG et envisage une scission

Après à peine 6 mois à la tête d'Atos, le directeur général Rodolphe Belmer jette l'éponge et annonce son départ de l'entreprise au 30 septembre 2022. La cause de ce départ : la décision des actionnaires de scinder l'entreprise en deux entités distinctes, chacune cotée en bourse. Si le plan se passe comme prévu, la société SpinCo (Evidian) récupérera les activités transformation digitale, big data, cybersécurité. De l'autre, TFCo (Atos) conservera les activités d'infogérance et de gestion de parc de la société.

Oracle APEX : construire une application à l'échelle d'une pandémie

Marc Gueury

Le 18 mars 2020, un de mes collègues, Shakeeb Rahman a reçu un SMS lui demandant de joindre un appel Zoom. Tout de suite. En réponse à l'incertitude au début de la pandémie, Oracle avait proposé au gouvernement américain un système pour comprendre l'efficacité des premiers traitements COVID. Nous devions le développer en quelques jours.

Dans ce monde d'incertitude, notre meilleure arme est la donnée. Nos efforts se sont transformés en une suite de systèmes qui permettent le suivi des traitements précoces du COVID, la sollicitation de volontaires pour les essais cliniques, l'échange des dossiers de vaccination aux États-Unis, la commande de vaccins… Nous avons travaillé en étroite collaboration avec plusieurs agences gouvernementales aux États-Unis, notamment avec Health and Human Services (HHS), National Institutes of Health (NIH), ou US Centers for Disease Control and Prevention (CDC), …

Les critères de réussite étaient :

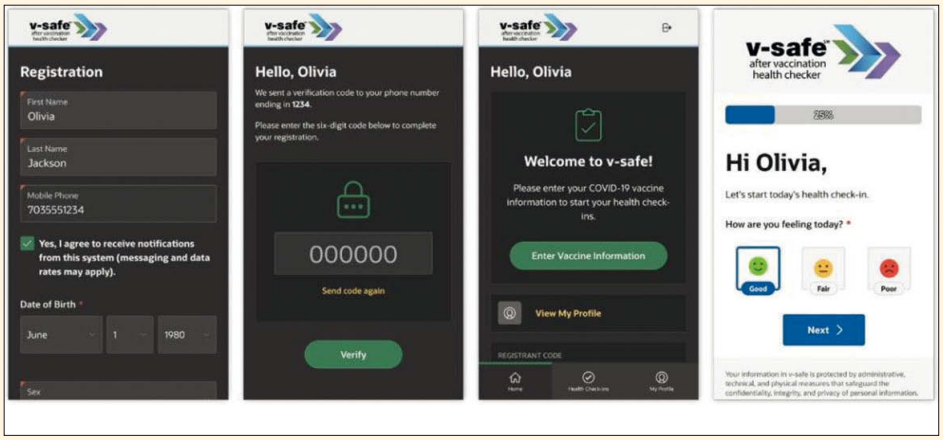
- **La sécurité** : Ces systèmes traitent des informations de santé, qui exigent les plus hauts niveaux de confidentialité et de sécurité.
- **L'évolutivité** : Ces systèmes doivent gérer un grand nombre de données à chaque stade de la pandémie, être faciles à utiliser, capables de servir des millions d'utilisateurs, tout le temps.
- **La rapidité** : On en avait besoin tout de suite. La rapidité était une priorité absolue. Les délais étaient fixés par la fabrication des vaccins, les réglementations …

Notre approche

Notre approche comprenait 3 éléments clés : APEX, notre plate-forme Low Code, la base de données Oracle et OCI.

Fonctionnement d'Oracle APEX

Avec APEX, nous avons donné le ton en développant des applications en quelques jours et en organisant des réunions quasi quotidiennes avec les parties prenantes. Nous y faisions la démonstration des nouvelles versions et recueillions les commentaires. À plusieurs reprises, nous avons participé à des réunions Zoom avec Larry Ellison et de hauts responsables d'organismes de santé publique pour développer l'application en temps réel. En une heure de conférence, nous avons accompli ce qui aurait pris plusieurs jours ou plusieurs semaines avec un approche classique. Ce projet n'était possible qu'avec une plate-forme telle qu'APEX, qui élimine une grande partie de la complexité inhérente au développement et permet de se concentrer sur le problème à résoudre.



Base de données Oracle

La base de données a été déployée sur Oracle Exadata pour répondre aux besoins de performance d'envergure pandémique.

Oracle Cloud Infrastructure (OCI)

Le cloud d'Oracle est un cloud moderne, très performant, extensible, et conforme aux normes gouvernementales strictes FedRAMP telles que IL5 PATO, SOC, ISO, HIPAA…

Faire partie de la solution

Dix jours après le premier SMS, le programme était en ligne. Mais personne ne s'attendait à mettre fin à la pandémie avec un seul système. Nous étions déjà en conversation avec plusieurs agences au sujet des problèmes ultérieurs à résoudre. Depuis lors, nous avons créé une série de systèmes qui nous aident à comprendre la propagation du COVID, à gérer l'échange de tous les dossiers de vaccination aux États-Unis, à solliciter des volontaires pour les essais cliniques de vaccins, à comprendre la sécurité des vaccins et à cartographier la propagation des nouveaux variants du COVID, à commander des vaccins, … L'expertise principale d'Oracle consiste à développer de grands systèmes de données. Nous n'avons pas eu besoin de construire APEX à partir de zéro, ou une base de données à l'échelle du cloud ou une infrastructure cloud. Nous avons simplement utilisé des solutions existantes.

Succès

V-safe avec plus de 1,2 million d'utilisateurs actifs quotidiens a permis de contrôler la sécurité des vaccins COVID en quasi-temps réel, de remplir des rapports VAERS en contactant les personnes vaccinées présentant des effets indésirables, et de constater que les vaccins COVID sont sans danger pour les femmes enceintes. V-safe contribue aussi à la réalisation d'un plus grand nombre d'études complémentaires pour comprendre la sécurité et l'efficacité de ces vaccins.

Conclusion

Nous avons pu mettre le plus jeune des clouds publics à l'épreuve en aidant à résoudre l'un des problèmes les plus importants de notre génération. La base de données Oracle n'est pas nouvelle. Depuis 40 ans, les clients l'utilisent dans le monde entier pour exécuter des systèmes essentiels. Oracle APEX n'est pas nouveau non plus. Les clients l'utilisent dans tous les secteurs d'activité depuis 20 ans.

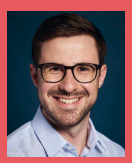
OCI est nouveau : ce cloud de nouvelle génération fait ses preuves dans tous les domaines, pour exécuter des systèmes essentiels, et ce, depuis le peu de temps qu'il est sur le marché. La combinaison de ces trois technologies nous a permis de développer des solutions de lutte contre la pandémie qui sont sécurisées, évolutives pour prendre en charge des centaines de millions d'utilisateurs, et qui ont été développées à une vitesse fulgurante.

Les 6 soft skills incontournables pour booster sa carrière de développeur

Les soft skills (que l'on peut traduire par "compétences douces") sont les compétences issues de l'intelligence émotionnelle et situationnelle qui régissent les rapports humains. Complémentaires aux hard skills qui se concentrent sur l'aspect technique et le savoir-faire, les soft skills ont longtemps été mises au second plan par les professionnels de l'informatique avant de faire un retour flamboyant sur le devant à partir des années 2010.

Rosalie Zandona

Après un parcours universitaire en psychologie, je me suis orientée vers les ressources humaines. J'ai fait mes premières armes pendant 7 ans chez CGI au Luxembourg, avant de finalement devenir HR Manager chez SFEIR. J'interviens ponctuellement en tant qu'enseignante vacataire dans plusieurs universités de Lorraine pour partager mon expérience avec les générations futures.



Nicolas Haag

Ingénieur en informatique de formation, j'ai très vite basculé vers la gestion de projets. Après un passage par Enedis à Lyon en tant que Product Owner, je suis arrivé au Luxembourg comme IT Project Manager chez KPMG. Dorénavant Delivery Manager chez SFEIR, je suis en charge du centre de services de l'entité luxembourgeoise.

Que ce soit lors d'un entretien d'embauche, d'un atelier fonctionnel, d'un stand-up meeting ou lors de son entretien annuel avec son manager, les soft skills sont devenues essentielles au point de devenir un critère de recrutement incontournable. Voyons ensemble les compétences que les recruteurs s'arrachent.

1 Faire-savoir

En entreprise, il est bien sûr nécessaire d'accumuler du savoir-faire, mais les compétences "brutes" ne suffisent pas si vous voulez donner un coup de fouet à votre carrière. Il est tout aussi important de faire en sorte que les autres vous considèrent comme compétent : c'est ce qu'on appelle le "faire-savoir".

Un développeur qui a du faire-savoir est un développeur qui rayonne. Pour être visible, vous devez faire en sorte d'être sous les feux de la rampe et que les gens autour de vous reconnaissent la valeur que vous apportez. Pour y parvenir, prenez l'habitude de parler en public, de prendre des initiatives et de faire des propositions. N'oubliez pas non plus de célébrer vos petites victoires !

Concrètement, cela signifie par exemple que vous pouvez vous porter volontaire pour certaines actions, même si elles ne relèvent pas de votre domaine de compétence. C'est aussi souvent une bonne idée de proposer votre aide sur un sujet que vous connaissez bien, même si on ne vous l'a pas demandé. Cela peut d'ailleurs être le cas pour des sujets non techniques, comme la correction orthographique ou le graphisme.

Pensez également à soigner votre image publique. On appelle cela le "personal branding", un sujet encore tabou chez nous en Europe, mais très bien développé outre-Atlantique. En développant votre image, vous aurez un meilleur contrôle sur votre carrière. Pour cela, il existe de nombreuses possibilités : écrire des posts sur LinkedIn, tenir un blog technique, contribuer à des projets publics, etc. Chez SFEIR, nous encourageons les consultants à devenir trainer pour des formations spécifiques et également des speakers lors de conférences techniques. Nous ne pouvons que vous encourager à suivre cette voie !

Veillez toutefois à rester modeste. Le plus dur dans le personal branding n'est pas tant de créer du contenu, mais plutôt de mettre le curseur au bon endroit entre pas assez de lumière et trop de lumière. Attention cependant : pour éviter de tomber dans l'arrogance, pensez à également mettre les autres en avant, de temps en temps.

2 Sens du collectif

Avoir le sens du collectif, c'est savoir écouter l'autre, éprouver un minimum d'empathie et être solidaire. Il s'agit également d'avoir l'envie profonde que chacun s'épanouisse au contact de l'autre et au sein de son travail, et surtout de servir l'intérêt de son équipe avant le sien. Avoir le sens du collectif vous permet de résoudre des problèmes complexes.

Plusieurs études psychologiques (Kurt Lewin, 1948, par exemple) ont démontré l'importance de la "dynamique de groupe". La mise en production d'un projet est rarement la résultante d'un travail individuel, mais bien d'un travail d'équipe. Par exemple, si chacun ne se soucie pas uniquement de ses tâches et corrige au fur et à mesure les bugs des autres, alors le projet avance de façon plus performante.

En outre, en psychologie, on parle de "mémoire transactive" (Daniel Wegner, 1985). Il s'agit de la mémoire de groupe, celle qui permet de se reposer sur les savoirs que son collègue maîtrise parfaitement, sans encombrer la sienne d'éléments hors de sa spécialité. C'est une sorte de réserve de connaissances. Avoir la capacité de savoir qui peut vous aider à trouver une solution, et comment il peut le faire peut être un énorme avantage. Tout comme il est important de pouvoir partager son savoir-faire à l'ensemble de l'équipe. Et c'est là toute la puissance du collectif !

En somme, regardez-la "big picture". Par exemple, lorsqu'on assiste à un match de hockey sur glace, si on se concentre sur le palet uniquement, on est perdu en quelques secondes. En revanche, si on prend du recul et qu'on se concentre sur les mouvements des joueurs, tout devient beaucoup plus clair.

3 Rigueur du développement

Le diable est dans les détails. Cela est d'autant plus vrai dans le monde du développement, où la différence entre un amateur et un pro se cache souvent dans les finitions.

Faire preuve de rigueur en développant conduit à une meilleure qualité, car cela signifie que l'on vérifie minutieusement son travail avant de le livrer. Ce n'est pas parce que vous avez la chance d'avoir un recetteur fonctionnel dans votre équipe que vous pouvez vous permettre de ne pas être irréprochable. Dès que les spécifications sont écrites, le développeur devient la première ligne de défense contre les anomalies.

N'oubliez pas d'ailleurs que plus une erreur est découverte tard, plus elle coûte cher. L'exemple le plus parlant à ce sujet est le fiasco du Samsung Note 7, dont le

problème de batterie a causé une perte estimée à 17 milliards $. Si le souci d'alimentation avait été découvert pendant la phase d'implémentation, il aurait coûté au moins 10 000 fois moins cher.
Comment pouvez-vous donc concrètement améliorer votre rigueur en matière de développement ? Une approche intéressante est ce que l'on appelle le "pessimisme professionnel", que l'on pourrait comparer à la loi de Murphy. Autrement dit, il vaut mieux partir du principe que tout ce qui est susceptible d'aller mal ira mal, et s'y préparer. Si vous n'êtes pas sûr, rapprochez-vous de la personne qui a écrit les spécifications et demandez-lui quels sont les cas aux limites.
Mais n'oubliez pas que la rigueur ne consiste pas seulement à faire les choses correctement, mais aussi à les faire à 100 %, sans rien mettre sous le tapis. Pour ce faire, il est de bon conseil de relire les spécifications une dernière fois avant de livrer, par sécurité. Si vous avez la chance d'avoir des critères d'acceptance dans vos tickets, lisez-les attentivement. Et n'oubliez pas que de nombreuses petites tâches mentionnées oralement peuvent avoir leur importance. Ayez donc toujours un carnet sur vous et notez les différents ajustements qui vous sont demandés. Même si votre mémoire est infaillible, ce ne sera jamais perdu et cela rassure toutes les parties prenantes, y compris vous. N'oubliez pas que notre cerveau est une RAM, pas une ROM.

4 Flexibilité

La flexibilité est une compétence importante, car elle témoigne de la capacité et de la volonté de relever de nouveaux défis avec sérénité et sans drame.
Un développeur flexible est toujours prêt à donner un coup de main là où c'est nécessaire et à s'adapter rapidement lorsque les plans changent. Il embrasse la pluralité et la diversité des tâches. Il accepte que sa journée ne se déroule pas comme il l'avait imaginé, voire il apprécie même cette imprévisibilité.
La flexibilité, tout comme les autres soft skills présentées ici, est une compétence qui n'est pas innée. Le meilleur moyen de s'améliorer est de s'efforcer chaque jour à sortir de sa zone de confort. Sur le lieu de travail, cela peut se traduire par l'acceptation de nouvelles responsabilités, même si elles n'ont rien à voir avec la description de poste. Par exemple, pourquoi ne pas devenir secouriste du travail ? Ou bien organiser des sessions basketball sur la pause de midi ?
Pour être flexible, vous devriez également être ouvert à des solutions alternatives si vos propositions ne sont pas acceptées. Il y a toujours plus d'une bonne réponse à un problème donné. Le but consiste simplement à trouver celle qui vous convient le mieux et avec laquelle vous pouvez vivre.
Il est crucial de savoir dire "non", mais encore faut-il savoir comment le faire. Cela pourrait faire l'objet d'un article à part entière, mais voici quelques conseils élémentaires.
Si vous ne pouvez pas assumer une tâche particulière parce qu'elle ne relève pas de votre domaine de compétence, ne vous contentez pas de dire "non", mais aiguillez plutôt votre interlocuteur vers la bonne personne ou le bon service. En revanche, si vous devez refuser une demande par manque de temps, au lieu de dire "non", demandez toujours d'abord "pour quand ?". Si votre interlocuteur a une échéance proche, proposez une solution de contournement à son problème ou suggérez des pistes de solution. S'il a une vision à plus long terme, ajoutez cette tâche à votre to-do list, sans vous engager.
Comme vous l'aurez remarqué, le mot "non" ne doit être utilisé qu'en dernier recours, car on ne construit pas une entreprise avec des "non".

5 Acceptation de la critique

Il est très important d'accepter que nous ne soyons pas parfaits. Et en tant qu'êtres imparfaits, nous sommes exposés à la critique. Qu'elle soit justifiée ou non, le plus important est d'être capable de réagir convenablement et d'être humble.
Si la critique est vraiment trop acerbe, n'hésitez pas à sortir et à faire quelques exercices de respiration. C'est toujours mieux que de réagir de manière non professionnelle et de le regretter une fois chez soi.
N'oubliez pas que la critique doit vous pousser à vous améliorer, tant pour votre carrière que pour le projet. Elle peut être perçue comme négative, mais elle a pourtant en général un effet positif. Parfois, la critique est mal présentée ou non constructive, mais dans un contexte professionnel, il n'est pas approprié de la prendre personnellement.
Essayez également de vous mettre à la place de vos interlocuteurs. Pensez-vous qu'ils y prennent plaisir ? Il est probable qu'ils se sentent encore plus mal à l'aise que vous, alors il est inutile d'aggraver la situation.
Rien ne vous empêche cependant de contester ces critiques. Vous pouvez y réfléchir objectivement à tête reposée et même demander l'avis d'autres collaborateurs. Et si, après une réflexion sereine, vous êtes toujours convaincu que la critique n'est pas justifiée, rien ne vous empêche de demander un nouvel entretien avec votre interlocuteur pour lui exposer votre point de vue. Concentrez-vous simplement sur les faits, pas vos impressions ou vos sentiments.

6 Goût de l'effort

Le mot "effort" a souvent une connotation négative, et pourtant, nous faisons des efforts tous les jours : pour sortir du lit, pour arriver à l'heure au travail, pour ne pas taper sur le collègue qui siffle tout le temps au bureau... Mais avez-vous déjà remarqué que certaines personnes sont plus enthousiastes que d'autres à l'idée de faire des efforts ? Que certains sont capables de se lever aux aurores tous les matins pour aller courir, juste pour leur bien-être intérieur ?
Sachez que tout cela peut être cultivé. L'effort appelle l'effort, et c'est une caractéristique qui concerne aussi bien la vie professionnelle que la vie privée.
Tout d'abord, faites du sport. Vraiment. Plus vous en ferez, plus vous y prendrez plaisir, c'est garanti. Encore faut-il trouver le sport qui vous convient ! Ce n'est pas parce que la course à pied vous rebute que vous devez abandonner, car il existe de nombreuses autres façons de faire de l'exercice. Dans tous les cas, l'effort physique vous pousse à vous dépasser constamment et à donner le meilleur de vous-mêmes. Vous verrez que, petit à petit, vous vous dépasserez aussi dans d'autres domaines de la vie quotidienne : vous ferez la vaisselle dès que vous aurez fini de manger, vous vous lèverez dès que votre réveil sonnera... et vous serez plus productif au travail !
Au-delà de l'effort physique, il est important de tout donner et de faire les choses dans l'immédiat, sans les remettre à plus tard, même s'il n'y a pas d'urgence. À ce sujet, un amiral américain, William McRaven, a prononcé un étonnant discours à l'Université du Texas en 2014, dans lequel il disait que pour changer le monde, il fallait commencer sa journée par... faire son lit.
"Si vous faites votre lit chaque matin, vous aurez accompli la première tâche de la journée. Ceci vous donnera un petit sentiment de fierté qui vous encouragera à faire une autre tâche puis une autre et encore une autre. À la fin de la journée, cette tâche que vous aurez accomplie se sera transformée en de nombreuses tâches achevées. Faire votre lit renforcera également le constat que les petites choses dans la vie comptent. Si vous ne pouvez pas bien faire les petites choses, vous ne ferez jamais bien les grandes."

Conclusion

Ces quelques conseils ne sauraient être exhaustifs. Si vous maîtrisez ces soft skills, cela ne veut pas nécessairement dire que vous allez être promu en un claquement de doigts. De même, ce n'est pas parce que vous pêchez encore sur certaines de ces compétences que rien de positif ne vous arrivera.
L'important est que vous développiez un état d'esprit qui met l'accent sur les compétences douces tout autant que sur les compétences techniques. Ayez en tête que si un manager a la possibilité de promouvoir un de ses collaborateurs, il choisira rarement le meilleur ou le plus efficace, mais plutôt celui en lequel il a le plus confiance. Et la confiance, ce n'est pas une compétence technique.

Minitel : réactivez vos terminaux

photo : A1AA1A

Déjà l'été ! C'est une tradition à Programmez!, en été, c'est un numéro un peu spécial. La preuve avec notre dossier spécial soleil et électronique : sortez le minitel et hackez-le ! « Bah, non, pas possible ! » ou encore « Minitel ? C'est quoi ? ».

Sans rentrer dans les détails, un prochain numéro de Technosaures reviendra sur l'histoire du Minitel et les conflits autour de l'objet, le Minitel est l'expression de la télématique poussée par la France à la fin des années 1970. Les investissements colossaux dans le réseau télécom permettent à la France de rattraper son retard technique et technologique. Mais comment rentabiliser ce réseau tout neuf ?

Le minitel est un terminal passif. Car finalement, il ne fait pas grand-chose à part recevoir les données et les afficher. Aucun traitement local n'est réalisé. Des dizaines de millions de Minitel furent installés.

Pour pouvoir transporter les données et communiquer, il faut un réseau public et un protocole : Transpac est le réseau et X25, le protocole. Une guerre entre les R&D publiques et les différentes institutions des télécoms et recherches fait rage dans les années 70. Le point chaud fut le projet de recherche Cyclades qui peut être vu comme une fondation d'Internet et de TCP – TCP/IP. Sans rentrer dans les détails, et les débats parfois très vifs, Cyclades, malgré son caractère décentralisé et sa capacité à être agnostique des ordinateurs et terminaux, reste fondamentalement un projet de recherche qui n'a pas vocation à être déployé ni commercialisé. Transpac va alors s'imposer avec X25 comme protocole de communication.

Le Minitel utilise ce protocole pour pouvoir se connecter aux commutateurs et aux serveurs. X25 couvre trois couches du modèle OSI : couche physique, couche liaison et couche réseau. Les transmissions restent lentes. Les premiers modèles utilisent des modems 1200 bauds, les dernières évolutions jusqu'à 9600 bauds !

Le minitel est donc un terminal téléinformatique qui supportent l'affichage et la connexion aux serveurs Télétel et Téléinformatique. Le Télétel est un standard qui supporte le Vidéotext (texte et graphisme) et un mode mixte supportant des jeux de caractères ASCII. Par défaut, Vidéotext est le mode d'affichage. Télétel est comme un réseau au sens informatique du terme. C'est sur ce réseau que fonctionne les fameux 3614, 3615, etc.

Carte UT Modem du Minitel 1B (modèle Matra)

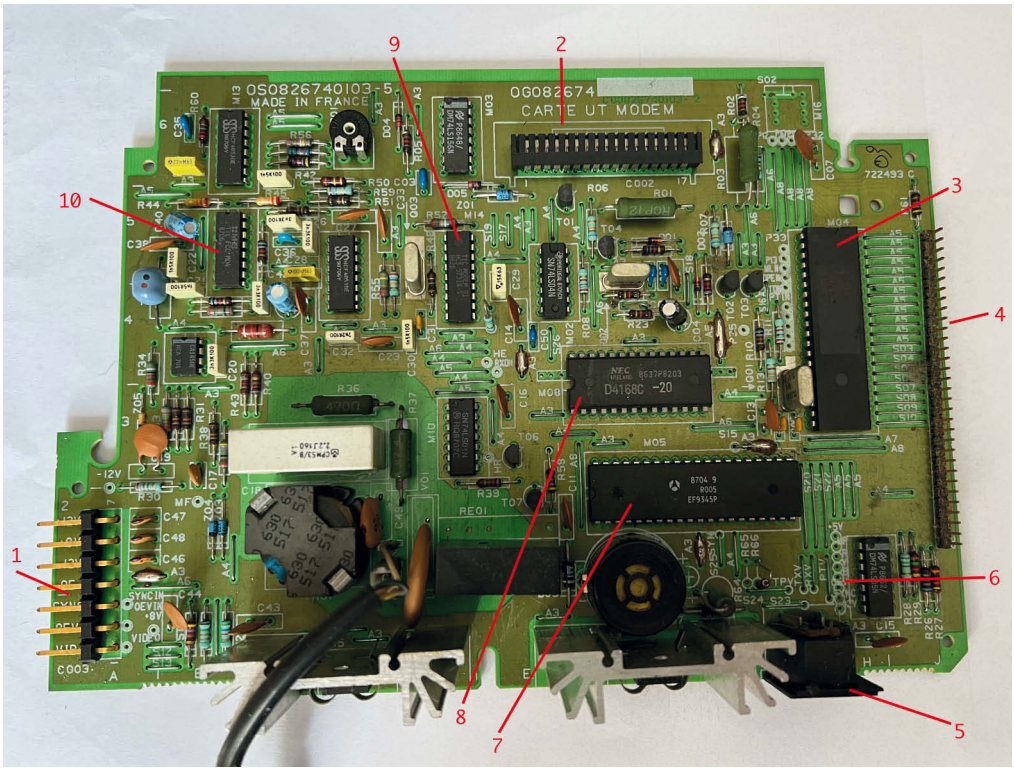
François Tonic

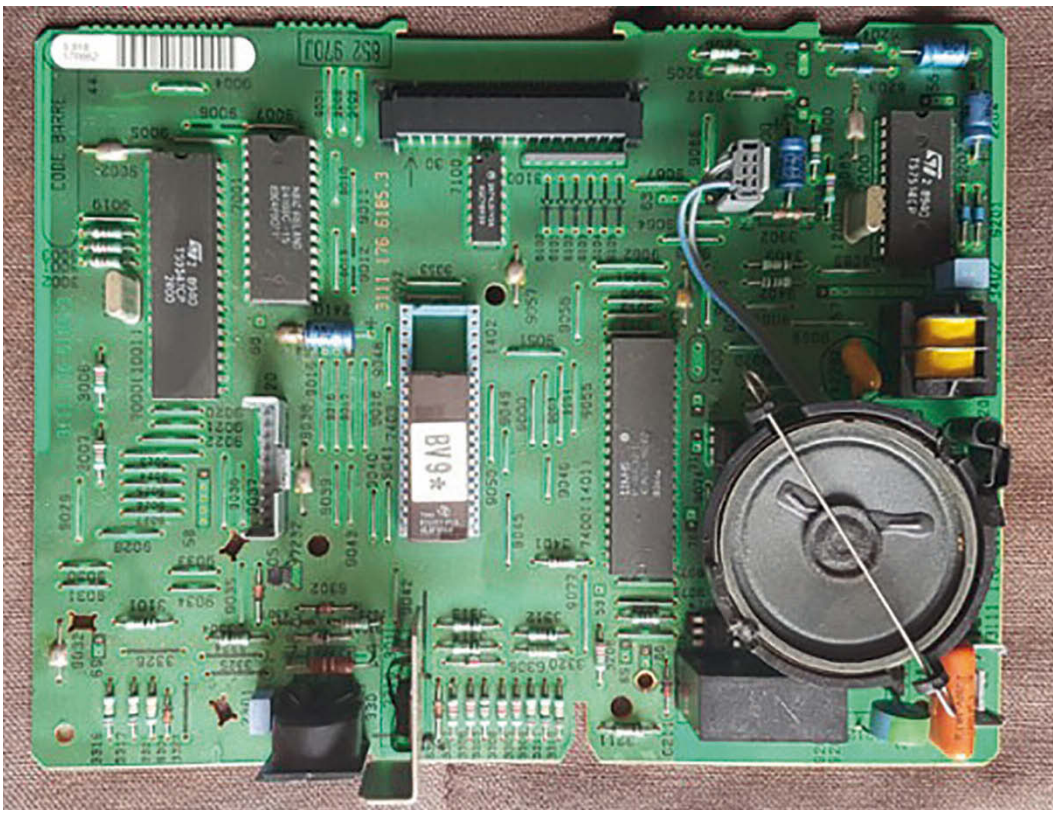
Quelques éléments techniques

1. Connecteur pour la carte vidéo / alimentation
2. Connecteur clavier
3. CPU Intel 8052 avec 8 Ko de ROM, 256 octets de RAM
4. Connecteur d'extension
5. Prise DIN
6. broches pour une sortie vidéo péritel (headers / câble non soudés)
7. EF9345 : processeur d'affichage couleur : oui le Minitel sait afficher la couleur !
8. Module mémoire 4168 (8 Ko de RAM) pour le processeur graphique
9. TIC Model HC3 5731R-5 : modem
10. TIC FCC M04 8706 : aucune référence trouvée

Une petite carte fixée à l'arrière du boîtier complète l'alimentation avec un ensemble de filtres et d'un fusible 630mA.

Sur la carte vidéo / alimentation, on notera les gros condensateurs et des rajouts de diodes sur la face côté boîtier. Sans doute pour corriger les problèmes vus après la fabrication.

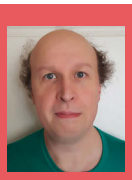




Minitel 2

La carte du Minitel 2 est très différente du Minitel 1b. Nous trouvons une EPROM de 32 Ko sur son support, un nouveau composant Intel (80C32) sans RAM ni ROM selon, les constructeurs. Les évolutions de la carte, le 24C02 permet de stocker le réperoire et le mot de passe, modem type V23.

Photo : jelora.fr



Jean-Christophe Quetin

J'ai débuté sur MO5 avec le LOGO et sa petite tortue (un des seuls langages en français). J'ai retrouvé l'électronique et la programmation avec le Raspberry Pi et l'Arduino. J'ai eu l'occasion de travailler avec des élèves de collège sur l'Arduino et d'écrire en 2018 mon premier livre (dont la seconde édition est parue en 2021).

Arduino - Apprivoisez l'électronique et le codage pour donner vie à vos projets *(2e édition) au Editions ENI.*

micro:bit - Programmez la carte avec MakeCode et MicroPython

twitter.com/jcquetin

arduiblog.com

youtube.com/channel/UCQXCp5srlwc8eCv0ULUjrdA

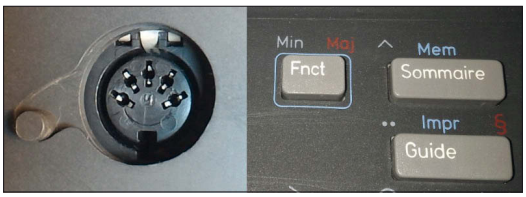
Ressuscitez le Minitel !

L'ère du Minitel aura quand même duré 30 ans. Pour simplifier (pour les plus jeunes), disons que le Minitel était l'ancêtre d'internet. C'était un petit terminal en noir et blanc qui permettait d'accéder à des services payants (Annuaire, Chat, Journaux et bien sûr les fameuses messageries roses). Pour les personnes qui l'ont connu, le minitel reste un engin mythique. Alors, pourquoi ne pas essayer de s'amuser en lui donnant une seconde vie ?

Choisir son Minitel

Le moins que l'on puisse dire, c'est que le Minitel est robuste. Au départ, il était prêté ou loué aux utilisateurs, il n'aurait donc pas été rentable qu'il tombe sans arrêt en panne. Résultat, un Minitel de 30 ans qui prend la poussière au fond d'un grenier a toutes les chances de s'allumer (après un petit coup de chiffon). On sait malheureusement que nos smartphones ne connaîtront pas le même destin…

On trouve sur internet ou dans les brocantes pour 20-30 euros des Minitel. Il faut évidemment s'assurer qu'il s'allume et que toutes les touches fonctionnent. Mais n'achetez pas n'importe quel modèle. Seuls les bistandards (1B et 2) peuvent être facilement transformés en terminal. Pour les reconnaître, c'est facile, ils possèdent (sur le clavier) une touche « Fnct » qui permet de changer la configuration du minitel. Mais aussi (à l'arrière), la fameuse prise DIN à 5 broches qui assurera la communication avec le Raspberry Pi.



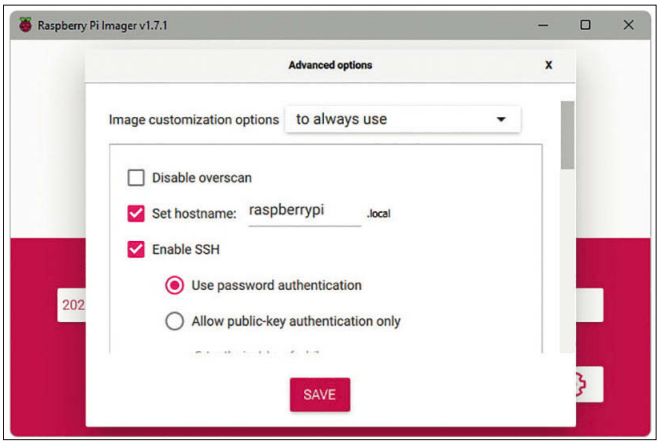
Installation du Rasberry Pi

Pour cela, vous n'avez besoin ni d'écran ni de clavier (ou de souris).

Préparation de la carte SD

Malheureusement, je n'ai pas réussi à faire fonctionner le Minitel avec la dernière version de Raspberry OS. Il existe certainement une solution, mais je ne la connais pas (au moment d'écrire cet article). Alors si vous la trouvez, n'hésitez pas à me contacter.

Figure 1



En attendant, je vous propose d'utiliser la dernière version qui fonctionne. Téléchargez donc la version ***Buster*** de ***Raspberry Pi OS Lite***, que vous trouverez à l'adresse suivante : https://downloads.raspberrypi.org/raspios_lite_armhf/images/raspios_lite_armhf-2021-05-28/

Décompressez le fichier zip et copiez l'image sur votre carte micro SD (le plus simple est certainement d'utiliser ***Raspberry Pi Imager***). Cliquez sur ***CHOISISSEZ L'OS***, puis sur ***Utiliser image personnalisée*** et sélectionnez l'image téléchargée.

Cliquez sur le petit engrenage pour activer le SSH, modifier le mot de passe et indiquer éventuellement vos paramètres WiFi. **Figure 1**

Sélectionnez la carte SD (dans ***Stockage***) et lancez la copie en cliquant sur ***ECRIRE***.

Contrôle du Raspberry Pi (en SSH)

Insérez la carte dans le Raspberry Pi et allumez-le.

Pour accéder au Raspberry Pi par SSH (depuis votre ordinateur), utilisez un émulateur de terminal (par exemple ***PuTTY***). Si vous connaissez l'adresse **IP** du Raspberry, vous pouvez l'indiquer, mais il est également possible d'indiquer le nom d'hôte : "***raspberrypi***" . Profitez-en pour enregistrer vos paramètres de connexion (pour la prochaine fois). **Figure 2**

La 1re connexion est un peu plus longue et ***PuTTY*** vous demandera certainement de valider la clé en cliquant sur "***Accept***". **Figure 3**

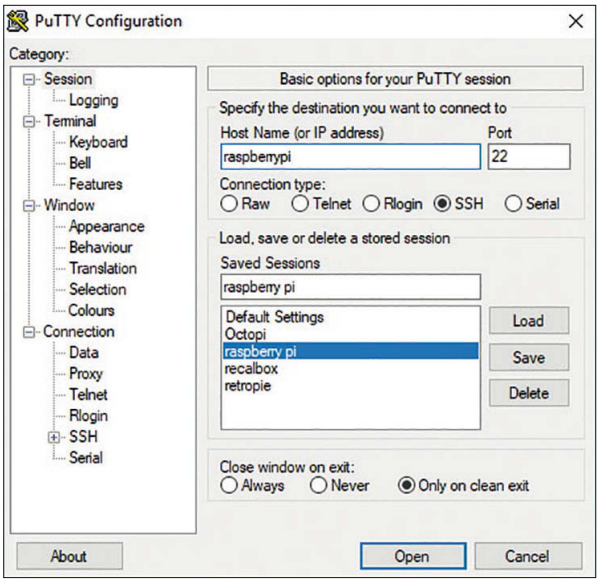


Figure 2

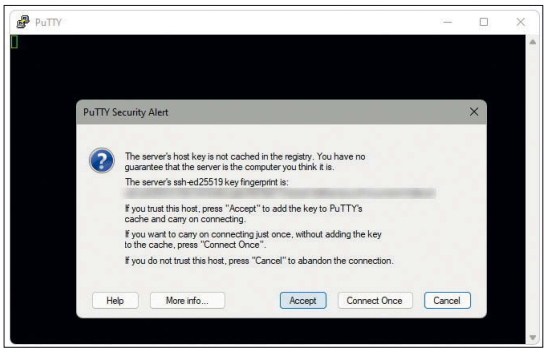
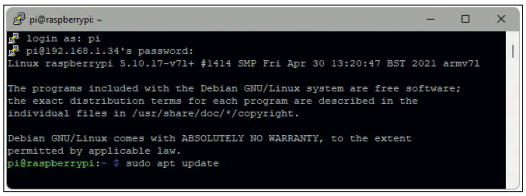


Figure 3

Par défaut le nom d'utilisateur est "***pi***" (et le mot de passe est celui que vous avez choisi).
Ensuite, vous pouvez mettre à jour Raspberry OS :

```
sudo apt update
sudo apt upgrade
```



Et redémarrez en tapant :

```
sudo reboot
```

Connexion du Minitel au Raspberry pi

Fabrication du câble

Le branchement n'est pas très compliqué, il suffit d'un transistor, de quelques résistances, mais surtout un convertisseur USB vers TTL série basée sur la puce PL2303HX. **Figure 4**
Il est également possible d'intégrer les composants à l'intérieur du minitel et de faire seulement ressortir la prise USB à la place du câble téléphonique (désormais inutile). **Figure 5**
Ensuite, vous pouvez brancher le minitel au Raspberry Pi.

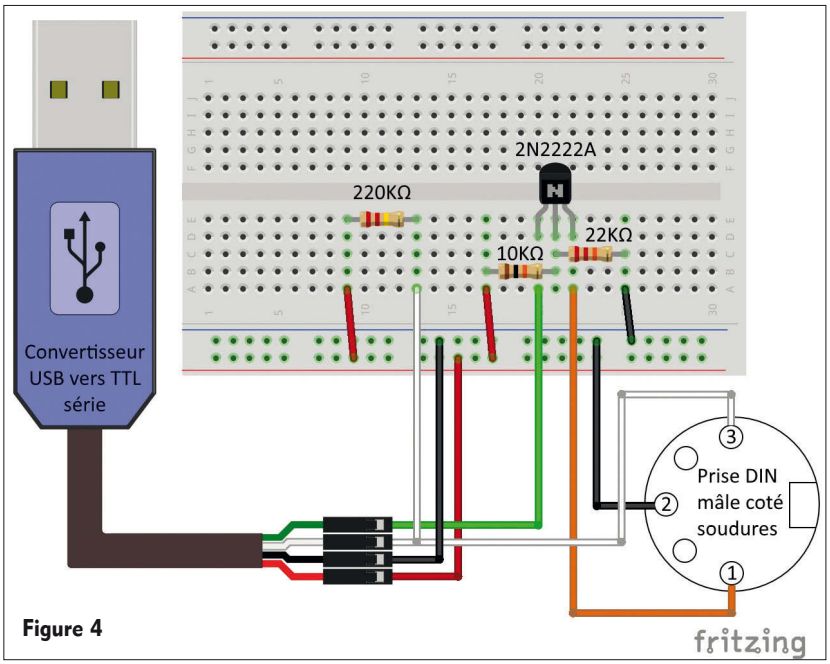


Figure 4

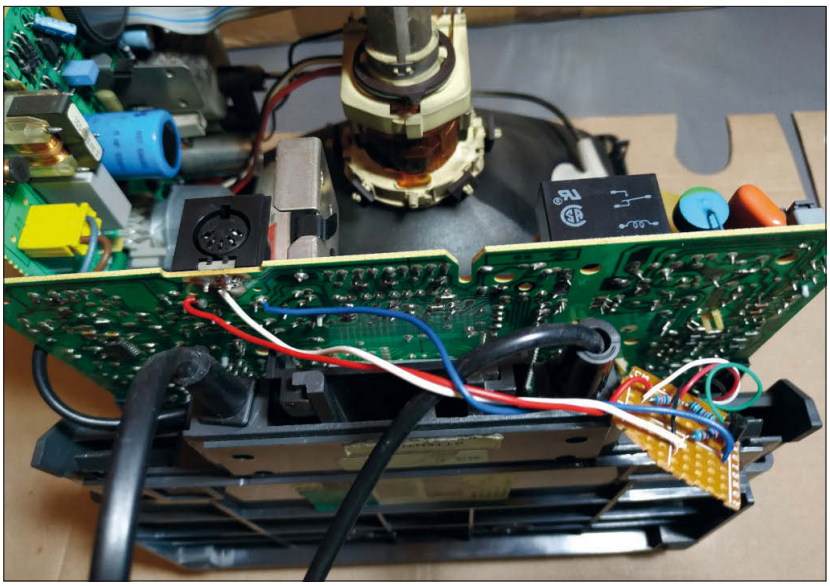


Figure 5

Préparation du minitel

Le minitel est malheureusement incapable de mémoriser les informations de connexion. Il faudra à chaque démarrage, refaire la manipulation suivante :

Fnct+Sommaire
(pour passer du mode répertoire au mode terminal)

Fnct+T
(en même temps), relâcher et ***A*** (passage en mode périphérique)

Fnct+T,
(en même temps) et ***E*** (désactivation de l'echo du terminal)

Fnct+P,
(en même temps), et ***4*** (connexion à 4800 bauds)

Avec le minitel 2, vous pouvez vous connecter à 9600 bauds en tapant Fnct+P, (en même temps), et 9 (à la place de 4), mais il faudra aussi modifier les paramètres du Raspberry Pi (ci-dessous).

Test de communication

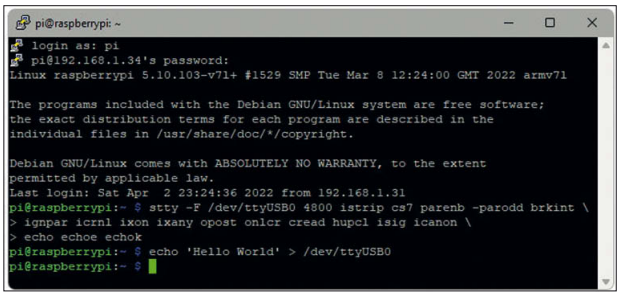
Branchez le minitel sur la prise USB du Raspberry Pi et allumez-les. Connectez-vous au Raspberry (avec PuTTY ou le client SSH de Windows). Copiez ces 3 lignes en même temps dans votre client SSH :

```
stty -F /dev/ttyUSB0 4800 istrip cs7 parenb -parodd brkint \
ignpar icrnl ixon ixany opost onlcr cread hupcl isig icanon \
echo echoe echok
```

puis :

```
echo 'Hello World' > /dev/ttyUSB0
```

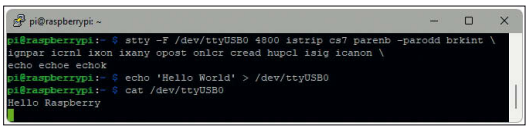
Figure 6



Nous allons maintenant tester la communication dans l'autre sens, tapez :

```
cat /dev/ttyUSB0
```

Sur le minitel, vous pouvez taper le texte que vous voulez (suivi de la touche entrée). Le texte devrait s'afficher dans le client SSH (tapez CTRL+C, pour quitter).



Pour la suite, je me suis surtout inspiré de l'excellent tuto de Maxime Vinzio :
http://sta6502.blogspot.com/2016/02/utiliser-un-minitel-comme-terminal-sur.html

Téléchargez le fichier mntl.ti d'Alexandre MONTARON (http://canal.chez.com/terminfo.htm).

```
wget http://canal.chez.com/mntl.ti
```

Et compilez-le :

```
tic mntl.ti -o /etc/terminfo
```

Pour que le fichier soit utilisé par agetty, tapez :

```
agetty -c ttyUSB0 4800 minitel1b-80
```

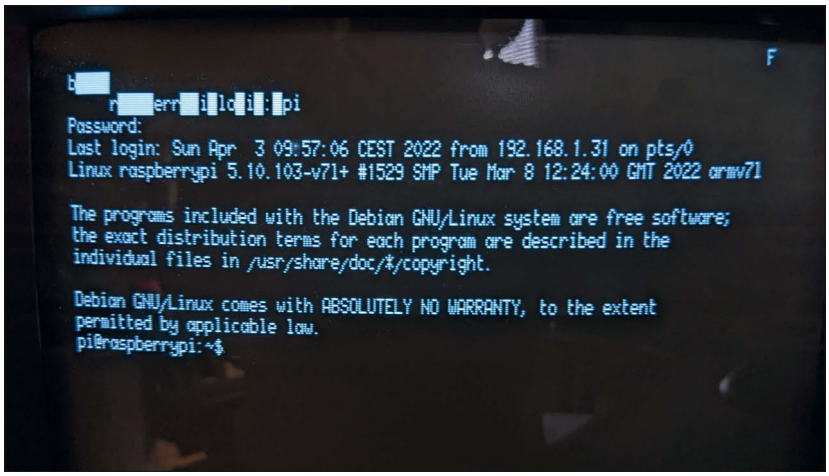
Créez le fichier de configuration

```
sudo nano /etc/systemd/system/serial-getty\@.service
```

Copiez/Coller le texte suivant dans le fichier de configuration (clic droit) :

Figure 6

Figure 7




```
# This file is part of systemd.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published
# by the Free Software Foundation; either version 2.1 of the License,
# or (at your option) any later version.

[Unit]
Description=Serial Getty on %I
Documentation=man:agetty(8) man:systemd-getty-generator(8)
Documentation=http://0pointer.de/blog/projects/serial-console.html
BindsTo=dev-%i.device
After=dev-%i.device systemd-user-sessions.service plymouth-quit-wait.service
After=rc-local.service
# If additional gettys are spawned during boot then we should make
# sure that this is synchronized before getty.target, even though
# getty.target didn't actually pull it in.
Before=getty.target
IgnoreOnIsolate=yes
[Service]
ExecStart=-/sbin/agetty -L -i -I "\033\143" 4800 %I minitel1b-80
Type=idle
Restart=always
UtmpIdentifier=%I
TTYPath=/dev/%I
TTYReset=yes
TTYVHangup=yes
KillMode=process
IgnoreSIGPIPE=no
SendSIGHUP=yes
[Install]
WantedBy=getty.target
```

Enregistrez le fichier (CTRL+x, y et Entrée).

Activez le service au démarrage, copiez les lignes en même temps dans votre client SSH :

```
sudo ln -s /etc/systemd/system/serial-getty@.service /etc/systemd/system/getty.target.wants/serial-getty@ttyUSB0.service
```

Lancez le service :

```
sudo systemctl daemon-reload
```

```
sudo systemctl start serial-getty@ttyUSB0.service
```

Ou redémarrez (pour vérifier qu'il se lance automatiquement) :

```
sudo reboot
```

Les 2 premières lignes sont un peu bizarres (avec des gros carrés blancs), mais après avoir tapé le nom d'utilisateur (pi) et validé, l'affichage devient normal et vous disposez d'un terminal fonctionnel. **Figure 7**

A vous de jouer !

Bibliothèque Minitel1B pour Arduino (2 versions)

Éric Sérandour
Bidouilleur, auteur de littérature expérimentale, enseignant dans le secondaire.
Site Web :
https://entropie.org

La programmation des microcontrôleurs ATmega avec Arduino m'a ramené quelques décennies en arrière, à l'époque où les micro-ordinateurs 8 bits faisaient rêver l'adolescent que j'étais. À la maison, il n'y avait des ordinateurs que pendant les vacances scolaires, mon père ramenant occasionnellement un ZX81 ou un TO7/70. La seule machine à clavier toujours présente était un Minitel.

Beaucoup de Français ont une histoire personnelle avec le Minitel. J'ai pu m'en rendre compte lorsque j'ai transformé un Minitel en « livre d'or numérique » lors d'un événement au musée d'art et d'histoire de Saint-Brieuc, puis lors d'une « Fête de la Science ». Le public était particulièrement enthousiaste et chacun y allait de ses souvenirs. Le Minitel a la particularité de relier les générations et d'inviter au dialogue : objet de curiosité pour les enfants, il amène les plus âgés à parler de l'arrivée des premiers outils numériques dans les foyers dans les années 1980.

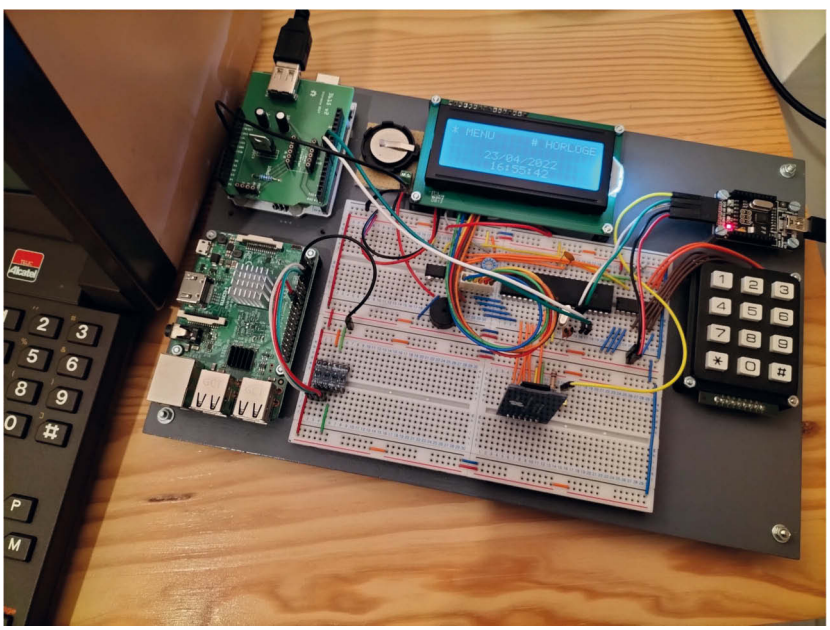
C'est en 2016 que j'ai initié l'écriture de la bibliothèque Minitel1B pour Arduino. Cette bibliothèque existe en deux versions : Minitel1B_Soft et Minitel1B_Hard. Toutes deux sont téléchargeables sur GitHub :
https://github.com/eserandour/Minitel1B_Soft
https://github.com/eserandour/Minitel1B_Hard

Minitel1B_Soft fait appel à SoftwareSerial.h qui permet d'ajouter un port série logiciel à un microcontrôleur du type ATmega328P, celui qui équipe la carte Arduino Uno. Dans ce cas, le port série matériel est conservé pour communiquer avec l'ordinateur et les broches indépendantes définies comme port série logiciel permettent d'échanger avec le Minitel.

Minitel1B_Hard est indépendante de SoftwareSerial.h et permet de faire communiquer le Minitel et un microcontrôleur par l'intermédiaire d'un de ses ports séries matériels. Personnellement j'utilise un ATmega1284P. J'ai été amené à développer cette nouvelle bibliothèque parce qu'assez vite la carte Arduino Uno a montré ses limites au niveau mémoire. Certains bugs apparaissaient dès que les programmes devenaient gourmands en données textuelles par exemple. Je me suis donc tourné vers l'ATmega1284P qui dispose de deux ports séries matériels, de 16 ko de RAM et de 128 ko de mémoire Flash (à comparer aux 2 ko de RAM et 32 ko de mémoire Flash de l'ATmega328P). Une autre alternative intéressante est d'utiliser la carte Arduino Mega 2560 (plusieurs ports séries matériels, 8 ko de RAM et 256 ko de mémoire Flash). Lomig Perrotin, le développeur du projet Zardos, utilise ma bibliothèque en basant son projet sur une carte Arduino Mega. De son côté, Iodeo développe un projet Minitel sur une base ESP32. Il a d'ailleurs contribué à quelques améliorations concernant la bibliothèque.

Comment relier un Minitel et une carte Arduino ?

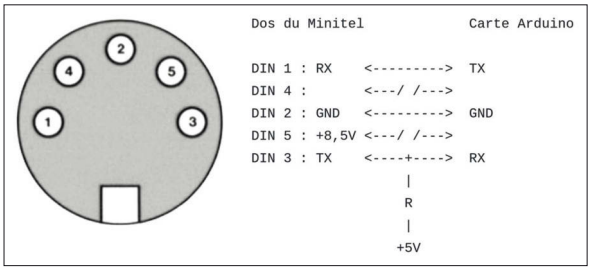
Dans la bibliothèque, vous trouverez un fichier README.txt qui indique les branchements à effectuer entre la prise DIN qui se trouve à l'arrière des Minitel 1B (prise péri-informatique 5 contacts à 45°) et la carte Arduino (voir schéma ci-dessous). Un élément très important, auquel je n'avais pas prêté attention au départ, concerne la sortie TX du Minitel : celle-ci est à collecteur ouvert, elle nécessite donc une résistance de tirage (pull-up). Personnellement, j'utilise une résistance de 10 kilohms. Sans cette résistance R, la communication ne pourra pas se faire correctement. Je récupère le +5V par l'intermédiaire de la broche DIN 5 (13 V à vide sur mon Minitel) sur laquelle je branche un régulateur 5V. Ceci permet d'alimenter la carte Arduino directement avec le Minitel. À noter que tous les Minitels ne permettent pas d'exploiter cette 5e broche (cette fonction n'est pas disponible sur les versions dont l'identification porte les références Cu2 à Cu4 incluses ; elle est par contre disponible pour les versions Cu5 / Cu; / Cu< / Bu0 et suivants).
Pour la fabrication du câble, je vous invite à aller voir sur mon site Web à cette adresse :
https://entropie.org/3615/index.php/hardware/.



Ma plate-forme de travail à base d'ATmega 1284P, connectée au Minitel.

Présentation de la bibliothèque Minitel1B

Pour écrire cette bibliothèque, j'ai utilisé un document de 193



pages édité par les PTT de l'époque : « Spécifications Techniques d'Utilisation du Minitel 1B », téléchargeable aujourd'hui à cette adresse : https://www.goto10.fr/minitel/specifications/stum1b.pdf

Ce document date de novembre 1986 et décrit le fonctionnement du Minitel. La lecture en a été pour moi passionnante. Elle m'a permis de rentrer dans l'intimité du Minitel et lorsque j'ai commencé à écrire la bibliothèque, j'ai tout de suite voulu coller au plus près de ce document. Mon code source renvoie d'ailleurs continuellement à ses pages.

Hello World !

Une étape importante qui fera adopter un outil plutôt qu'un autre est le fameux « Hello World ! ». C'est donc par son entremise que je vous invite à essayer la bibliothèque Minitel1B, dans une de ses deux versions. Au préalable, il faudra télécharger la bibliothèque au format ZIP depuis GitHub (Minitel1B_Soft-master.zip ou Minitel1B_Hard-master.zip) et la décompresser dans votre répertoire sketchbook/libraries/, sketchbook étant l'emplacement du carnet de croquis dans le logiciel Arduino. Le fichier HelloWorld.ino décrit ci-dessous fait partie des quelques exemples proposés avec la bibliothèque Minitel1B.

Avec Minitel1B_Soft :

On commence par importer deux bibliothèques :

```
#include <Minitel1B_Soft.h>
#include <SoftwareSerial.h>
```

Puis on définit un port série logiciel pour la connexion avec le Minitel. J'utilise les broches 8 et 9 de la carte Arduino Uno pour la simple raison que j'ai mis au point un shield Arduino (3615) qui utilise ces deux broches (voir mon site Web). On peut bien évidemment en changer :

```
Minitel minitel(8, 9);  // RX, TX
```

Avec Minitel1B_Hard :

On n'importe qu'une seule bibliothèque :

```
#include <Minitel1B_Hard.h>
```

Puis on choisit un port série matériel pour la connexion avec le Minitel. Sur l'ATmega 1284P avec lequel je travaille, je choisis le deuxième port série (Serial1 / RXD1 TXD1) :

```
Minitel minitel(Serial1);
```

Commun à Minitel1B_Soft et Minitel1B_Hard :

La connexion avec l'ordinateur se fait par le port série matériel Serial (RXD TXD pour l'ATmega 328P, RXD0 TXD0 pour l'ATmega 1284P). À la mise sous tension du Minitel, la vitesse des échanges entre le Minitel et le périphérique est de 1200 bauds par défaut. On envisage cependant le cas où le Minitel se trouve dans un autre état, ce qui peut arriver lorsque vous serez amené à bidouiller avec un Minitel à chaud.

```
void setup() {
  Serial.begin(9600);  // 9600 bauds
  minitel.changeSpeed(minitel.searchSpeed());
  minitel.newScreen();
}

void loop() {
  minitel.print("Hello World ! ");
}
```

Les fonctions disponibles

Les fonctions sont réparties en 11 catégories. Elles sont décrites dans la bibliothèque elle-même, dans le fichier d'entête (.h). Je vais juste les lister, ce qui vous permettra d'avoir un aperçu des possibilités :

Vitesse de la liaison série (3 fonctions) :
int changeSpeed(int bauds), int currentSpeed(), int searchSpeed()

Séparateurs (2 fonctions) :
void newScreen(), void newXY(int x, int y)

Modes du standard Télétel (6 fonctions) :
void textMode(), void graphicMode(), byte pageMode(), byte scrollMode(), byte modeMixte(), byte modeVideotex()

Standards (2 fonctions) :
byte standardTeleinformatique(), byte standardTeletel()

Contenu (12 fonctions) :
void attributs(byte attribut), void print(String chaine), void println(String chaine), void println(), void printChar(char caractere), void printDiacriticChar(unsigned char caractere), void printSpecialChar(byte b), byte getCharByte(char caractere), void graphic(byte b), void graphic(byte b, int x, int y), void repeat(int n), void bip()

Curseur (10 fonctions) :
void cursor() - void noCursor(), void moveCursorXY(int x, int y), void moveCursorLeft(int n), void moveCursorRight(int n), void moveCursorDown(int n), void moveCursorUp(int n), void moveCursorReturn(int n), int getCursorX(), int getCursorY()

Effacements, Suppressions, Insertions (13 fonctions) :
void cancel(), void clearScreenFromCursor(), void clearScreenToCursor(), void clearScreen(), void clearLineFromCursor(), void clearLineToCursor(), void clearLine(), void deleteChars(int n), void insertChars(int n), void startInsert(), void stopInsert(), void deleteLines(int n), void insertLines(int n)

Géométrie (3 fonctions) :
void rect(int x1, int y1, int x2, int y2), void hLine(int x1, int y, int x2, int position), void vLine(int x, int y1, int y2, int position, int sens)

Clavier (6 fonctions) :
unsigned long getKeyCode(bool ascii = true), byte smallMode(), byte capitalMode(), byte extendedKeyboard(), byte standardKeyboard(), byte echo(boolean commande)

Protocole (4 fonctions) :
byte aiguillage(boolean commande, byte emetteur, byte recepteur), byte statusAiguillage(byte module), byte connexion(boolean commande), byte reset()

Écrire un octet ou un mot / Lire un octet (3 fonctions) :
void writeByte(byte b), void writeWord(word w), byte readByte()

Parmi ces fonctions, l'une d'elle est vraiment particulière, c'est la fonction attributs dans la catégorie contenu. Elle permet plusieurs choses comme de fixer la couleur d'un caractère (CARACTERE_NOIR / CARACTERE_ROUGE / CARACTERE_VERT / CARACTERE_JAUNE CARACTERE_BLEU / CARACTERE_MAGENTA / CARACTERE_CYAN / CARACTERE_BLANC) ou sa couleur de fond (FOND_NOIR / FOND_ROUGE / FOND_VERT / FOND_JAUNE / FOND_BLEU / FOND_MAGENTA / FOND_CYAN / FOND_BLANC). Elle permet également de jouer sur la taille des caractères (GRANDEUR_NORMALE / DOUBLE_HAUTEUR / DOUBLE_LARGEUR / DOUBLE_GRANDEUR), de les faire clignoter ou pas (CLIGNOTEMENT / FIXE), d'inverser le fond ou pas (INVERSION_FOND / FOND_NORMAL), de les masquer (MASQUAGE / DEMASQUAGE). L'attribut de lignage (DEBUT_LIGNAGE / FIN_LIGNAGE) correspond à un soulignage en mode texte et à des caractères disjoints en mode semi-graphique. Pour utiliser la fonction attributs, il suffit de taper par exemple :

```
minitel.attributs(CARACTERE_NOIR);
minitel.attributs(FOND_JAUNE);
minitel.attributs(DOUBLE_HAUTEUR);
minitel.print("Hello World ! ");
```

Une autre fonction, qui sera beaucoup moins utilisée, fait également appel à des constantes particulières : il s'agit de printSpecialChar dans la catégorie contenue. Les possibilités sont : LIVRE / DOLLAR / DIESE / PARAGRAPHE / FLECHE_GAUCHE / FLECHE_HAUT / FLECHE_DROITE / FLECHE_BAS / DEGRE / PLUS_OU_MOINS / DIVISION / UN_QUART / UN_DEMI / TROIS_QUART / OE_MAJUSCULE / OE_MINUSCULE / BETA. Pour utiliser cette fonction, il suffit de taper par exemple :

```
minitel.printSpecialChar(FLECHE_HAUT) ;
```

Dans tous les cas, je vous invite à tester les quelques exemples proposés avec la bibliothèque, notamment Demo.ino pour ce qui concerne les possibilités d'affichage.

Afficher une image sur l'écran

Outre le texte, la bibliothèque permet également d'afficher des images. Pour cela la fonction clé est **writeByte** (écrire un octet). L'image, stockée sous forme d'instructions dans la mémoire Flash du microcontrôleur de la carte Arduino, nécessite en effet de lire une trame octet par octet et de l'aiguiller vers le Minitel. L'exemple Portrait.ino qui accompagne la bibliothèque donne un aperçu des possibilités en matière graphique. Obtenir ce type d'images à l'écran demande un peu de préparation. Je suis parti d'une image au format JPG, puis en utilisant le programme minitel11 qui fonctionne sous Processing 3 (https://drive.google.com/file/d/0BxdNVN-CFfF5aTdUUU1uREsxZ3M/view), j'ai pu créer un fichier Vidéotex correspondant à l'image en question, au format VDT. Reste ensuite à convertir ce fichier Vidéotex en suite d'octets pour le programme Arduino (https://github.com/eserandour/Conversion_Videotex_Hex). Le but de cet article n'étant pas l'affichage Vidéotex, je m'en tiendrai là, il s'agit juste d'ouvrir des perspectives.

Lorsque j'ai démarré ce projet de bibliothèque, j'avais également le désir de réaliser une sorte d'ordinateur 8 bits qui aurait fonctionné avec un Minitel. Je n'ai pas encore pris le temps de développer cette idée, cependant d'autres personnes se sont emparées de la bibliothèque Minitel1B_Hard et ont quelque part amorcé ce que je souhaitais faire. J'en suis très heureux. Il s'agit du projet ZARDOS de Lomig Perrotin (https://zardos.fr) et du projet Minitel ESP32 de Iodeo (https://github.com/iodeo/Minitel-ESP32). Merci à eux.

Quelques idées de projets

Un de mes premiers projets fut de réaliser une plate-forme dédiée à l'envoi de tweets depuis un Minitel (tutoriel sur mon site Web). J'ai choisi pour ce projet d'utiliser une carte Arduino Uno et un Raspberry Pi 2. Mais pourquoi donc utiliser une carte Arduino ? Je voulais que lorsqu'on allume le Minitel (qui alimente électriquement la carte Arduino), l'interface apparaisse tout de suite ; un microcontrôleur permet cette rapidité.

Une ESP32 sur Minitel ? C'est possible avec Minitel ESP32 Dev Board

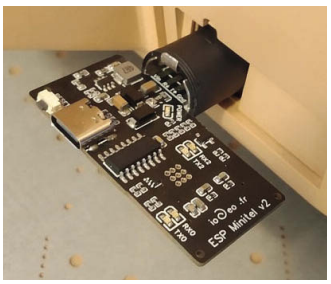
Louis Henrionnet

Ingénieur de formation et développeur / réparateur hardware indépendant, je cherche à apporter des solutions sur-mesure sur les thèmes du retrofit et du upcycling.

contact@iodeo.fr

Un Minitel connecté ? C'était un pléonasme jusqu'en 2012, année où le réseau x25 a été définitivement coupé. Mais aujourd'hui, c'est moins évident. Internet a sûrement sa part de responsabilité dans le déclin du minitel, mais il peut aussi aider à sa survie ! C'est l'idée de ce projet qui apporte une connexion WiFi au minitel grâce une carte d'extension auto-alimentée programmable basée sur un ESP32 qui se branche directement sur la prise péri-informatique !

La carte est disponible ici : https://www.tindie.com/stores/iodeo/



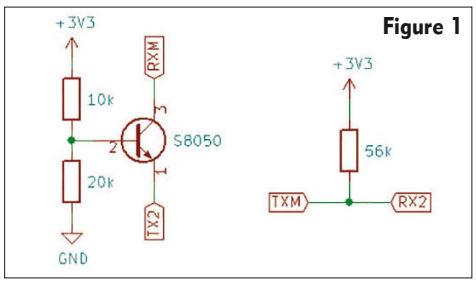


Figure 1

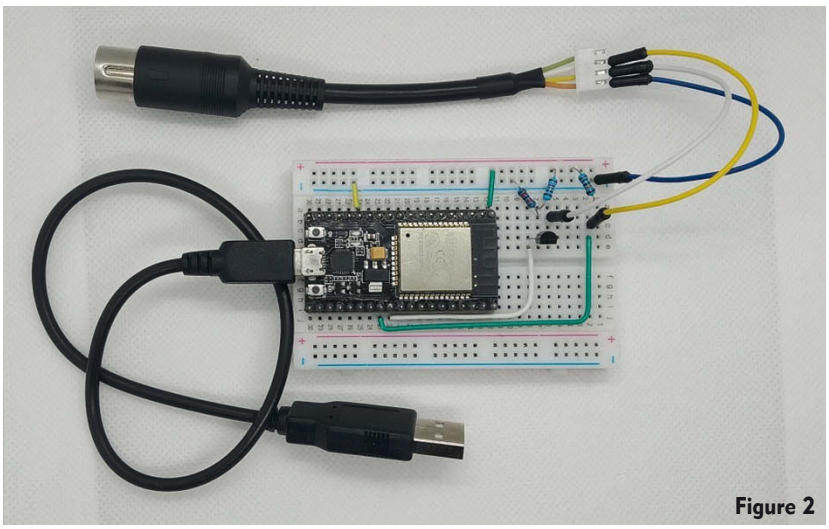


Figure 2

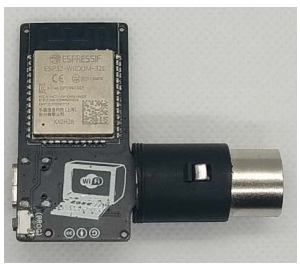


Figure 3

La prise péri-informatique

La prise péri-informatique est un connecteur DIN-5 femelle qui se trouve à l'arrière de la plupart des minitels et qui servait à ajouter des périphériques tel qu'une imprimante, un modem ou un lecteur de carte. Cette prise contient un port série TTL à collecteur ouvert que l'on peut adapter en UART avec quelques composants usuels. **Figure 1**

La **Figure1** donne un schéma de convertisseur pour un port UART à 3,3 volts. La broche Tx du périphérique TX2 est reliée à l'émetteur d'un transistor NPN dont la base est mise à 2v2 avec un diviseur de tension et le collecteur constitue la sortie vers le Rx du minitel RXM. Ainsi, lorsque TX2 est au niveau bas, le courant peut circuler de la base vers l'émetteur, le transistor est passant et RXM se retrouve au niveau bas. A l'inverse, lorsque TX2 est au niveau haut, le transistor est bloqué, RXM se retrouve en circuit ouvert, ce que le minitel interprète comme un niveau haut avec sa résistance de tirage interne. Dans l'autre sens, la broche Tx du minitel TXM est simplement relié au Rx du périphérique RX2 avec une résistance de tirage à 3,3V de 56k.

En pratique, ce port série permet d'interagir avec l'écran et le clavier du minitel avec le même protocole que par la ligne téléphonique, ce qui offre une voie idéale pour contrôler l'appareil avec un micro-contrôleur ou un ordinateur. De plus, le protocole est précisément documenté dans les Spécifications Techniques d'Utilisation du Minitel1B (STUM1B) qui sont disponibles en version html [1].

Le minitel et l'ESP32

On trouve toute une variété de projets de réutilisation du Minitel exploitant la prise péri-informatique, souvent avec des Arduino ou des Raspberry Pi. Certains projets visent à recréer des services d'époque comme l'annuaire électronique [2] et d'autres détournent l'usage pour en faire un terminal Linux [3], afficher des images jpeg [4] ou même faire le café [5] ! L'ESP32 est un compromis intéressant pour ce genre de projets car il allie la compacité et la rapidité de démarrage des micro-contrôleurs avec la connectivité des ordinateurs. C'est un module accessible, soutenu par une large communauté et que l'on peut programmer en C++ avec Arduino IDE aussi bien qu'en Micropython avec Thonny IDE par exemple.

Avec un câble DIN-5 de récupération, les quelques composants usuels de la **Figure1** et une carte de développement ESP32 générique, on peut déjà essayer de redonner vie au minitel du grenier pour une dizaine d'euros ! Pour savoir si votre minitel est compatible, il suffit de regarder si la prise péri-informatique et la touche Fnct du clavier sont présentes. C'est notamment le cas des modèles Minitel 1 bistandard et Minitel 2. **Figure 2**

Une carte sur-mesure auto-alimentée

Pour rendre ces projets plus portables et peut-être plus élégants, le minitel ne mérite-t-il pas une carte d'extension dédiée ? Prenez donc un ESP32. Ajoutez un port DIN-5 mâle pour le connecter au minitel. Incorporez un port USB-C pour le programmer facilement comme un Arduino. Assaisonnez avec un abaisseur de tension à découpage capable d'alimenter la carte indifféremment par le minitel ou le port USB. Dressez le tout sur un circuit imprimé. Et vous obtenez la carte minitel-ESP32 ! **Figure 3**

La carte, en vente sur Tindie [6] pour un prix allant de 30 à 65 € en fonction des options, est assemblée artisanalement en France. N'hésitez donc pas à faire une demande en cas de rupture de stock ! Le projet étant sous licence *CC-BY-SA 4,0*, vous pouvez la reproduire et l'améliorer à votre guise sous réserve de créditer l'auteur (BY) et de diffuser vos travaux sous la même licence (Share Alike) afin que tout le monde en profite. **Figure 4**

Quelques détails de conception

Un connecteur DIN-5 mâle est intégré à la carte de sorte qu'elle puisse se brancher directement à l'arrière du minitel. Ce connecteur étant introuvable en version à souder sur circuit imprimé, l'astuce a été d'utiliser un connecteur de câblage avec des pattes assez longues. Le circuit imprimé est inséré entre 4 pattes qui sont soudées sur de larges pads. La 5ème patte est alors légèrement pliée de manière à venir en contact avec le circuit pour être soudée à son tour. Le pcb comporte un autre emplacement pour un connecteur JST au pas de 2,54mm, expliquant la présence des oreilles sécables (**Figure 5**).

Du côté de l'alimentation, un abaisseur de tension à découpage s'est avéré nécessaire pour l'auto-alimentation à cause des niveaux de tensions mesurés : une tension de 13,5V était mesurée à vide, contre 5V maximum spécifié dans les STUM1B. L'ESP32 fonctionnant en 3,3V (soit potentiellement une dizaine de volts de dropout) avec un courant de 500mA, cela ferait 5W à dissiper avec un régulateur linéaire classique ! La conversion à découpage résout ce problème en restant compacte et a aussi le bon goût de fonctionner sur une grande plage de tension, englobant le 5V du port USB. Il a donc suffi d'une diode anti-retour pour obtenir une alimentation multi-source.

Il est aussi intéressant de remarquer le circuit d'autoreset de l'ESP32 (habituellement composé de transistors et de résistances) qui est réalisé avec un seul composant UMH3N dans un package SOT-363 de 1,8mm x 1,8mm, offrant un gain de place significatif. Ce circuit permet de programmer la carte sans avoir à manipuler les boutons Reset et GPIO0. **Figure 5**

Ressources et usages

Le dépôt GitHub [7] rassemble des ressources utiles pour développer en Arduino et en Micropython, à savoir des librairies et quelques exemples d'applications comme un jeu Pong, un client telnet pour faire une partie d'échecs en ligne sur Freechess.org, une liaison série bluetooth avec un RaspberryPi, un client de session ssh pour administrer une machine distante. Une vidéo montrant quelques exemples est disponible [8] et des instructions de démarrage sont données sur la page Hackaday du projet [9]. **Figure 6**

On peut mentionner aussi le « 3615 SSH » [10] développé par jbellue en C++ avec plateformIO. Il s'agit d'un client SSH doté d'un écran d'accueil et d'une interface utilisateur basé sur un ESP32 ! Pour le faire tourner sur la carte minitel-

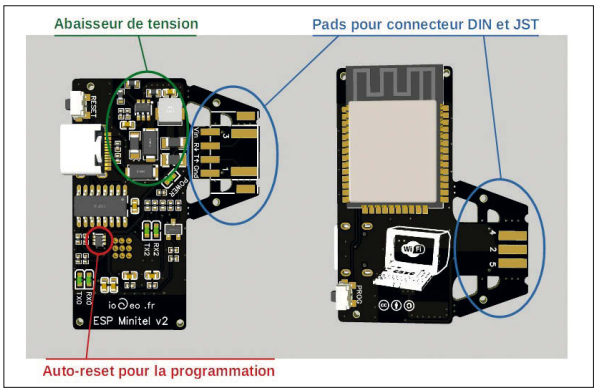


Figure 5

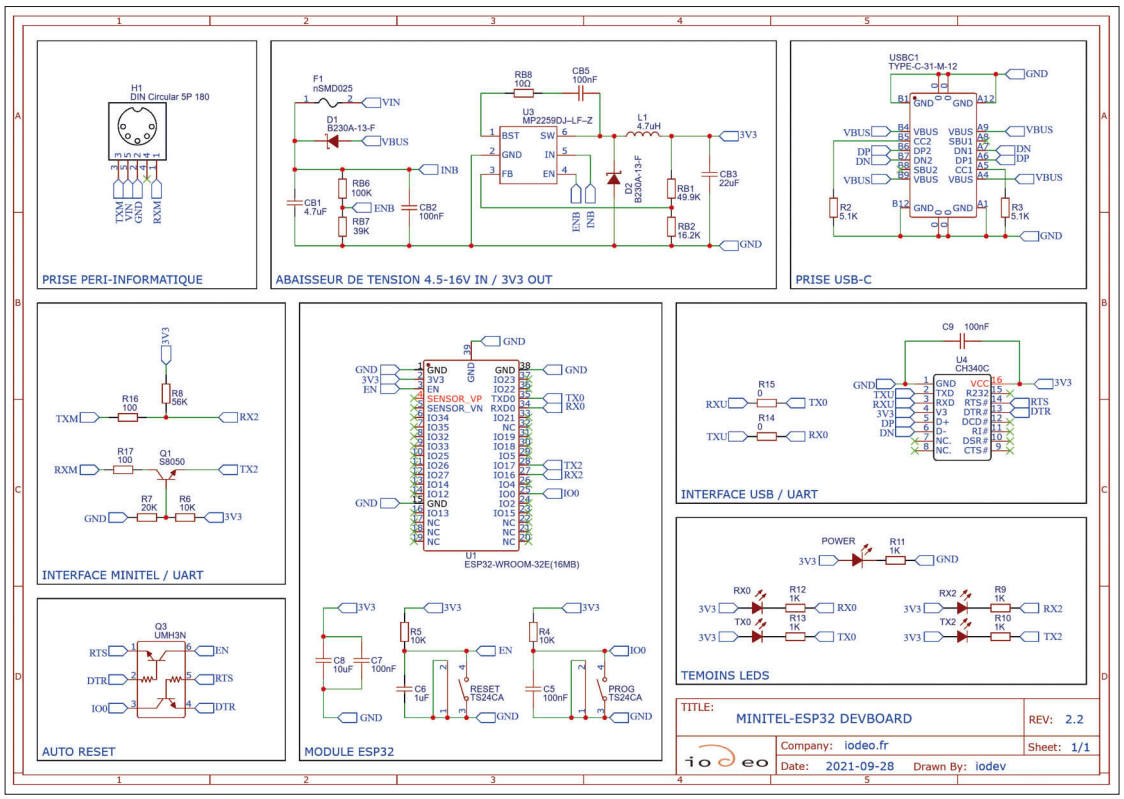


Figure 4

Figure 6

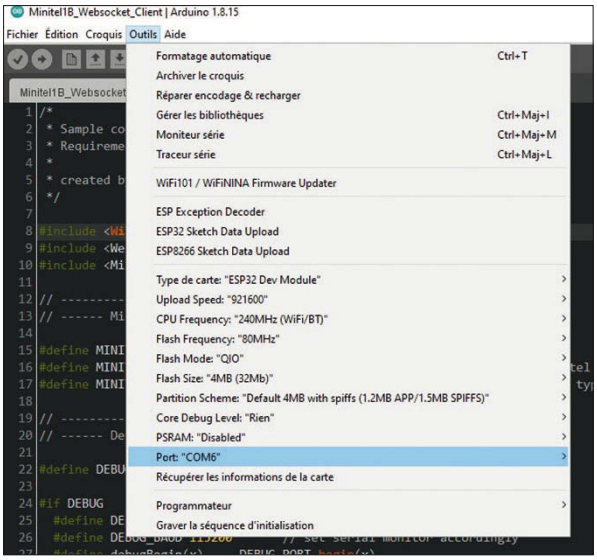


Figure 7

ESP32, il faut juste adapter le port série en remplaçant « minitel(Serial) » par « minitel(Serial2) ».

Enfin, pour ceux qui s'intéressent à la domotique open source, un usage pourrait bien justifier le retour du minitel dans le salon : un moniteur mqtt capable d'afficher les données des capteurs de la maison et de piloter les actionneurs par envoi de commandes à la box domotique. On pourrait aussi imaginer un simple lecteur de flux RSS pour afficher les actualités de vos magasines et blogs préférés. En fait, les possibilités n'ont de limite que l'imagination !

Un exemple d'utilisation en hommage

10 ans après la coupure du réseau, nous allons voir comment l'ESP32 peut être utilisé pour consulter un service minitel comme à l'époque, mais en wifi ! Parce que oui, si le réseau a bel et bien disparu, quelques serveurs videotex ont été ressuscités et sont maintenus gracieusement par quelques passionnés qu'il convient de remercier chaleureusement ! Ces services sont accessibles sur minitel par la prise téléphonique en VOIP ou sur ordinateur avec un simple navigateur web grâce à un canal websocket et un émulateur minitel (voir les services ouverts [11]). C'est cette 2e option qui va nous intéresser avec l'ESP32. Nous allons utiliser le wifi pour nous connecter à un serveur minitel, ouvrir un canal websocket, puis faire transiter les flux directement entre le websocket et le port série du minitel.

En micropython ?

Ce cas d'utilisation a fait l'objet d'un développement en micropython sous le nom socketel qui serait un peu lourd à présenter ici car il inclut une interface utilisateur et donc beaucoup de ligne de codes. Pour en savoir plus, les sources et références sont disponibles sur le dépôt dédié [12].

Ou avec Arduino IDE !

On va plutôt s'intéresser au développement d'une version plus minimaliste et peut-être plus stable en c++ avec Arduino IDE. La version utilisée est la 1.8.15 avec le core esp32 d'Espressif Systems v1.0.6 [13]. Le type de carte par défaut « ESP Dev Module » est sélectionné ainsi que le port série correspondant. Pour notre application, la partition par défaut de la flash convient très bien. **Figure 7**

Commençons par vérifier les communications série. D'après les STUM1B [1], on sait que le minitel communique à 1200 bauds par défaut et que les données sont transmises sur 7 bits avec un bit de parité pair et un stop bit, ce qui se traduit par la configuration SERIAL_7E1.

```
void setup() {
 Serial.begin(115200); // port debug
 Serial2.begin(1200, SERIAL_7E1); // port minitel
}

void loop() {
 // redirection debug -> minitel
 while (Serial.available() > 0) {
  Serial2.write(Serial.read());
 }
 // redirection minitel -> debug
 while (Serial2.available() > 0) {
  Serial.write(Serial2.read());
 }
}
```

Après téléversement et ouverture du moniteur série à 115200 bauds, ce programme va simplement rediriger les entrées du moniteur série de l'Arduino vers l'écran du minitel et les entrées clavier du minitel vers le moniteur série. On vient en fait de programmer un convertisseur UART USB minimaliste. On peut par exemple saisir « hello world » dans le moniteur, et le voir s'afficher sur l'écran du minitel et inversement. C'est un bon début, mais le protocole minitel ne prévoit pas que l'affichage de caractère alphanumérique. On peut aussi avoir des niveaux de gris, l'affichage du curseur, un mode semi-graphique pour afficher des images ou encore un mode ascii qui permet de passer en affichage à 80 colonnes (contre 40 en videotex), sans compter les touches spéciales, les différents modes du clavier, la gestion du modem et des aiguillages, etc. Bref, le protocole est très complet et on va avoir besoin d'une couche d'abstraction. Heureusement, les précédents projets ont donné lieu à des librairies dont c'est justement le but. Ici, on va utiliser la librairie Minitel1B_Hard [14] qui à l'avantage d'être complète et fidèle aux STUM1B, ce qui s'avère très pratique pour comprendre comment fonctionne le protocole. Cette librairie va gérer les écritures / lectures du port minitel en simplifiant les échanges protocolaires et nous permettre de gérer le minitel avec des fonctions plus

évocatrices que des suites de caractères hexadécimaux. Par exemple, demander au minitel de passser à 4800 bauds s'écrit simplement « minitel.changeSpeed(4800); ».
Pour ce qui est du réseau, nous allons utiliser la librairie WiFi.h fournie avec le core de l'ESP32 et l'excellente librairie WebSocketsClient de Links2004 [15].
Voilà le programme :

```
// libraries
#include <WiFi.h>
#include <WebSocketsClient.h>
#include <Minitel1B_Hard.h>

// minitel config
#define MINITEL_PORT Serial2
#define MINITEL_BAUD 4800
#define MINITEL_DISABLE_ECHO true

// wifi credentials
const char* ssid     = "mySsid";
const char* password = "myPassword";

// host parameters
char* host = "home.teletel.org";
int port = 9001;
char* path = "/";

WebSocketsClient webSocket;
Minitel minitel(MINITEL_PORT);

void setup() {

 // minitel set-up
 int baud = minitel.searchSpeed();
 if (baud != MINITEL_BAUD)
  baud = minitel.changeSpeed(MINITEL_BAUD);
 if (MINITEL_DISABLE_ECHO)
  minitel.echo(false);

 // WiFi connection
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED)
  delay(10);

 // Websocket connection and callback
 webSocket.begin(host, port, path);
 webSocket.onEvent(webSocketEvent);

}

void loop() {

 // Websocket -> Minitel
 webSocket.loop();

// Minitel -> Websocket
 uint32_t key = minitel.getKeyCode(false);
 if (key != 0) {
  // prepare data to send over websocket
  uint8_t payload[4];
  size_t len = 0;
  for (len = 0; key != 0 && len < 4; len++) {
   payload[3-len] = uint8_t(key);
   key = key >> 8;
  }
  webSocket.sendTXT(payload+4-len, len);
 }
}

void webSocketEvent(WStype_t type, uint8_t * payload, size_t len) {
 if (type == Wstype_TEXT) {
   if (len > 0) {
    for (size_t i = 0; i < len; i++)
     minitel.writeByte(payload[i]);
   }
 }
}
```

La fonction setup permet de préparer les ingrédients nécessaires à l'application à savoir : la communication avec le minitel, la connexion au wifi et l'ouverture du canal websocket. L'établissement de la communication avec le minitel est un peu plus évolué que dans l'exemple précédent avec la fonction searchSpeed dont le but est de deviner la vitesse dans le cas où elle aurait déjà été modifiée avant le démarrage du programme. La réception du websocket est gérée par le callback webSocketEvent qui transfert simplement la donnée du serveur au minitel avec la fonction writeByte de la librairie minitel1B_hard. Dans l'autre sens, les saisies du clavier sont récupérées avec getKeyCode et conditionnées pour éviter de tronquer les codes spéciaux, avant d'être envoyées sur le canal websocket avec sendTXT en sautant les éventuels zéros de tête de trame. Le code est disponible sur le dépôt du projet [7] dans une version commentée avec d'autres services à consulter.
Si vous avez bien renseigné vos identifiants wifi (ssid et password), vous devriez voir apparaître le service kiosque du 3614 Télétel sur votre minitel qui va vous permettre d'accéder à d'autres services avec la touche Guide. Alors il ne me reste plus qu'à vous souhaiter quelques bonnes heures de rétro-découverte ! **Figure 8**

Un autre exemple : MiniChess UI

Pour mieux appréhender les capacités graphiques du minitel, nous allons nous intéresser au développement d'une interface de jeu d'échecs. En effet, dans l'exemple précédent l'affi-

Figure 8

chage à l'écran est entièrement géré par le service distant et nous avons très peu eu à utiliser la librairie Minitel1B_Hard. Ici, il va falloir rentrer un peu plus dans le détail.

L'écran du minitel se compose de 24 lignes et 40 colonnes en mode videotex standard, dans lesquels on peut afficher des caractères semi-graphiques constitués de matrices de 2 x 3 cellules. Il y a aussi un mode 80 colonnes dans lequel on ne peut afficher que du texte, avis aux amateurs d'art ascii ! Le mode videotex nous permet de plus d'afficher 8 couleurs en nuances de gris que l'on peut assigner à chaque caractère et à son arrière-plan. On peut donc dire que l'écran du minitel à une résolution graphique de 2*40 par 3*24, soit 80x72 pixels avec 8 couleurs en nuances de gris. Ce qui fait une faible résolution, d'autant plus que les pixels ne sont pas indépendants puisque découpés en paquet de 2x3 où seules 2 couleurs sont possibles. Heureusement, nous avons un outil de prototypage graphique basé sur l'émulateur JavaScript miedit [16] accessible en ligne comme par exemple sur le site du PAMAL [17]. **Figure 9**

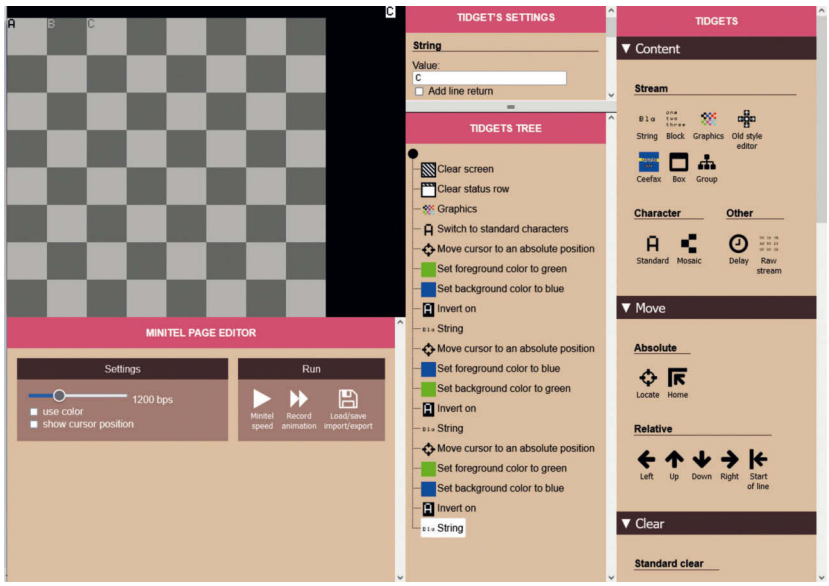


Figure 9

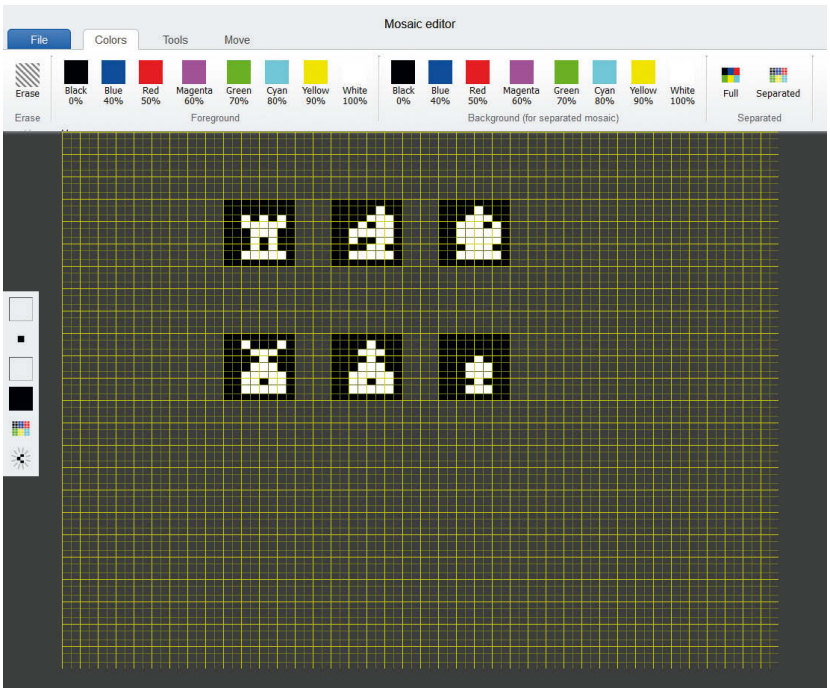


Figure 10

Un plateau d'échecs se constituant de 8*8 cases à représenter sur 24*40 caractères, on sait déjà qu'une case prendra au maximum 3 lignes et 5 colonnes. En pratique, en prenant une case de 3 lignes, on obtient un aspect carré avec 4 colonnes. Ce qui nous fixe la taille des cases à 3 lignes et 4 colonnes, dans lesquelles on va vouloir afficher le numéro de la case et la pièce. La 1ère colonne sera utilisée pour afficher le numéro de la case et les 3 suivantes pour la pièce, ce qui donne donc 9*6 pixels pour la pièce. La **Figure 10** montre le dessin des pièces réalisé avec miedit.

Maintenant qu'on a une idée graphique de l'application, voyons comment cela se traduit en programme avec les mêmes outils que dans l'exemple précédent à savoir Arduino IDE et la librairie Minitel1B_Hard [14]. Tout d'abord, il faut savoir que les fonctions de la librairie sont en fait très similaires à celles de miedit, puisque basées fidèlement sur le protocole du minitel. Ainsi, les fonctions utilisées pour le prototypage ont leur équivalent sous Arduino. Pour connaître le nom des fonctions et comment les utiliser, il suffit de se référer au fichier d'en-tête de la librairie et au programme « Demo » fourni par l'auteur (disponible pour l'ESP32 sur le dépôt du projet [7] sous le nom arduino/Minitel1B_Demo). Par exemple, sur la **Figure 9**, on voit que l'affichage des lettres se fait en 5 étapes dont voici les traductions :

- Move cursor to an absolute position : minitel.moveCursorXY(x,y);
- Set foreground color to green : minitel.attributs(FOND_VERT);
- Set background color to blue : minitel.attributs(CARACTERE_BLEU);
- Invert on : minitel.attributs(INVERSION_FOND);
- String str : minitel.print(str)

Pour information, l'utilisation de l'inversion des couleurs permet de contourner une contrainte de changement de couleur de fond qui impose de précéder le texte par un espace pour que le changement soit effectif. C'est à cause de cette contrainte que le caractère « A » sur la **Figure 9** est noir alors que les suivants sont en nuances de gris dénommées bleu et vert.

Ainsi, l'affichage du plateau se traduit par la méthode suivante :

```
void MiniChess::drawChessBoard() {
 // Fonction d'affichage du plateau de jeu seul

 // on se place en haut à gauche sur la case A8 qui est blanche
 minitel.moveCursorXY(BOARD_LEFT, BOARD_TOP);
 bool dark = false; //couleur de la case
 int cy = 8; // numero de case en ordonnées
 // on boucle sur les cases de haut en bas
 while (cy > 0) {
  int row = 1;
  // on boucle sur les 3 lignes constituant une case
  while (row <= CASE_HEIGHT) {
   int cx = 1; // numero de case en abscisses
   // on boucle sur les cases de gauche à droite
   while (cx < 9) {
    // on passe en mode texte pour afficher le nom de la case
    minitel.textMode();
    // on règle la couleur de caractère que l'on passe en arrière plan ensuite
```

```
    if (dark) minitel.attributs(CARACTERE_BLEU);
    else minitel.attributs(CARACTERE_VERT);
    minitel.attributs(INVERSION_FOND);
    if (row == 1) minitel.printChar(cx+64);   // A-H
    if (row == 2) minitel.printChar(cy+48);   // 1-8
    if (row == 3) minitel.printChar(SP);      // ESPACE
    // on passe en mode semi-graphique
    minitel.graphicMode();
    // on règle la couleur de la case
    if (dark) minitel.attributs(FOND_BLEU);
    else minitel.attributs(FOND_VERT);
    // on dessine un carcatère vide
    minitel.graphic(0b000000);
    // qu'on répète 2 fois
    minitel.repeat(CASE_WIDTH - 2);
    dark = !dark; // changement de couleur de case
    cx++; // case suivante
   }
   minitel.moveCursorLeft(CASE_WIDTH*8);
   minitel.moveCursorDown(1);
   row++; // ligne suivante
  }
 dark = !dark; // changement de couleur de case
 cy—; // rangée suivante
 }
}
```

Pour dessiner les pièces, nous allons utiliser les fonctions suivantes :

- minitel.graphicMode() pour passer en caractère semi-graphique
- minitel.graphic(byte b) pour afficher le caractère b

Le caractère b est écrit comme un nombre binaire à 6 bits dont chaque bit représente l'état d'une cellule dans le sens de la lecture. Par exemple, les trois premiers caractères de la tour s'écrivent 0b000010, le quatrième s'écrit 0b110101 et ainsi de suite. **Figure11**

On peut donc facilement définir une pièce par 9 octets décrivant chacun les 6 pixels de la cellule correspondante. Pour son affichage sur le plateau, il suffira d'ajuster la couleur de fond en fonction de la couleur de la case et la couleur de caractère en fonction de la couleur de la pièce.

```
#define COLOR_MASK 0b1000
#define PIECE_MASK 0b0111

  // Identification des pièces de jeux (PIECE+COLOR)
  enum { VOID, PAWN, ROOK, KNIGHT, BISHOP, QUEEN, KING}; // PIECES
  enum {_BLACK = 0b0000, _WHITE = 0b1000}; // COLORS

  // Pieces en 3*3 caractères semi-graphiques décrits dans le sens de la lecture
  byte piece[7][PIECE_WIDTH*PIECE_HEIGHT] = {
   {0b000000, 0b000000, 0b000000, 0b000000, 0b000000, 0b000000, 0b000000,
0b000000, 0b000000}, // VOID
   {0b000000, 0b000000, 0b000000, 0b000101, 0b101111, 0b000000, 0b000100,
0b101100, 0b000000}, // PAWN
   {0b000010, 0b000010, 0b000010, 0b110101, 0b111101, 0b100000, 0b011100,
0b011100, 0b001000}, // ROOK
   {0b000000, 0b000111, 0b000010, 0b011110, 0b011101, 0b101010, 0b001100,
0b111100, 0b001000}, // KNIGHT
   {0b000001, 0b001011, 0b000000, 0b111111, 0b101111, 0b101010, 0b011100,
0b111100, 0b001000}, // BISHOP
   {0b001001, 0b000011, 0b001000, 0b000111, 0b101111, 0b000010, 0b111100,
0b011100, 0b101000}, // QUEEN
   {0b000001, 0b001011, 0b000000, 0b000111, 0b101111, 0b000010, 0b111100,
0b011100, 0b101000} // KING
  };

[...]

void MiniChess::drawPiece(int cx, int cy, byte pc) {
 // Fonction pour afficher une pièce dans une case donnée
 // avec pc: l'identifiant de la pièce et de la couleur

 // on détermine les coordonnées du curseur du minitel
 int x = cx * CASE_WIDTH + BOARD_LEFT;
 int y = cy * CASE_HEIGHT + BOARD_TOP;

 // on récupère la couleur et la pièce
 byte color = pc & COLOR_MASK;
 byte p = pc & PIECE_MASK;

 // on passe en mode graphique et on règle les couleurs
 minitel.graphicMode();
 if (color == _WHITE) {
  // effet lignage pour améliorer la netteté
  minitel.attributs(DEBUT_LIGNAGE);
  minitel.attributs(CARACTERE_BLANC);
 }
 else { // _BLACK
  minitel.attributs(CARACTERE_NOIR);
 }
 // on détermine la couleur de la case
 if ((cx+cy)%2 == 1) minitel.attributs(FOND_BLEU);
 else minitel.attributs(FOND_VERT);
 // on dessine la pièce caractère par caractère
 for (int j = 0; j < PIECE_HEIGHT; j++) {
  minitel.moveCursorXY(x+1,y+j);
  for (int i = 0; i < PIECE_WIDTH; i++) {
   minitel.graphic(piece[p][i+j*PIECE_WIDTH]);
  }
 }
 if (color == _WHITE) {
  minitel.attributs(FIN_LIGNAGE);
 }
}
```

Avec cette méthode, on va pouvoir dessiner le plateau pièce par pièce. Il reste à indiquer au joueur la position du curseur de sélection sur le plateau afin qu'il puisse se repérer et définir son prochain coup. Nous allons pour cela utiliser le caractère restant en bas à gauche de la case, en la remplissant d'une couleur donnée : lorsque les noirs jouent, le marqueur de position est noir et le marqueur de sélection est blanc, et inversement lorsque c'est le tour des blancs.

```
void MiniChess::markCase(int cx, int cy, bool mark, bool force_erase) {
 // Fonction pour marqué la case courante dans le coin bas-gauche
 // Si mark = true, la marque de position ou de selection est affichée
 // Si mark = false, la marque est effacée sauf s'il s'agit de la selection
```

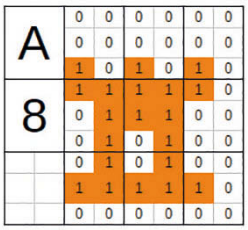


Figure 11

```
// auquel cas il faut mettre force_erase = true pour l'effacer

// on place le curseur du minitel au bon endroit
int x = cx*CASE_WIDTH + BOARD_LEFT;
int y = cy*CASE_HEIGHT + BOARD_TOP + 2;
minitel.moveCursorXY(x,y);
// on se met en mode graphique
minitel.graphicMode();
// on détermine si on est sur une case sombre ou claire
bool dark = false;
if ((cx+cy)%2 == 1) dark = true;
// on ajuste la couleur de fond
if (dark) minitel.attributs(FOND_BLEU);
else minitel.attributs(FOND_VERT);
// on détermine si on est sur la case sélectionnée
bool selected = (cx == scx && cy == scy);
if (mark) {
 // on détermine la couleur à mettre
 byte color = CARACTERE_BLANC;
 if (selected && (player == _WHITE))
  color = CARACTERE_NOIR;
 if (!selected && (player == _BLACK))
  color = CARACTERE_NOIR;
 // on affiche la marque
 minitel.attributs(color);
 minitel.graphic(0b111111);
}
else { // mark = false
 if (!selected || force_erase)
  // on efface la marque
  minitel.graphic(0b000000);
}
}
```

Les méthodes de déplacement du curseur de position sont alors simplement de la forme :

Code complet sur programmez.com & github

Pour acquérir les saisies du joueur au clavier, on utilise la méthode getKeyboardInput qui va filtrer et rediriger les touches clavier vers les commandes de jeu. Par exemple, la sélection d'une case va pouvoir se faire avec la touche ENVOI, SP (espace) ou CR (entrée). Les touches de déplacement ne sont pas filtrées mais retournées telles quelles par défaut. Cette méthode est elle-même utilisée par getMove qui se charge de traduire cela en déplacement et en sélection sur le plateau.

```
unsigned long MiniChess::getKeyboardInput() {
 // Fonction pour rediriger les saisies du clavier

 unsigned long key = minitel.getKeyCode();
 switch (key) {

  // Annulation de la sélection en cours
  case CORRECTION:
  case ANNULATION:
  case RETOUR:
  case ESC:
  case DEL:
                return CAN;  break;

  // Sélection
  case ENVOI:
  case SP:
                return CR;   break;

  // Touches inhibées
  case CONNEXION_FIN:
  case SOMMAIRE:
  case REPETITION:
  case GUIDE:
  case SUITE:
                return 0;    break;

  // Autres touches
  default:
                return key;  break;

 }
}

void MiniChess::getMove() {
 // Fonction pour acquérir le prochain coup
 bool gotMove = false;

 while (!gotMove) {
  // Saisie clavier
  unsigned long c = getKeyboardInput();
  if (c != 0) {
   switch (c) {
    // Mouvement du curseur
    case TOUCHE_FLECHE_HAUT:  moveUp();   break;
    case TOUCHE_FLECHE_BAS:   moveDown();  break;
    case TOUCHE_FLECHE_GAUCHE: moveLeft();  break;
    case TOUCHE_FLECHE_DROITE: moveRight(); break;
    // Selection
    case CR:
     // Case de départ
     if (scx == -1) {
      if (validateSelection()) {
       markCase(cx, cy, true); // marqueur de selection
      }
     }
     // Case d'arrivée
     else {
      if (validateMove()) {
       gotMove = true;
      }
     }
     break;
    // Annulation
    case CAN:
     // S'il y a une selection en cours
     if (scx != -1) {
      // on efface le marqueur de selection
      markCase(scx, scy, false, true);
      // on affiche le marqueur de position si besoin
      if (scx == cx && scy == cy)
```

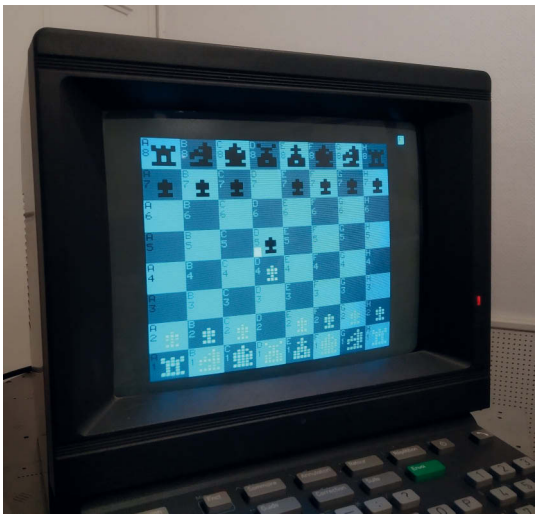


Figure 12

Figure 13

```
        markCase(cx, cy, true);
       scx = -1; scy = -1;
      }
      break;
    }
  }
 }
}
```

Une fois que le joueur a défini son coup avec getMove, la méthode makeMove se charge de l'afficher sur le plateau :
Code complet sur programmez.com & github
Et pour finir, le programme principal se réduit à :

```
#include "Minichess.h"
#include <Minitel1B_Hard.h>

#define MINITEL_PORT Serial2 // For ESP32 dev board
//#define MINITEL_PORT Serial1 // For ATMEGA32u4
#define BAUDS 4800

MiniChess chess(MINITEL_PORT);

void setup() {
 // on attend que le minitel démarre
 delay(500);
 // réglage port série et config minitel
 chess.initializeMinitel(BAUDS);

 // initialisation du jeu
 chess.drawChessBoard();    // affichage plateau vide
 chess.initPiecesPosition(); // placement des pieces
 chess.drawAllPieces();     // affichage des pièces
}

void loop() {
 while (true) {
  chess.getMove();
  chess.makeMove();
  chess.changePlayer();
 }
}
```

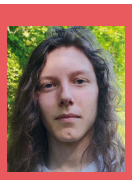
Eh bien, cela fait déjà un bon aperçu du programme. Le code source est disponible sur le dépôt du projet [7] sous le nom arduino/Minitel1B_MiniChess_UI. L'interface obtenue est jouable pour 2 joueurs mais elle n'intègre pas les règles du jeu; aux joueurs de les respecter ! Une première voie de développement serait d'implémenter les règles du jeu et une seconde, propre à l'ESP32, consisterait à utiliser le WiFi pour se connecter à un serveur d'échecs afin de jouer contre d'autres joueurs en ligne ou contre une IA. Toute contribution sera la bienvenue ! **Figure 12**

Avant de vous laisser vous amuser

Sur le même principe que la carte Minitel-ESP32, une cartouche basée sur un ATMEGA32u4 est en cours de développement. L'ATMEGA32u4 est le micro-contrôleur utilisé sur les Arduino Leonardo et Micro. La cartouche est donc compatible Arduino IDE et supporte l'USB nativement. Avec l'absence de WiFi et des ressources plus limitées que l'ESP32, la carte est plutôt dédiée au développement d'applications locales telles que les jeux Pong et MiniChess. Le gros intérêt est d'avoir un form factor intégrable dans un boîtier cylindrique de la taille d'un connecteur DIN-5. Une sorte de cartouche de jeux pour Minitel ! **Figure 13**

Références :

[1] https://jbellue.github.io/stum1b
[2] https://cq94.medium.com/fb59a843e86c
[3] https://pila.fr/wordpress/?p=361
[4] https://ressources.labomedia.org/processing_vs_arduino_vs_minitel
[5] https://arduiblog.com/2019/05/06/cafetiere-minitel/
[6] https://www.tindie.com/products/25418/
[7] https://github.com/iodeo/Minitel-ESP32
[8] https://youtu.be/iOB85X8F1vI
[9] https://hackaday.io/project/180473
[10] https://github.com/jbellue/3615_SSH
[11] https://www.museeminitel.fr/
[12] https://github.com/iodeo/Socketel
[13] https://github.com/espressif/arduino-esp32
[14] https://github.com/eserandour/Minitel1B_Hard.git
[15] https://github.com/Links2004/arduinoWebSockets.git
[16] https://github.com/Zigazou/miedit
[17] https://pamal.org/miedit/



Pascal Engélibert

Développeur libriste militant, bricoleur et étudiant en mathématiques, ça me fend le cœur de voir une machine au rebut, fonctionnelle ou non.

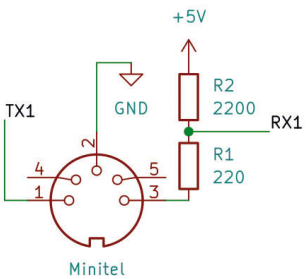
Le Minitel comme terminal GNU/Linux

À la rentrée de Jeunes-Science Bordeaux, le club de loisirs scientifiques et techniques associatif, j'ai vu le Minitel dans la pile d'antiquités, entre un Apple II, un Commodore et un disque dur de la taille d'une roue de voiture. Je me suis alors fixé l'objectif d'en faire un terminal pour un ordinateur moderne.

L'ère de la péri-informatique

Le Minitel dispose d'une prise "péri-informatique", qui est un port série d'entrée-sortie. Il suffit pour communiquer avec d'une interface série. J'ai donc utilisé une Arduino comme pont entre le Minitel et l'ordinateur. Mais attention, il faut un Arduino possédant plusieurs interfaces série (comme le modèle Mega). Cela nous évite d'avoir à coder un pilote série logiciel. Le standard péri-informatique utilise à peu près de l'ASCII 7 bits et un découpage en octets. Le bit de poids fort est la somme de contrôle des 7 autres.

Électronique



- Connectez le GND du Minitel (pin 2, celui du milieu) au GND de l'Arduino ;
- Connectez le RX du Minitel (pin 1, tout à gauche du GND) au TX1 de l'Arduino ;
- Connectez le RX1 de l'Arduino au TX du Minitel (pin 3, tout à droite du GND) à travers une résistance de 220Ω, et au +5V à travers une résistance pull-up de 2200Ω (toute valeur entre 1kΩ et 20kΩ devrait marcher). La résistance est nécessaire car la sortie du Minitel est en +8,5V.

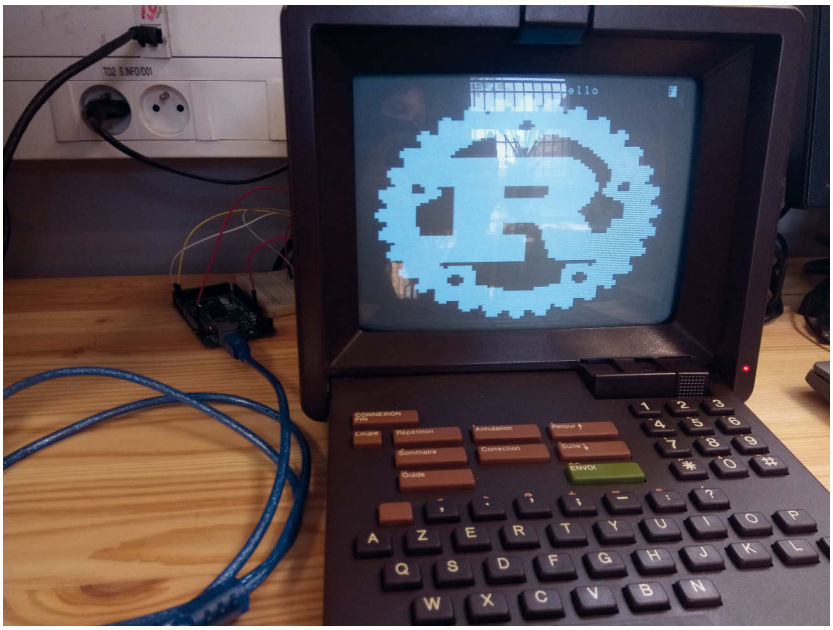
Logiciel

J'ai d'abord codé un contrôleur en Python. C'est une sorte d'émulateur de terminal, avec le port USB comme entrée-sortie. Il prend une ligne de commande et l'exécute à l'appui de la touche Envoi.

Rien que ça demandait de surmonter plusieurs difficultés :

- Il n'y a pas de touche Nouvelle Ligne. Seulement une touche Retour Chariot. Mais par miracle le caractère Nouvelle Ligne existe, on peut donc utiliser du CRLF. Ça veut dire qu'il faut interpréter la touche Envoi (qui a un code sur deux octets) comme LF, et transformer les LF venant de Linux en CRLF.
- Les accents ne sont pas standard. Une lettre accentuée fait trois octets : un qui indique un caractère spécial, un qui spécifie l'accent, et la lettre. Et attention ! On a droit aux majuscules accentuées (coucou Windows, le Minitel fait une chose mieux que toi), mais seulement les accents franchouillards. La piñata et le glacier würmien n'ont pas la nationalité.
- Le clavier est en majuscule, et les minuscules demandent l'appui de Maj. Si on veut inverser ce comportement, on doit pour chaque lettre arrivant au contrôleur, renvoyer un déplacement vers la gauche puis la lettre en casse inversée. On voit le clignotement à l'écran et ça oblige à taper assez lentement pour que la réécriture ait le temps de s'opérer.
- Le port péri-informatique supporte 1200 bit/s, soit un rafraîchissement complet de l'écran en plus de 6 secondes. Certains modèles sont plus rapides, jusqu'à 9600 bit/s. Cette lenteur oblige à optimiser l'affichage pour plus d'immédiateté. On ne peut pas rafraîchir tout l'écran en permanence, il faut envoyer uniquement les caractères qui changent. Donc s'il y a un bug (un imprévu dans le programme, un problème de signal, un utilisateur qui tape trop vite), l'écran ne sera pas dans l'état attendu, et le programme, qui n'a aucun moyen de le savoir, continuera même s'il écrit n'importe quoi.
- En mode semi-graphique, les caractères sont remplacés par une grille de 2×3 pixels. On peut alors dessiner, en convertissant les images correctement. On a même quelques nuances de gris ! L'inconvénient, c'est que le changement de couleur agit sur le caractère entier, soit 6 pixels, et non sur chaque pixel individuellement. La gestion de la couleur est donc un peu délicate.
- Si l'affichage est modifié en permanence, par exemple dans un jeu de Snake pour déplacer le serpent, l'appui de n'importe quelle touche peut interférer avec un caractère spécial de plusieurs octets envoyé par le contrôleur. Il en résulte un comportement imprévisible, et souvent un caractère aléatoire qui s'affiche. C'est pourquoi dans le Snake et dans le Tetris que j'ai faits, l'écran se retrouve vite constellé de caractères. J'ai essayé de nettoyer des petites parties différentes de l'écran en permanence, par exemple ligne par ligne, mais cela prend trop de temps, réduit la réactivité du jeu et n'est pas très efficace.
- L'écran est très petit, 40×24 caractères seulement. Aussi, il ne défile pas mais reboucle en haut quand le curseur arrive en bas. Il n'est donc pas vraiment adapté à un terminal de type Unix. Il faut soit créer un système de pagination, qui nettoie l'écran à chaque page, soit accepter de reboucler et nettoyer la fin de la ligne.
- Certains caractères manquent, comme la barre verticale `|`. Cette dernière étant indispensable dans un terminal, j'ai décidé de la substituer aux accents circonflexes `^` qui sont bizarrement affichés ` ` sur le Minitel.
- Côté Linux ce n'est pas facile non plus… Il faut pouvoir indiquer aux programmes la taille du terminal, qu'ils comprennent qu'ils sont dans un terminal et non dans un script, pour que par exemple bash écrive son invite de commande (prompt), gérer le retour arrière (qui n'est pas géré directement par bash), etc.

Toutes ces contraintes rendent la conception d'un terminal générique compliquée. Il faut que chaque programme adapte



sa gestion du clavier et de l'affichage, car les besoins sont différents pour un utilitaire en ligne de commande, un éditeur de texte, une messagerie instantanée ou un jeu en temps réel.

On m'a donc donné l'idée d'une approche différente : plutôt que d'envoyer la sortie des programmes au Minitel, on garde un tampon de l'écran du Minitel en mémoire. C'est un tableau de l'état de chaque caractère, dont le mode texte ou semi-graphique, la couleur, le clignotement, le code du caractère, etc. Les programmes écrivent dedans, puis demandent au contrôleur de l'envoyer. Au prix de calculs plus lents (ce qui reste négligeable), l'algorithme cherche alors à envoyer le moins de données possibles. Les caractères inchangés ne sont pas renvoyés, on profite de l'instruction plaçant le curseur à la position donnée et de celle permettant de répéter les N derniers caractères.

J'ai implémenté ce nouveau contrôleur en Rust (un langage bas-niveau offrant des abstractions haut-niveau sans coût et la sécurité de la mémoire), ce qui permet un code plus propre, sûr et rapide qu'en Python. C'est une bibliothèque réutilisable (publiée sous licence libre GNU AGPL), que j'ai pu utiliser pour faire un terminal, un Tetris et un Snake.

Au prix de quelques nanosieverts(1) et d'un ultrason désagréable, cette petite aventure rétro m'a plongé dans une époque que je n'ai pas connue, quoique j'y vis peut-être encore(2).

Dépôt Git du code du projet :
https://git.jeunes-science.asso.fr/tuxmain/Minitel

(1) Un tube cathodique émet une faible quantité de rayons X, négligeable en conditions normales d'utilisation par rapport à la radioactivité naturelle.

(2) Voir la conférence Internet libre ou Minitel 2.0 de Benjamin Bayart, aux Rencontres Mondiales du Logiciel Libre 2007, disponible sur PeerTube.

ZARDOS ou “Ze ARDuino Operating System”

ZARDOS est un modeste projet de système d’exploitation dédié aux microcontrôleurs compatibles Arduino et utilisant un terminal Minitel comme interface homme-machine.

Lomig PERROTIN

Né en 1982, comme le Minitel, et formé aux métiers de l’environnement, une certaine passion pour les chambres noires et les hasards de la vie m’ont amené à créer, il y a dix ans, ma propre société de fabrication de pellicules photographiques. A priori rien à voir avec l’informatique, mais en voulant automatiser mes propres outils de production, je me suis intéressé à la programmation des microcontrôleurs et une fois que l’on a mit le pied dedans...

Le but de ce projet est de créer un système d’exploitation, primitif, mais fonctionnel, qui puisse évoluer au fil du temps et être éventuellement enrichi par des contributeurs extérieurs.
Le principe de base est simple : l’Arduino communique avec le Minitel via la liaison série du câble péri-informatique. Dans un sens il récupère les informations en provenance du clavier, et dans l’autre il envoie ses commandes de configuration et d’affichage à l’écran.

Mais pourquoi diable essayer de créer un système d’exploitation pour un Minitel ?! Surtout quant à la base on est pas du tout informaticien et qu’on soude comme une patate.

Comme la plupart des gens nés dans les années 80, le Minitel fait partie du paysage de mon enfance, et j’en ai des souvenirs très présents pour y avoir cherché des adresses, consulté des résultats d’examens et… c’est à peu près tout ! Objectivement c’était un objet à la fois fascinant et terriblement frustrant. Quand nous avons reçu le nôtre c’était « Cosmos 1999 » qui débarquait à la maison ! Mais le prix des communications et la facture détaillée ont vite douché tout espoir d’une utilisation extravagante, j’en étais quitte pour m’amuser avec le clavier en imaginant que je faisais des jeux vidéo.

Arduino + Minitel

Bien des années après, je me suis intéressé à l’Arduino. Cela m’a amené à me renseigner sur l’histoire de l’informatique, les ordinateurs 8bits et les OS et quelque part sur Internet j’ai vu qu’on pouvait utiliser un Minitel comme terminal pour un Arduino. L’idée m’intéressait et au fil de mes recherches je suis tombé sur un montage et une bibliothèque Arduino/Minitel d’Eric Serandour (https://entropie.org/3615/). C’était exactement ce qu’il me fallait. Côté montage c’est très simple : les broches Rx & Tx de la prise péri-informatique du Minitel branchées en direct sur l’Arduino Uno avec juste une résistance de rappel. Côté bibliothèque, un modèle de clarté, facile à comprendre et à éditer. Je tiens d’ailleurs à le remercier ici, car sans son travail sur lequel je me suis entièrement appuyé je n’aurai absolument rien pu faire.

Le Minitel est en soi une grosse boîte vide, un simple terminal qui affiche et transmet les données sans véritablement les traiter : ce n’est pas un ordinateur. Mais avec ce montage, sans avoir besoin de modifier le Minitel, on intercale une unité logique, l’Arduino, pour obtenir quelque chose qui commence à se rapprocher d’un… ordinateur.

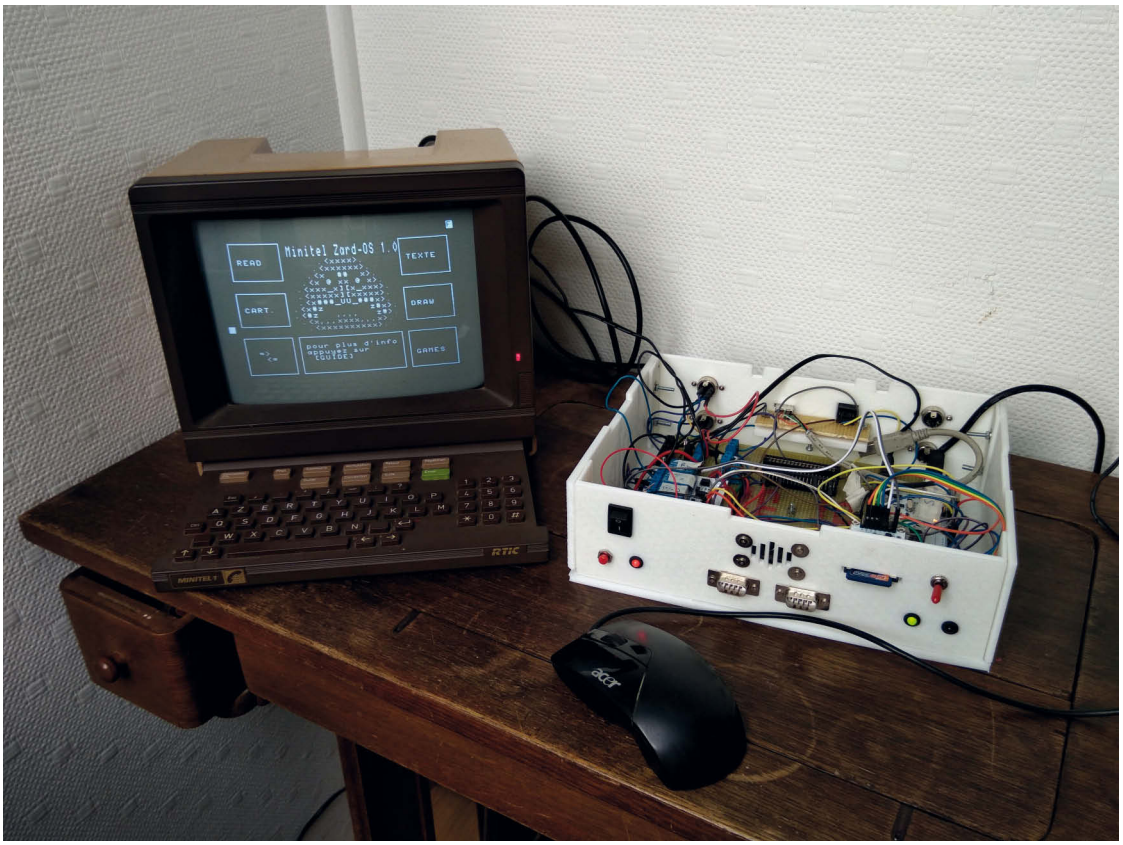
J’ai opté pour une interface graphique, plus intuitive à mon goût, et offrant plus de possibilités d’évolution. Pour cela j’ai détourné un code servant à utiliser une souris PS/2 comme capteur Arduino afin de modifier la localisation du curseur sur l’écran du Minitel.

Ce code n’avait bien sûr pas été prévu pour le Minitel, mais il permettait à l’Arduino de lire les mouvements relatifs de la souris sur les axes x & y ainsi que l’état des boutons. Je l’ai donc utilisé de la manière suivante : si le mouvement est supérieur ou inférieur à 0, alors la position correspondante du curseur sur l’écran du Minitel est augmentée ou diminuée de 1. Cela n’est pas très fluide, mais cela fonctionne quand même et permet de promener la souris sur l’écran ainsi que d’utiliser les clics droit & gauche pour des actions prédéfinies dans les sous-programmes.
La bibliothèque permettant d’afficher des cadres à l’écran et donc de les définir par des plages de coordonnées pouvant être définies comme des zones cliquables, ce qui me donnait donc un système d’icônes simples.

J’ai ainsi pu expérimenter et doter mon « OS » de quelques sous-programmes très basiques, pour dessiner avec la souris sur l’écran, écrire un texte et l’imprimer (mais sans sauvegarde pour le moment), ainsi qu’un petit jeu, « Marauder », dans lequel un unique monstre (caractère « M » !) descend de plus en plus vite vers un chevalier (caractère « 0 » !) qui cherche à lui échapper en allant à droite ou à gauche et en donnant des coups de sabre (caractère « / » !). On est loin de Space Invader, mais c’était surtout pour tester ce qu’il était possible de faire ainsi que mes propres compétences de codages qui sont, avouons-le, très limitées. Mais sur le principe ça marche et ça donne envie d’aller plus loin. J’ai promis à mon fils que je lui ferai un jeu vidéo rien que pour lui avec un petit aventurier qui explore un donjon.

Et pour quelques soudures de plus…

Lorsque j’ai voulu inclure des programmes plus volumineux dans mon système je me suis heurté à un double problème. Tout d’abord la taille du programme dépassait alors les possibilités de stockage de l’Arduino, mais surtout je ne parvenais pas à les inclure tel quel dans mon code. C’était notamment le cas avec TinyBasic+, un interpréteur BASIC qui avait été porté sur Arduino en utilisant un clavier PS/2 et un écran. J’avais réussi à le modifier pour utiliser l’écran et le clavier du Minitel, mais impossible de l’inclure au sein de ZARDOS.



Je me suis alors inspiré des vieux ordinateurs 8 bits utilisant des cartouches. Le principe étant que lorsque la cartouche est insérée dans son port, et activée via une icône sur l'écran, elle prend le relais de l'Arduino et gère les flux Tx & Rx du Minitel de manière indépendante. Ladite cartouche contenant elle-même un autre Arduino chargé d'un programme précis et interagissant avec le Minitel. Ne sachant pas comment procéder de manière électronique pour aiguiller les flux Tx/Rx j'ai choisi une solution « physique » : lorsque la cartouche est lancée à l'écran, l'Arduino active trois relais qui vont alimenter la cartouche en électricité et dérouter les flux Tx/Rx du Minitel vers celle-ci. Cela marche plutôt bien et il est possible d'améliorer ce système en le rendant plus flexible.
En établissant une liaison série entre la cartouche et l'Arduino qui gère ZARDOS, on pourrait alors, par exemple, connecter le système à Internet grâce à une cartouche équipée d'un shield Ethernet.

Au niveau du code justement, comment ça se passe ?

Le code Arduino de base est assez limité. Il contient principalement le contenu de l'écran de démarrage, du menu, la boucle qui vérifie l'état de la liaison série avec le Minitel, celui de la souris et du clavier, ainsi que le codage du clavier, et de certaines configurations spécifiques clavier/souris en fonction des sous-programmes utilisés.

Tout le reste du système est dans la bibliothèque qui contient les fonctions d'affichage codées par Eric Serandour. Les différentes actions possibles sur l'écran, le curseur et les caractères sont codés dans des fonctions séparées qu'il suffit d'appeler par une ligne de code. On peut donc très facilement afficher du texte, l'effacer, déplacer le curseur, les caractères et tracer des lignes et des carrés. Cela reste assez rudimentaire, mais c'est suffisant pour créer un environnement visuel. J'y ai ajouté les codes du clavier, de la souris et les sous-programmes qui peuvent être appelés en cliquant sur une icône ou via des touches de raccourcis.

La gestion du clavier est un peu particulière. Concrètement il y a deux méthodes pour afficher le résultat de la frappe d'une touche à l'écran. La plus simple est d'activer la fonction « echoON » qui établit un écho local entre le clavier et l'écran, au niveau du Minitel et sans passer par l'Arduino. C'est dans cette configuration que le Minitel fonctionne normalement et que l'Arduino n'a alors pas besoin d'interpréter ce qui est frappé au clavier. Cependant, dans le cas d'applications telles que TinyBasic+ il est impératif que chaque frappe sur le clavier génère le caractère correspondant dans l'Arduino afin qu'il puisse interpréter ce qui est tapé au clavier. Dans ce cas lorsqu'une touche est frappée c'est l'Arduino qui reçoit le signal en provenance du Minitel et lui renvoie alors l'ordre d'afficher à l'écran le caractère correspondant via la fonction « minitel.print() », tout en coupant l'écho local avec la fonction « echoOFF » (sans quoi les caractères frappés au clavier s'afficheraient deux fois).

Il est également possible de configurer le clavier différemment en fonction du programme qui est lancé. Par exemple pour le jeu « Marauder », l'écho local est coupé et seules les touches utiles au jeu et les raccourcis clavier sont activés. Une commande type « maraude_r = true » active alors cette

configuration spéciale. Lorsque le jeu est terminé ou que l'on utilise le raccourci clavier pour revenir au Menu, cela entraîne la remise à zéro des indicateurs de configuration (ici : « maraude_r = true »), ce qui évite les bugs en passant d'une application à l'autre.

Work in progress…

Voilà, sur le principe, ça fonctionne, mais vous l'aurez compris, tout cela est encore bien loin d'être fini ! Comme vous le voyez sur les photos, j'ai ajouté quelques options à mon système qui ne sont pas encore branchées ou au point (haut-parleur, ports pour manette de jeux, lecteur de carte SD…).
Au niveau des graphismes je me suis limité à utiliser les caractères de bases, mais le Minitel dispose d'un jeu de caractères semi-graphiques et d'un système de niveau de gris qui peuvent être combinés pour créer des images beaucoup plus abouties.

Malheureusement l'électronique n'étant pas mon activité principale je n'ai pas encore pu aller au bout de ce projet. Ma principale ambition était de créer un système modulaire que les gens pourraient s'approprier en écrivant des programmes qui seraient facilement installables au sein de ZARDOS, et ce, sans avoir à réécrire l'ensemble du code ou programmer une cartouche dédiée, car je voulais que le montage de base reste le plus simple possible.

Pour cela, j'avais imaginé de refondre entièrement le code de ZARDOS en une version 2.0 : chaque sous-programme y serait isolé dans une bibliothèque spécifique qui elle-même utiliserait les fonctions d'une bibliothèque « maîtresse » basée sur celle d'Eric Serandour. Il y aurait eu dans le code de base une demi-douzaine d'emplacements vides, mais ayant des noms prédéfinis, comme autant de cases vides. Ainsi, pour rajouter un programme il suffirait d'écrire une nouvelle bibliothèque ayant un nom correspondant à l'une des cases vides et en respectant des normes prédéfinies. L'installation du programme reviendrait donc à simplement rajouter une bibliothèque au code de base. J'aimais bien cette idée, mais je me suis heurté à un problème que, par inexpérience, je n'avais pas prévu : on peut utiliser plusieurs bibliothèques dans un même code Arduino, mais ces différentes bibliothèques ne peuvent pas s'appeler entre elles. Ce qui fait que mon système modulaire tel que je me l'imaginais ne pouvait pas fonctionner.

Mais je ne désespère pas de trouver une solution, en tout cas si parmi les lecteurs quelqu'un en a une, je suis preneur ! En tout cas d'un point de vue personnel j'ai énormément appris et cela va effectivement me servir pour mon travail, car je vais avoir besoin d'un terminal pour afficher les données de sondes de température, vitesse, etc., et commander des moteurs, chauffages et actionneurs via des raccourcis clavier et c'est ZARDOS et un Minitel que je vais utiliser, mais ça, c'est une autre histoire.

BRANCHEMENTS

- Dos du Minitel — Carte Arduino
- DIN 1 : RX <———-> TX (Tx1 sur Arduino Mega)
- DIN 4 : <—/ /—->
- DIN 2 : GND <———-> GND
- DIN 5 : +8,5V <—/ /—->
- DIN 3 : TX <—+—> RX (Rx1 sur Arduino Mega)

Résistance 10 kilo Ohms

+5V

port PS/2 souris	ARDUINO
DATA	<———-> 5
CLK	<———-> 6
GND	<———-> GND
5v	<———-> 5v

Codes & vidéos de démonstration sur : http://zardos.fr/

Bibliographie

- ***Montages autour d'un Minitel*** – Christian TAVERNIER – Éditions ETSF - 1994
- ***Spécifications Techniques d'Utilisation du Minitel 1B*** (STUM 1B) – France Telecom - 1986

CHOISIR SON LANGAGE DE PROGRAMMATION

Philippe BOULANGER
Manager des expertises C/C++ et Python
www.invivoo.com
INVIVOO BEYOND TECH

Un langage de programmation est un outil permettant de transformer un concept en un programme qui s'exécutera sur un ordinateur. Il existe beaucoup de langages de programmation et de nouveaux sont créés régulièrement. Comment choisir celui qui correspond à son besoin ? Quelles contraintes seront les conséquences de ce choix ? Que l'on soit un jeune développeur qui souhaite s'investir sur une technologie ayant un avenir ou un architecte/directeur de projet qui souhaite faire le bon choix technique (le langage le plus intéressant efficacité/coût) ou humain (un langage ayant une plus grande communauté de développeurs afin de faciliter les recrutements), le choix d'un langage est souvent important et engage pour une longue période (dans les banques des logiciels ayant 20 à 30 ans ne sont pas des cas isolés)…

A-T-ON DES CONTRAINTES ?

La première question à se poser est : a-t-on des contraintes techniques ou fonctionnelles ? En effet, répondre à une contrainte peut limiter les solutions techniques disponibles. Je vais vous en présenter quelques-unes :

- Je souhaite créer une application plug-in dans le logiciel FreeCAD (https://www.freecad.org/). FreeCAD est écrit en C++ et fournit soit des API C++ soit du scripting en Python. Et donc pour faire mon application je me retrouve limité sur mes choix…
- Dans le milieu de la finance, nous sommes bien souvent liés à des progiciels financiers ou à des choix d'entreprise (base de données, data lake, etc.). Les progiciels sont souvent couplés à des langages (Orchestrade à C#, Summit à C/C++ pour le core et C# pour le GUI par exemple)
- La contrainte peut être liée à des performances attendues. Dans le cas du trading à haute fréquence, on cherche à économiser les nanosecondes ce qui force le choix à des langages très efficaces comme le C++ ou le C…
- La contrainte peut être le choix d'un métier : dans quel domaine ai-je envie de travailler ? Lequel me garantira le plus de travail dans l'avenir, etc. ?
- etc.

Il est donc nécessaire de faire la liste de ses objectifs, quelles sont nos contraintes et qu'est ce que l'on souhaite éviter ?

DE QUELS LANGAGES PARLE-T-ON ?

Il existe de très nombreux langages dans la nature. Nous allons faire une liste (non exhaustive) des principaux langages parmi les plus utilisés. Nous rajouterons des exemples de code afin que le lecteur se fasse une idée de sa syntaxe.

> FORTRAN (1957)

Le premier langage commercial dont le nom est la contraction de « FORmula TRANslating ». C'est le langage utilisé pour le calcul intensif et c'est un des langages ayant la plus grande base de code scientifique installée. Basé, à l'origine, uniquement sur le paradigme impératif structuré, il a évolué au cours du temps pour supporter le paradigme objet. Son gros point faible est lié aux outils d'entrées/sorties : nombre de projets avaient des IO écrites en C.

> C (1969)

Inventé pour développer le système UNIX, c'est un langage de bas niveau permettant de créer des applications très efficaces. On le retrouve dans le cœur de la majorité des systèmes d'exploitation et dans les systèmes embarqués. Il supporte deux paradigmes de programmation : impératif structuré et fonctionnel. Certains choix dans la définition des pointeurs et dans la gestion mémoire nécessitent, de la part des développeurs, une très grande rigueur. Les coûts de maintenance des applications anciennes ont tendance à augmenter avec le temps…

> Pascal/Delphi (1970)

Conçu pour enseigner l'informatique, il a longtemps été le langage de l'éducation nationale française. Mais, malheureusement selon moi, il n'a jamais réussi à percer dans l'industrie : contrairement au C un pointeur sur un entier n'est pas du même type qu'un pointeur sur un tableau d'entiers. Il a été popularisé par Borland en 1983 grâce à *Turbo Pascal* (après la programmation en *Basic* je suis passé au *Turbo Pascal* 3 dans ma jeunesse). Il existe cependant des logiciels de comptabilité écrits en *Delphi* (est le successeur de Turbo Pascal).

> SQL (1974)

Le langage des bases de données relationnelles est utilisé dans chaque téléphone, dans chaque banque. Le langage normé est un langage de requêtes, mais ses variantes propriétaires (comme le *PL/SQL* d'Oracle) proposent des fonctionnalités qui en font des langages exécutés côté bases de données. Beaucoup d'applications, écrites en C++ par exemple, exécutent des requêtes SQL ou des procédures stockées pour lire/écrire les données dans le référentiel. Voici un exemple de fonction écrite en PL/SQL :

```
CREATE OR REPLACE FUNCTION get_total_sales(
  in_year PLS_INTEGER
)
RETURN NUMBER
IS
  l_total_sales NUMBER := 0;
BEGIN
  -- get total sales
  SELECT SUM(unit_price * quantity)
  INTO l_total_sales
  FROM order_items
  INNER JOIN orders USING (order_id)
  WHERE status = 'Shipped'
  GROUP BY EXTRACT(YEAR FROM order_date)
  HAVING EXTRACT(YEAR FROM order_date) = in_year;

  -- return the total sales
  RETURN l_total_sales;
END;
```

> Ada (1980)

Conçu pour le département de la défense américaine, c'est un langage très exigeant, mais qui permet de créer du code efficace et sécurisé. Il est employé dans l'aéronautique, dans le monde des transports ou dans des applications temps réels. Écrire un programme qui compile peut paraître un peu long, car le compilateur ne laisse rien passer : globalement les seules erreurs qui restent sont les erreurs d'algorithmes. Par contre c'est un langage verbeux.

> Objective C (1983)

C'est une extension du *C ANSI* qui est un langage orienté objet réflexif et qui a longtemps été le langage phare pour développer des applications sur macOS ou sur iPhone. Il n'est détrôné que depuis peu par *Swift*.

```
#import <Foundation/Foundation.h>

@interface SampleClass:NSObject
- (NSNumber *)multiplyA:(NSNumber *)a withB:(NSNumber *)b;
@end

@implementation SampleClass

- (NSNumber *)multiplyA:(NSNumber *)a withB:(NSNumber *)b {
  float number1 = [a floatValue];
  float number2 = [b floatValue];
  float product = number1 * number2;
  NSNumber *result = [NSNumber numberWithFloat:product];
  return result;
}

@end

int main() {
  NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

  SampleClass *sampleClass = [[SampleClass alloc]init];
  NSNumber *a = [NSNumber numberWithFloat:10.5];
  NSNumber *b = [NSNumber numberWithFloat:10.0];
  NSNumber *result = [sampleClass multiplyA:a withB:b];
  NSString *resultString = [result stringValue];
  NSLog(@"The product is %@",resultString);

  [pool drain];
  return 0;
}
```

> Matlab (1984)

Langage émulé dédié aux algorithmes mathématiques. Il est utilisé entre autres choses pour le traitement du signal. Traders et quants s'en servent aussi pour effectuer des calculs financiers. Aujourd'hui, il y a de nombreux projets de migration *Matlab* vers *Python*.

> C++ (1984)

Créé par Bjarne Stroustrup c'est le langage qui a été adopté par les entreprises pendant la période allant des années 80 à 2000… On retrouve C++ dans la plupart des clients lourds comme Excel et Word ou dans les applications bancaires nécessitant de grosses performances. BeOS était écrit en C++. Tout comme le C dont il hérite, C++ nécessite une grande rigueur. Nombre d'applications développées depuis plus de 15 ans existent dans les banques et l'industrie et très peu d'écoles forment les élèves à ce langage : il y a donc une pénurie de ressources.

```
#include <iostream>
using namespace std;

int main()
{
  int i, n;
  bool isPrime = true;

  cout << "Enter a positive integer: ";
  cin >> n;

  // 0 and 1 are not prime numbers
  if (n == 0 || n == 1)
  {
    isPrime = false;
  }
  else
  {
    for (i = 2; i <= n / 2; ++i)
    {
      if (n % i == 0)
      {
        isPrime = false;
        break;
      }
    }
  }
  if (isPrime)
    cout << n << " is a prime number";
  else
    cout << n << " is not a prime number";
```

```
  return 0;
}
```

> perl (1987)

Créé pour manipuler des fichiers textes, il est basé sur les langages C et *awk* ainsi que sur les outils *grep*, *sed*, *cut*, *test* et *expr*. Très utilisé par les ingénieurs système Unix pour générer des rapports à partir de logs ou pour faire du déploiement. Sa syntaxe complexe est peu lisible surtout que celui-ci contient beaucoup d'astuces comme des variables cachées en entrée et sortie de fonctions. Il est, de plus en plus, délaissé au profit de *python*. Mais on trouve encore beaucoup de scripts dans les banques, par exemple, ou en data science.

```
#!/usr/bin/perl
use warnings;
use strict;
my @a = qw(3 2 1 4 7 6);
print("unsorted: ", "@a", "\n"); # unsorted: 3 2 1 4 7 6
@a = sort {$a <=> $b} @a;
print("sorted:", "@a", "\n"); # sorted:1 2 3 4 6 7
```

> Python (1991)

Peu connu à ses débuts, il se taille la part du lion depuis 2014. C'est le langage du Big Data et de la Data Science. Sa syntaxe simple et lisible en fait un très bon langage pour l'apprentissage de la programmation : il a d'ailleurs été retenu par l'éducation nationale française pour l'enseignement de l'algorithme dès la classe de seconde. C'est le langage le mieux documenté sur internet, il dispose d'une communauté importante et d'un écosystème très complet. C'est le couteau suisse de l'informaticien.

```
from math import sqrt

def ListePremiers( n ):
    # initialisation
    N=n+1
    last=int(sqrt(N))
    t=[True]*N

    # boucle d'elimination
    for i in range(2, last):
        if t[i]:
            for j in range(i*i, N, i):
                t[j]=False

    # retourne le resultat
    return [i for i in range(2, N) if t[i]]
```

> Visual Basic (1991)

Microsoft l'a créé en le basant sur son modèle COM et sur le langage BASIC. Il permet le développement rapide d'applications. Facile à apprendre et facile à utiliser, il a beaucoup été utilisé par des non-développeurs (des personnes ayant un autre métier : assureur, trader, etc.) pour faire des démonstrateurs ou des logiciels internes… Une version .Net est maintenue.

> Visual Basic for Application (1993)

Langage de macro dans la suite Office de Microsoft, il a, peu à peu, été intégré dans d'autres softwares. Nombre de programmes tournent dans des pages Excel au sein des banques et n'oublions pas certains technico-commerciaux qui vous vendent du rêve via des « slidewares » : des démos dans des vignettes PowerPoint à grand renfort de macros… Voici un exemple de fonction VBA dans Excel :

```
Sub AddSerialNumbers()
    Dim i As Integer
    On Error GoTo Last
    i = InputBox("Enter Value", "Enter Serial Numbers")
    For i = 1 To i
        ActiveCell.Value = i
        ActiveCell.Offset(1, 0).Activate
    Next i

    Last:
        Exit Sub
End Sub
```

> Lua (1993)

C'est moteur de scripts conçus pour être facilement embarqués dans des applications C/C++… Bien que disposant de moins de bibliothèques que Python, il a fini par s'imposer dans les moteurs de jeux vidéo le développement réseau ou l'embarquer notamment. LuaJIT est une variante incluant un compilateur à la volée donnant au moteur Lua des performances proches du C.

```
require "core"
local items = core.get_items()
if items then
  for _, item in ipairs(items) do
    if item.name == "switch" then
      local device = core.get_device(item.device_id)
      if device and device.name == "NAME_OF_DEVICE" then
        core.set_item_value(item.id, true)
      end
    end
  end
end
```

> R (août 1994)

Dédié aux statistiques et à la science des données, ce langage a connu un regain d'intérêt avec le crise du covid. Mais il est en concurrence avec *Python* qui dispose d'un écosystème plus riche et qui bénéficie de meilleures performances.

```
# take input from the user
num = as.integer(readline(prompt="Enter a number: "))
factorial = 1
# check is the number is negative, positive or zero
if(num < 0) {
print("Sorry, factorial does not exist for negative numbers")
else if(num == 0) {
print("The factorial of 0 is 1")
} else {
```

```
for(i in 1:num) {
    factorial = factorial * i
}
print(paste("The factorial of", num ,"is",factorial))
}
```

> PHP (1994)

Ce langage a été démocratisé au sein des architectures LAMP (*Linux*, *Apache*, *MySql* & *PHP*) qui ont servi à bâtir les applications web dynamiques. Facebook en fait un grand usage et a travaillé à l'écriture d'un compilateur. La syntaxe est un mix entre C et perl…

```
<?php
$fruits = array ( "fruits"  => array   ( "a" => "orange",
                                         "b"  => "banana",
                                         "c" => "apple"
                                         ),
                  "numbers" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6
                                       ),
                    "holes"  => array (    "first",
                                        5 => "second",
                                             "third"
                                       )
                  );

// Quelques exemples pour retrouver les valeurs dans le tableau ci-dessus
echo $fruits["holes"][5];   // affiche "second"
echo $fruits["fruits"]["a"]; // affiche "orange"
unset($fruits["holes"][0]);  // efface "first"

// Création d'un tableau multidimensionnel
$juices["apple"]["green"] = "good";
?>
```

> Ruby (1995)

Langage associé au framework Ruby On Rails il a eu un bon succès jusqu'en 2008, mais est en érosion lente depuis si l'on en croit l'index Tiobe ou PyPL.

```
#
# This program evaluates polynomials.  It first asks for the coefficients
# of a polynomial, which must be entered on one line, highest-order first.
# It then requests values of x and will compute the value of the poly for
# each x.  It will repeatly ask for x values, unless you the user enters
# a blank line.  It that case, it will ask for another polynomial.  If the
# user types quit for either input, the program immediately exits.
#

#
# Function to evaluate a polynomial at x.  The polynomial is given
# as a list of coefficients, from the greatest to the least.
def polyval(x, coef)
  sum = 0
  coef = coef.clone          # Don't want to destroy the original
  while true
    sum += coef.shift        # Add and remove the next coef
    break if coef.empty?     # If no more, done entirely.
    sum *= x                 # This happens the right number of times.
  end
  return sum
end
```

> Java (1995)

Ce langage est très populaire depuis 1995. SUN a beaucoup investi dans l'organisation d'événements avec des évangélisateurs afin de le faire connaître ; et l'aide de Netscape en l'intégrant dans son navigateur a permis la naissance du web dynamique grâce aux applets. Le ramasse-miette intégré lui permet d'être plus accessible aux débutants que C++. Son écosystème est l'un des, si ce n'est le plus, riche de tous les langages. Appartenant désormais à la société Oracle qui a « rationalisé » son développement, les utilisateurs professionnels commencent à le remplacer.

> JavaScript (1996)

Un des langages emblématiques du web dynamique avec PHP. Un des plus utilisés pour faire des interfaces utilisateurs dans les applications web. Il a bien évolué jusqu'à intégrer un compilateur Just-In-Time… C'est un langage au typage dynamique qui déchaîne les passions comme Python…

> C# (2002)

C# est la réponse de Microsoft à Sun suite aux procès liés à Java. C# est un langage bâti autour de la plateforme .Net, il est un compromis entre C++ et Java : il reprend le meilleur des deux langages. Il est notamment utilisé dans les banques. Comme exemple, je pourrais citer le projet CrossOne (de la Société Générale) ou le progiciel financier Orchestrade qui ont été créés en C#.

> Scala (2004)

Scala est un langage fonctionnel et orienté objet conçu pour fonctionner initialement dans la JVM. Il a été utilisé pour développer Spark et Kafka utilisé dans le domaine du Big Data. Scala a depuis été porté dans la LLVM du MIT. Il donne du code plus compact que Java, mais est plus difficile à apprendre.

```
def createPrimes (MAX: Int) : Array[Boolean] = {
 val pri = (false :: false :: true :: List.range (3, MAX + 1).map (_ % 2 != 0)).toArray
 for (i <- List.range (3, MAX)
  if (pri (i))) {
   var j = 2 * i;
   while (j < MAX) {
    if (pri (j))
     pri (j) = false;
    j += i;
   }
  }
 pri
}
val MAX = 1000*1000
(1 to MAX).filter (createPrimes (MAX))
```

> Go (novembre 2009)

Inspiré du C et du Pascal, c'est un langage concurrent développé par Google. La concurrence est basée sur un système analogue aux coroutines et sur une synchronisation par passage de messages qui est plus intuitive que celle basée sur le modèle multi-threadé synchronisé par des sémaphores ou des mutex. Il a de bonnes qualités pour la programmation réseau.

```
package main

import (
  "fmt"
  "time"
)

func f(from string) {
  for i := 0; i < 3; i++ {
    fmt.Println(from, ":", i)
  }
}

func main() {
    f("direct")
    go f("goroutine")

    go func(msg string) {
        fmt.Println(msg)
    }("going")

    time.Sleep(time.Second)
  fmt.Println("done")
}
```

> Julia (août 2009)

Julia est un langage de programmation dédié au calcul scientifique, avec une syntaxe qui se veut familière pour les utilisateurs d'autres environnements comme Matlab, R, Scilab, Python, etc. Il fournit :

- un compilateur
- du typage dynamique
- du polymorphisme
- une exécution parallèle distribuée

des appels directs de fonctions C, Fortran et Python.
Voici un exemple de code Julia :

```
# functions can also be defined more succinctly
quadratic(a, sqr_term, b) = (-b + sqr_term) / 2a

# calculates x for 0 = a*x^2+b*x+c,
# arguments types can be defined in function definitions
function quadratic2(a::Float64, b::Float64, c::Float64)
  # unlike other languages 2a is equivalent to 2*a
  # a^2 is used instead of a**2 or pow(a,2)
  sqr_term = sqrt(b^2-4a*c)
  r1 = quadratic(a, sqr_term, b)
  r2 = quadratic(a, -sqr_term, b)
  # multiple values can be returned from a function using tuples
  # if the return keyword is omitted, the last term is returned
  r1, r2
end
```

> rust (juillet 2010)

Créé par la fondation Mozilla pour être fiable, concurrent et pratique, c'est un bon remplaçant pour le langage C pour l'embarqué ou la programmation système. Récemment accepté par Linus Torvald pour le développement de Linux : c'est un langage à surveiller.

```
struct Rectangle {
  width: u32,
  height: u32,
}

fn main() {
  let rect1 = Rectangle {
    width: 30,
    height: 50,
  };

  println!(
    "The area of the rectangle is {} square pixels.",
    area(&rect1)
  );
}

fn area(rectangle: &Rectangle) -> u32 {
  rectangle.width * rectangle.height
}
```

> Kotlin (juillet 2011)

Créé par Jetbrains et adoubé récemment par Google, il est devenu un langage pour le développement Android. Il permet de compiler aussi bien dans la JVM que dans le moteur JavaScript.

```
class, Car(var brand: String, var model: String, var year: Int) {
 // Class function
 fun drive() {
  println("Wrooom!")
 }

 // Class function with parameters
 fun speed(maxSpeed: Int) {
  println("Max speed is: " + maxSpeed)
 }
}

fun main() {
 val c1 =, Car("Ford", "Mustang", 1969)

 // Print property values
 println(c1.brand + " " + c1.model + " " + c1.year)

 // Call the functions
 c1.drive()
 c1.speed(200)
}
```

> TypeScript (février 2012)

Langage inventé par Microsoft pour augmenter la production de code et la sécurité en JavaScript. Il transcompile en JavaScript lui permettant d'être interprété par n'importe quel navigateur web ou moteur JavaScript.

```
interface User {
 name: string;
 id: number;
}

class UserAccount {
 name: string;
 id: number;

 constructor(name: string, id: number) {
  this.name = name;
  this.id = id;
 }
}

const user: User = new UserAccount("Murphy", 1);
```

> Swift (2014)

C'est le langage qu'Apple a créé pour remplacer Objective C. Il est utilisé pour faire des applications sur iPhone ou Mac : c'est la seule plateforme où il est réellement répandu.

```
let interestingNumbers = [
    "Prime": [2, 3, 5, 7, 11, 13],
    "Fibonacci": [1, 1, 2, 3, 5, 8],
    "Square": [1, 4, 9, 16, 25],
]
var largest = 0
for (_, numbers) in interestingNumbers {
    for number in numbers {
        if number > largest {
            largest = number
        }
    }
}
print(largest)
```

DÉFINIR DES CRITÈRES

Choisir une technologie pour un nouveau projet ou comme choix d'avenir pour son métier n'est pas chose facile. La première difficulté est de définir des critères objectifs et, si possible, mesurables. Trop souvent on entend les développeurs vanter les mérites de leur langage de prédilection ou descendre en flèche le(s) langage(s) détesté(s), mais sans recul sur aucune des technologies citées ni informations vérifiées et/ou vérifiables… Et, grand jamais, je n'ai dit que Python et C/C++ étaient les meilleurs langages et que Java ne servait à rien.

> Le domaine de développement

Si l'on souhaite travailler dans un domaine précis ou sur des problématiques précises, certains langages vont s'imposer d'eux-mêmes.

Calcul scientifique

C'est un domaine de niche (il y a peu d'emplois disponibles), mais qui est très exigeant et très riche en termes de technologies. C'est ce domaine qui fut le point de départ de l'informatique moderne: décrypter les messages codés de l'Allemagne nazie… Les langages phares sont :

- ***Fortran*** : probablement le langage ayant la base de code installée la plus importante. Il y a tellement de codes que les réécrire coûterait trop cher. Fortran 90 a ajouté la vectorisation dans sa syntaxe et le langage continu d'évoluer.
- **C++** + ***OpenMP*** : on le retrouve dans les programmes n'ayant pas à assumer un historique. OpenMP est une extension du compilateur qui permet de paralléliser les boucles. MPI peut aussi être utilisé pour faire de la programmation distribuée.
- ***Python*** + ***numpy*** + ***scipy*** : connaît un essor important depuis 2014 et l'arrivée des clouds. Numpy et scipy sont deux bibliothèques dédiées aux calculs et ayant des versions optimisées par Intel. On le retrouve notamment dans les calculs financiers dans les organismes bancaires, dans le domaine de la recherche. Pour les statistiques, on peut rajouter la librairie ***pandas***.
- ***Julia*** : dernier arrivé dans le monde du calcul, ce langage est un outsider proposant une syntaxe simple, moderne et adapté aux enjeux de la vectorisation et parallélisation. Il est basé sur la LLVM pour être portable.

Programmation système

C'est toujours un sujet d'actualité, les ordinateurs sont omniprésents dans nos vies (téléphones portables, box internet, ordinateurs, etc.) ; il y a donc des OS partout faisant souvent partie d'une lignée (Unix, BSD, Linux et Windows pour les plus connus, mais il y en a d'autres comme RTOS par exemple). Tous les OS d'aujourd'hui ont une partie core écrite en C et éventuellement en assembleur. Les langages les plus courants sont :

- **C** : c'est le langage le plus utilisé pour cette tâche
- ***assembleur*** : chaque processeur a un assembleur qui lui est propre rendant complexe les évolutions. D'autant plus que les nouveaux processeurs ont nombres d'unités travaillant en parallèle : écrire du code optimisé n'est pas au niveau de tout le monde. Il est donc réservé à des fonctions très précises.
- **C++** : BeOS a été écrit avec des API en C++. Moins performant que le C et plus gourmand en ressources, il n'est pas le mieux représenté des langages dans cette catégorie.
- ***rust*** : conçu pour être un concurrent à C/C++, il a été autorisé pour le développement dans le noyau Linux et Google compte s'en servir pour améliorer la sécurité du noyau d'Android.
- ***go*** : le langage de google a de bonnes qualités pour développer des outils autour des OS

Programmation embarquée

Nous faisons un usage de plus en plus important d'objets connectés ou embarqués : drones, montres connectées, imprimantes, thermomix, TV connectée, etc. Travaillant sur des processeurs aux faibles performances, la gestion de la mémoire et des performances est souvent une nécessité…

- **C** : c'est le langage le plus utilisé pour cette tâche
- ***assembleur*** : comme pour les OS.
- **C++** : Moins performant que le C et plus gourmand en ressources, il n'est pas le mieux représenté des langages dans cette catégorie.
- ***rust*** : il commence à faire son chemin.
- ***Micropython*** : popularisé par le Raspberry PI pour rendre plus accessible la programmation.

Big Data

Le Big Data est devenu un enjeu économique de très grande importance. Le domaine a crû en importance avec la venue du cloud. Les langages les plus utilisés sont :

- ***R*** : le langage des statistiques par excellence, mais en perte d'utilisation sauf dans les milieux de la recherche
- ***scala*** : spark et kafka, deux des grands outils utilisés dans les solutions Big Data, sont écrits dans ce langage… Il permet d'accéder aux performances, mais au prix d'un long apprentissage. Longtemps cantonné à la JVM, il est depuis peu dans LLVM.
- ***java*** : compatible JVM avec une forte base de développeurs, il est plus facile à apprendre que scala.
- ***python*** + ***numpy*** + ***scipy*** + ***pandas*** + ***matplotlib*** : il remplace petit à petit R. De plus avec docker et Kubernetes, on peut facilement déployer en production.

Machine Learning

Le Big Data est le point de départ, ensuite il faut analyser les données et en déduire des enseignements comme la reconnaissance des visages, la conduite autonome, etc. C'est le but de l'intelligence artificielle ou du deep learning (apprentissage profond). On y retrouve les mêmes langages que pour le Big Data, mais avec des bibliothèques différentes et souvent propres aux fournisseurs de cloud : PyTorch, TenserFlow, scikit-learn, etc...

- ***python*** + bibliothèques : c'est le langage phare.
- **C/C++** : pour implémenter les algorithmes de calcul interne et les fonctionnalités où la performance est primordiale.
- ***java*** : il suit C/C++ en taux d'utilisation.
- ***prolog*** : basé sur le paradigme logique c'est un des langages historiques de l'IA.
- ***lisp*** : premier langage commercial de l'IA, il est à la base de beaucoup de découverte et d'amélioration dans le monde de l'informatique.

Programmation web

L'économie est aujourd'hui centrée autour du web d'autant plus avec la crise du covid : l'utilisation des services de commandes et livraisons a fortement augmenté. Les GAFA ont toutes au moins un business model web ; des banques comme Boursorama ou N26 n'ont plus d'agence et sont 100% web et téléphone. Alors HTML et CSS sont des langages de représentation, mais non des langages de programmation, mais ils restent nécessaires pour définir la disposition des éléments. Les principaux langages dans l'ordre de popularité :

- ***JavaScript*** : il peut être utilisé du côté client (avec React ou Angular) ou du côté serveur (avec node.js). Il est en tête des langages web.
- ***Python*** : avec des bibliothèques comme Django, flask ou CherryPy il s'impose petit à petit pour les applications web ou REST. Ma richesse de sa bibliothèque
- ***TypeScript*** : créé par Microsoft pour compenser les carences de JavaScript. TypeScript permet d'en faciliter le développement.
- ***PHP*** : dédié au développement côté serveur, il est présent dans de nombreux sites (Facebook, Wordpress, Joomla, etc.). Il se couple facilement avec la base de données MySQL.
- ***Ruby*** + ***Rails*** : intuitif, facile à apprendre, il a une syntaxe propre. Ce framework web libre écrit en Ruby est la base des sites web suivant : GitHub, Shopify, Airbnb, Groupon, GoodReads et Kickstarter.

Ces 2 langages peuvent aussi être utilisés du côté serveur notamment dans les banques :

- ***Java***
- ***C#*** + ***ASP.NET***

Jeux vidéo

Le monde du jeu vidéo rapporte énormément d'argent chaque année. On les retrouve aussi bien sur des consoles de salon, des ordinateurs individuels ou des smartphones. Le besoin est souvent très orienté performance. Voici les langages pertinents dans cette industrie pour les ordinateurs et les consoles:

- **C/C++** : leurs performances les rendent tout indiqués pour développer les moteurs des jeux professionnels. Il est notamment utilisé pour World of Warcraft qui a marqué une génération.
- ***lua*** : langage de script souvent utilisé pour développer les mods (notamment pour World of Warcraft)
- ***java*** : utilisé entre autres pour Runescape ou Minecraft
- ***C#*** : excellent compromis entre C++ et java en termes de syntaxe ou de fonctionnalités. Il est utilisé notamment dans Hearthstone et Super Mario Run. Le framework XNA le rend indiqué pour développer sur Xbox.
- ***JavaScript*** : utilisé pour faire des jeux dans des browsers web

Pour les jeux mobiles, reportez-vous au paragraphe correspondant.

Programmation mobile

Les arrivées des Psion et Palm puis des iPod et iPhone ont été une révolution tant dans les habitudes des gens (il est courant de voir les gens vissés à leurs portables) que dans l'économie (elle pèse financièrement très lourd). Par programmation mobile nous allons cibler les tablettes et les smartphones pour les OS majoritaires Android et iOS. Les langages utilisés sont :

- Pour les plateformes Android
  - ***Java*** : langage officiel, mais longtemps source de conflits avec Oracle
  - ***Kotlin*** : langage secondaire de la plateforme, mais qui a le vent en poupe et qui est le langage à privilégier selon Google
  - **C/C++** : disponible et permettant de tirer au mieux parti des performances du hardware.
- Pour l'écosystème iOS

Abonnez-vous à Programmez! Abonnez-vous à Programmez! Abonne

PROGRAMMEZ!
Le magazine des développeurs

NOS CLASSIQUES

1 an → 10 numéros (6 numéros + 4 hors séries) — 55€ *(1)

2 ans → 20 numéros (12 numéros + 8 hors séries) — 90€ *(1)

Etudiant
1 an → 10 numéros (6 numéros + 4 hors séries) — 45€ *

Option : accès aux archives — 20€

** Tarifs France métropolitaine*

(1) Au lieu de 69,90 € ou 139,80 € selon l'abonnement, par rapport au prix facial.

ABONNEMENT NUMÉRIQUE

PDF ..... 45€
1 an

Souscription directement sur
www.programmez.com

Offres spéciales 2022

1 an Programmez!

+ 1 mois d'accès à la bibliothèque numérique ENI — **56€**

1 mois d'accès offert à la bibliothèque numérique ENI, la plus grande bibliothèque informatique française. Valeur : 49 €

2 ans Programmez! — 109€*

+ 1 mois d'accès à la bibliothèque numérique ENI
+ 1 an de Technosaures (2 numéros)

1 mois d'accès offert à la bibliothèque numérique ENI, la plus grande bibliothèque informatique française. Valeur : 49 €. Prix abonnement Technosaure : 29,90 €

* au lieu de 119,9 €

Sous réserve d'erreurs typographiques

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
PROGRAMMEZ, Service Abonnements
57 Rue de Gisors, 95300 Pontoise

- ☐ **Abonnement 1 an : 55 €**
- ☐ **Abonnement 2 ans : 90 €**
- ☐ **Abonnement 1 an Etudiant : 45 €**
  Photocopie de la carte d'étudiant à joindre
- ☐ **Option : accès aux archives 20 €**

- ☐ **1 an Programmez! : 56 €**
  + 1 mois d'accès à la bibliothèque numérique ENI
- ☐ **2 ans Programmez! : 109 €**
  + 1 mois d'accès à la bibliothèque numérique ENI
  + 1 an de Technosaures (2 numéros)

☐ Mme ☐ M. Entreprise : ..... Fonction : .....

Prénom : ..... Nom : .....

Adresse : .....

Code postal : ..... Ville : .....

Adresse email indispensable pour la gestion de votre abonnement

E-mail : ..... @ .....

☐ Je joins mon règlement par chèque à l'ordre de Programmez !

☐ Je souhaite régler à réception de facture

*** Tarifs France métropolitaine**

Offres pouvant s'arrêter à tout moment, sans préavis **PROG 253**

Les anciens numéros de PROGRAMMEZ!

Le magazine des dévs

241

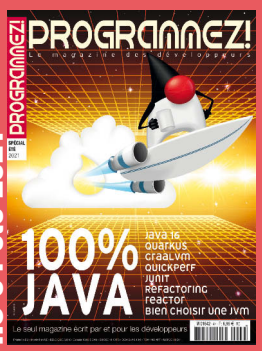
242

243

HS 02

246

HS 04 été 2021



249

HS 05 automne 2021

250

251

252

Complétez votre collection....

Tarif unitaire 6,5 € (frais postaux inclus)

☐ *241*	: __ ex	☐ *246*	: __ ex	☐ *250*	: __ ex
☐ *242*	: __ ex	☐ *HS4 été 2021*	: __ ex	☐ *251*	: __ ex
☐ *243*	: __ ex	☐ *249*	: __ ex	☐ *252*	: __ ex
☐ *HS2 automne 2020*	: __ ex	☐ *HS5 automne 2021*	: __ ex		

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

soit __ exemplaires x 6,50 € = __ € ..... __ € soit au **TOTAL** = __ €

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

Prénom : __ Nom : __

Adresse : __

Code postal : __ Ville : __

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

Offres pouvant s'arrêter à tout moment, sans préavis **PROG 253**

- ***Objective-C*** : langage d'origine, il accuse le poids des années.
- ***swift*** : c'est le langage principal des plateformes Apple. Il est conçu pour interopérer avec Objective-C.

• Multi-plate-forme :
- ***HTML*** + ***JavaScript*** : il suffit d'un navigateur, c'est une solution simple accessible à tous. Angular et React sont des frameworks permettant d'aller plus loin. Cordova est un autre framework utilisé par Wallmart et Adobe.
- ***C#*** : Microsoft a fourni avec Visual Studio pour Xamarin un environnement multi-plateforme iOS, Android et autres…
- ***Dart*** + ***Flutter*** : il peut aussi être utilisé. Dart a été inventé par Google en 2011.

Ingénieur système

Les ingénieurs systèmes, qui passaient beaucoup de temps à mettre à jour des paramètres systèmes, sont de plus en plus amenés à développer pour automatiser les tâches, déployer des mises à jour avec Ansible ou créer des conteneurs docker/Kubernetes. Les langages que l'on retrouve pour ce métier sont :

• ***perl*** : langage historique utilisé par les ingénieurs systèmes sur les plateformes UNIX. Sa syntaxe complexe et ses astuces internes n'en font pas un langage facile à apprendre ou à maintenir.
• ***Python*** : c'est le langage qui s'impose de plus en plus, il dispose des mêmes fonctionnalités que perl avec une syntaxe plus abordable et couvre fonctionnellement plus de besoins que perl. Ansible (l'outil de déploiement) a été écrit en Python ; de même faire des conteneurs docker ou Kubernetes s'automatise très bien en python.
• ***bash*** : c'est un langage de script, mais qui n'est pas adapté pour les gros projets
• ***powershell*** : très utilisé pour les développements sous Windows
• ***C/C++*** : pour les développements bas niveau

Le monde des réseaux

On peut séparer différentes utilisations : programmation d'outil réseau et/ou automatisation de tâches pour les ingénieurs réseau.

La programmation d'outils est faite par des développeurs qui ont reçu des formations avancées en programmation. Ils vont utiliser des technologies orientées performance et mémoire (§ Performances & écologie). Les langages que nous allons trouver sont :

• ***C/C++*** : ce sont les langages historiques du domaine
• ***rust*** : langage qui monte et qui est performant…
• ***go*** : le langage de Google propose des fonctionnalités intéressantes pour la gestion efficace des I/O asynchrones. Avec son ramasse-miettes il est de plus haut niveau.
• ***python*** : outsider inattendu, python dispose des fonctionnalités utiles et une simplicité de mise en œuvre par rapport aux autres langages

Les ingénieurs réseau se préoccupent de la mise en place, de la configuration et de l'exploitation des réseaux : ils utilisent des outils et automatisent des tâches répétitives via des scripts et de petits outils : ce ne sont pas des développeurs. Les outils de déploiement des routeurs CISCO sont basés sur Ansible ce qui impacte les choix de langages :

• ***python*** : Ansible est écrit en python. Les opérateurs de téléphonie investissent, aujourd'hui, sur cette technologie.
• ***perl*** : langage de script historique, il est petit à petit abandonné
• ***ksh*** ou ***bash*** : adaptés à l'automatisation de petite tâche, ils ne sont pas faits pour développer de gros scripts

> La facilité d'apprentissage

Certains langages sont plus complexes à apprendre que d'autres et cela est dû soit à des éléments de syntaxes complexes soit à des concepts plus compliqués : la syntaxe des pointeurs et la rigueur nécessaire pour bien gérer la mémoire rendent le C et le C++ plus difficiles à maîtriser que des langages avec des ramasse-miettes. Si on en croit « Interview Kickstart », les sept langages les plus simples à apprendre sont :

• HTML
• JavaScript
• Java
• PHP
• Python
• C
• Go

Et les plus complexes sont :

• Haskell
• C++
• ASM
• Prolog
• LISP
• Rust
• Esoteric languages (comme brainfuck)

Python est l'un des plus simples à apprendre : en France il a progressivement remplacé Pascal pour l'apprentissage en classes préparatoires et désormais il fait partie des outils utilisés à partir de la classe de seconde pour enseigner l'algorithmique. Il est aussi devenu le langage des calculatrices programmables grâce à la société Numworks. Sa syntaxe est proche du langage naturel.

> La communauté

La communauté est essentielle, car c'est la garantie de pouvoir trouver des outils et de l'aide en cas de problème. Une communauté importante démontre que le langage intéresse et répond à des besoins : c'est donc un signal positif. Il y a différents indicateurs qui montrent la santé d'une communauté, mais l'un de ce que je préfère est basé sur les statistiques des projets hébergés sur GitHub (la plus grosse plateforme de projets collaboratifs : plus de 128 millions de dépôts publics). Voici les volumes de pull-requests en pourcentages du volume globale (accessible sur le site : https://madnight.github.io/githut/#/pull_requests/2021/4). **Figure 1 et 2**

Il en ressort que python est le langage sur lequel il y a eu le plus d'activités sur GitHub en 2021, suivi par JavaScript puis Java. Un second indicateur, qui me semble pertinent, est le nombre de recherches de tutoriel pour apprendre le langage (via des requêtes sur Google). En effet, un langage qui n'at-

tire plus les nouveaux développeurs est langage dont l'intérêt est en baisse et qui va devenir un langage de niche (comme Fortran ou Cobol) voir disparaître. Cet index est l'index PyPL : **Figure 3**

Et Python reste le langage le plus recherché : je vous l'avais dit, c'est le meilleur langage…

> L'employabilité

Le monde du travail est un monde de l'offre et de la demande, pour trouver facilement un emploi il faut que le nombre de postes disponibles soit proche ou plus important que le nombre de développeurs disponibles. Pour ce point, on peut porter attention au résultat du developer survey du site codingame (https://www.codingame.com/work/fr/codingame-developer-survey-2021/): **Figure 4**

Pour le go, on se rend compte qu'il y a plus de postes que de développeurs, de même pour l'ObjectiveC, le Scala, le Swift, le Typescript ou le Rust.

> Performances & écologie

Dans un domaine comme le temps réel, on a besoin d'un résultat en un temps contraint : on aura besoin d'un programme qui sera rapide/performant. Dans le domaine du calcul scientifique, comme la simulation numérique de crash-test de voiture ou le calcul de la météo du lendemain, on veut un résultat le plus rapidement possible. La vitesse d'exécution ainsi que la facilité à optimiser le code sont des critères qui peuvent conditionner le choix d'une technologie.

Pour travailler sur de grandes quantités de données ou sur des applications embarquées, on devra maîtriser sa consommation mémoire afin d'éviter le dépassement des capacités mémoires de l'ordinateur cible. Nous traitons des problèmes scientifiques de plus en plus grands, nous manipulons de plus en plus de données, l'optimisation mémoire peut devenir une nécessité. Et malheureusement tous les langages ne permettent pas une gestion fine de ces problématiques.

Si notre application doit être exécutée sur une grille de calculs H24 et 7J/7 ou sur un dispositif mobile (un téléphone par exemple) on préférera avoir une application moins énergivore :

- soit pour diminuer les coûts en électricité (maintenir les bitcoins consomment autant d'énergie électrique qu'un pays comme l'Argentine si l'on en croit les derniers rapports sur le sujet). Or le prix de l'énergie ne fait que croître ces dernières années.
- soit pour augmenter la durée entre 2 recharges de batteries pour un téléphone portable ou un drone.

À l'heure du réchauffement climatique et d'une énergie de plus en plus chère, réduire son empreinte carbone est à la fois un choix économique et moral.

Cette problématique a été abordée par un groupe de chercheurs portugais en 2017 dans le papier « Energy Efficiency across Programming Languages / How Do Energy, Time, and Memory Relate? ». Voici un classement des langages en fonctions des 3 critères : **Figure 5**

Mais souvent, le besoin est une combinaison de deux de ces critères. Les chercheurs ont alors effectués un tri selon des combinaisons de critères en employant le principe de Pareto : **Figure 6**

Figure 1

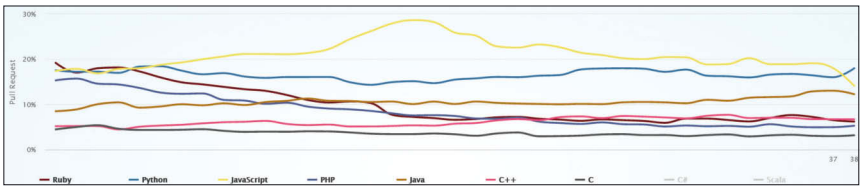


Figure 2

# Ranking	Programming Language	Percentage (YoY Change)
1	Python	17.926% (+1.438%)
2	JavaScript	14.058% (-4.714%)
3	Java	12.208% (+0.662%)
4	TypeScript	8.472% (+1.818%)
5	Go	8.161% (+0.027%)
6	C++	6.670% (-0.331%)
7	Ruby	6.165% (-0.783%)
8	PHP	5.252% (-0.322%)
9	C#	3.372% (-0.301%)
10	C	3.150% (+0.023%)

Figure 3

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %
6		PHP	6.06 %	+0.0 %
7		R	4.19 %	+0.3 %
8		Objective-C	2.27 %	-1.4 %
9		Swift	1.91 %	-0.2 %
10		TypeScript	1.74 %	-0.0 %

Figure 4

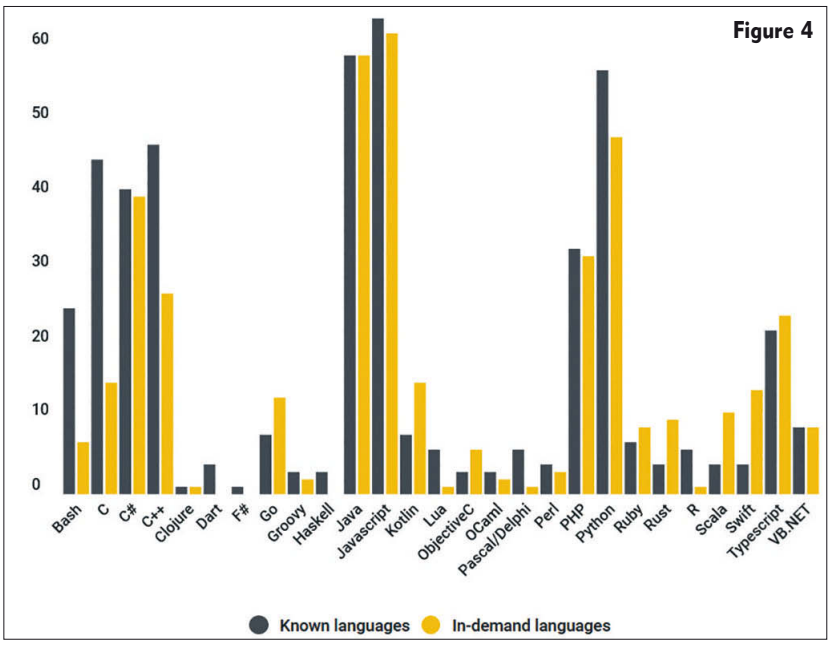


Figure 5

Normalized global results for Energy, Time, and Memory

Total

	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Globalement, on retrouve souvent les mêmes langages dans les 6 langages les plus efficaces : C, C++, rust, go, Pascal et Fortran…

CONCLUSION

Comme a dit Abraham Marslow : « Si le seul outil que l'on a sous la main est un marteau, alors tous les problèmes finissent par ressembler à des clous ». Appliqué à l'informatique cela nous donne : les langages que je connais conditionnent ma capacité à résoudre des problèmes de manière efficace. Sans programmation orientée objet, je ne pourrais pas créer facilement une application comme un modeleur géométrique 3D. En conclusion, aucun langage n'est parfait et la plupart ne couvrent pas tous les besoins : certains langages sont adaptés à certaines classes de problèmes et inadaptés à d'autres. Baser son avenir sur une combinaison de plusieurs langages me semble plus sûr même si cela nécessite un investissement plus important.

Figure 6

Pareto optimal sets for different combination of objectives.

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C • Pascal • Go	C	C • Pascal	C • Pascal • Go
Rust • C++ • Fortran	Rust	Rust • C++ • Fortran • Go	Rust • C++ • Fortran
Ada	C++	Ada	Ada
Java • Chapel • Lisp • Ocaml	Ada	Java • Chapel • Lisp	Java • Chapel • Lisp • Ocaml
Haskell • C#	Java	OCaml • Swift • Haskell	Swift • Haskell • C#
Swift • PHP	Pascal • Chapel	C# • PHP	Dart • F# • Racket • Hack • PHP
F# • Racket • Hack • Python	Lisp • Ocaml • Go	Dart • F# • Racket • Hack • Python	JavaScript • Ruby • Python
JavaScript • Ruby	Fortran • Haskell • C#	JavaScript • Ruby	TypeScript • Erlang
Dart • TypeScript • Erlang	Swift	TypeScript	Lua • JRuby • Perl
JRuby • Perl	Dart • F#	Erlang • Lua • Perl	
Lua	JavaScript	JRuby	
	Racket		
	TypeScript • Hack		
	PHP		
	Erlang		
	Lua • JRuby		
	Ruby		

Créer un jeu d'aventure avec AGE

AGE est un logiciel gratuit permettant de créer des jeux d'aventure Point & Click ou Visual Novel grâce à une interface épurée et basée essentiellement sur une approche scénaristique. Dans ce tutoriel, nous allons prendre en main l'outil en découvrant les différents éditeurs proposés au sein du logiciel afin d'apprendre rapidement à créer son propre jeu d'aventure.
Site officiel : **https://www.seccia.com/age**

Sylvain Seccia
Auteur et Développeur
Créateur de jeux vidéo et de logiciels depuis la fin des années 90, Sylvain s'est fait connaître par son jeu d'aventure Désiré en 2016 en remportant de nombreux prix.
https://www.seccia.com

Je vous invite à regarder d'abord la petite vidéo du tutoriel montrant l'objectif à atteindre avec les énigmes à résoudre. Ouvrez ensuite le projet dans lequel j'ai déjà préparé le décor et la création des objets avec leurs animations pour nous concentrer sur l'implémentation des interactions. **Figure 1**

La version finale se trouve dans le dossier final_project.

Le scénario

L'éditeur de scénario est au cœur de l'environnement, permettant de définir le déroulement narratif du jeu. Il n'est donc pas nécessaire de programmer une machine à état pour contrôler les énigmes puisque **AGE** propose déjà un outil prêt à l'emploi. Il suffit d'ajouter des boîtes nodales au scénario et de les connecter entre elles. Au runtime, un interpréteur se charge d'actualiser les états lorsque le créateur du jeu décide de valider une énigme à la suite d'une action du joueur. En d'autres termes, le créateur n'a besoin que de notifier la validation d'une boîte lorsque le joueur résout l'énigme en question. Les scripts s'intègrent aux boîtes nodales et aux événements des éditeurs. Vous trouverez la liste exhaustive des fonctions du langage en cliquant sur le bouton de la page d'aide en bas de la fenêtre.
La première étape consiste à définir les énigmes de notre histoire. Reproduisez le schéma du scénario entre les boîtes START et END. La boîte Puzzle est la plus courante car elle sert à définir les actions du joueur. Il faut imaginer la boîte bleue comme une mini boucle infinie où la seule façon d'en sortir est de valider l'énigme. Il existe plusieurs manières de le faire : soit par le code via la fonction success soit par les propriétés de certains éditeurs. **Figure 2**

Nous pouvons dès à présent écrire le code des boîtes car ce sont des scripts simples qui ne font que changer l'état de la partie. Nous ne sommes pas encore aux interactions, nous nous occupons de la vue d'ensemble de notre projet. Cliquez sur #getPie et ajoutez l'instruction **take P_HERO O_PIE** dans le script EXIT. Il existe deux modes d'édition pour les scripts : le mode classique et le mode assisté. Le mode peut être changé à tout moment en cliquant sur l'icône souris en haut à droite de la fenêtre principale. À vous de choisir le mode qui vous convient en sachant que le tutoriel se base sur l'édition classique. La fonction take permet tout simplement d'ajouter un objet à l'inventaire du joueur, en l'occurrence de P_HERO. Même opération pour #getCard et #getGlue avec les instructions respectives suivantes : **take P_HERO O_PRESSCARD** et **take P_HERO O_GLUE**. L'énigme #applyGlue est légèrement différente puisque nous devons retirer deux objets de notre inventaire pour en ajouter un troisième qui sera l'association des deux premiers. C'est-à-dire que le joueur devra utiliser la colle avec la photo pour obtenir un nouvel objet : la photo collante. Écrivez les deux instructions : **dump O_GLUE O_PHOTO** puis à la ligne **take P_HERO O_PHOTOGLUE**. La fonction dump enlèvera donc de l'inventaire les deux objets indiqués. Rebelotte pour l'énigme #pastePhoto où le joueur doit coller sa photo sur une carte de presse volée : **dump O_PRESSCARD O_PHOTOGLUE** puis **take P_HERO O_PRESSCARD2**. Nous en avons fini avec l'éditeur de scénario. On pourrait jouer des sons, changer l'état d'un dialogue, d'un objet, la valeur d'une variable… Par contre on évitera d'écrire des actions plus complexes, ce n'est pas l'endroit approprié.

Figure 1

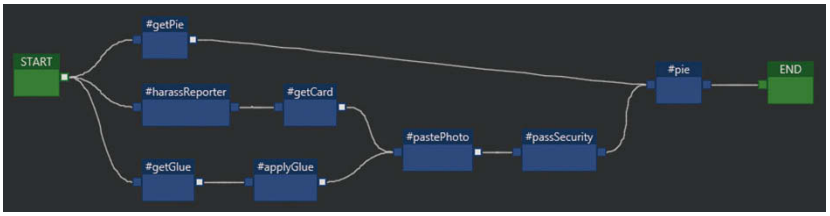


Figure 2

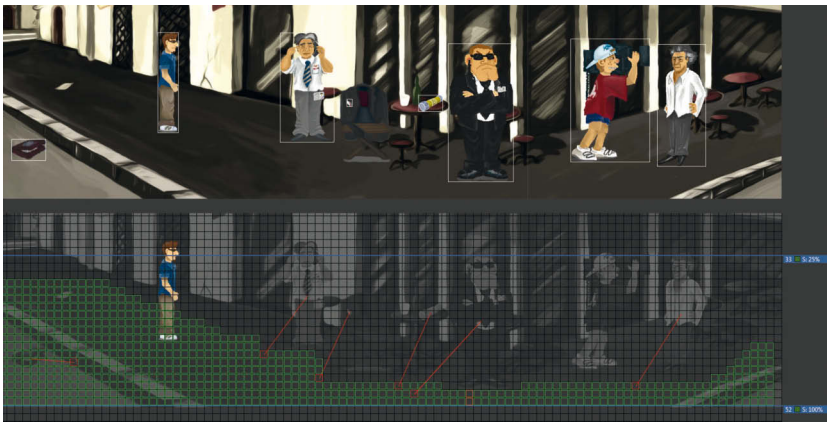


Figure 3

Figure 4

La scène et les objets

Dans l'éditeur de scènes, la première étape consiste à placer les objets et de définir le chemin du héros. Pour ajouter un objet, il suffit soit de cliquer sur le bouton en forme de diamant depuis la barre d'outils soit d'appuyer sur la touche O. Depuis la boîte de dialogue, ajoutez les objets suivants : O_HERO, O_REPORTER, O_GUARD, O_CAMERAMAN, O_BOTUL, O_PIE, O_GLUE, O_PRESSCARD. Placez-les ensuite comme sur le modèle de la capture d'écran. Tous les objets sont fixes à l'exception du héros. Pour éditer son déplacement, double-cliquez sur ce dernier pour afficher la grille par défaut. Les cases vertes définissent la zone de marche. Pour en ajouter, maintenez la touche Shift enfoncée et balayez les cases à l'aide du bouton gauche de la souris. Sans la touche Shift, vous pourrez les sélectionner pour les éditer ou les supprimer. Les deux étiquettes bleues à droite permettent de changer la taille du personnage par rapport à sa position Y afin de reproduire un effet de profondeur et d'éloignement. Choisissez les valeurs qui vous semblent cohérentes. Pour voir en aperçu directement dans l'éditeur, maintenez la touche Shift enfoncée et déplacez le curseur de la souris sans cliquer. **Figure 3**

Les cases rouges sont reliées aux objets. Cliquez sur une case proche du journaliste et changer la propriété Linked object du groupe SELECT object en choisissant O_REPORTER. Cela permet à **AGE** de déplacer le héros à cette case lorsque le joueur clique sur le journaliste, et ainsi recevoir un appel à l'événement SELECT uniquement lorsque la case sera atteinte. C'est le principe du point and click et cette partie peut être configurée sans code. Faîtes de même pour tous les objets. Pour la carte de presse et la tarte, vous pouvez ouvrir la propriété IF resolved et renseigner le champ Animation en choisissant TAKE pour animer le héros après avoir atteint la case, juste avant d'appeler l'événement. Pour le garde, en plus de SELECT Object, vous devez renseigner les champs Linked object A/B avec les objets : O_PRESSCARD2 et O_GUARD. L'événement USE est appelé lorsque le joueur combine deux objets de l'inventaire ou utilise un objet de l'inventaire avec un objet présent dans la scène. C'est le même principe pour Botul avec les objets : O_PIE et O_BOTUL. Par contre il ne s'agit plus de l'animation TAKE mais de l'animation PIE. Une dernière chose reste à faire, qui nous servira à bloquer le personnage s'il tente de s'approcher trop de la star, est de définir des triggers. Sélectionnez les cases près du garde, changez leur nom en GUARD et activez la propriété Event pour pouvoir recevoir l'événement CELL IN. Vous pouvez à présent double-cliquer sur le décor pour quitter l'édition de la grille et appuyez sur la touche F6 pour tester le jeu actuel.

Les dialogues

L'éditeur de dialogues permet d'écrire et de structurer vos conversations interactives. Pour un jeu d'aventure narratif, vous serez souvent amené à travailler en parallèle sur l'éditeur de scènes et l'éditeur de dialogues pour déclencher des changements d'états. Vous pouvez faire l'impasse sur cet éditeur si votre gameplay est essentiellement basé sur des énigmes d'objets sans texte narratif. Pour des raisons pratiques, durant une conversation, le joueur n'a pas la possibilité de sauvegarder la partie. Il doit d'abord quitter la conversation lorsqu'une liste de choix lui est proposée. Un accès rapide à la langue courante permet de traduire ou de vérifier les phrases sans exporter les fichiers CSV. Les boîtes rouges permettent de proposer au joueur un choix de répliques à différents moments de la conversation. Ces répliques sont associées au player courant. Les boîtes bleues permettent de donner la réplique à un personnage présent dans la scène.

Par convention, il est préférable de reprendre le nom de la scène pour nommer l'asset lorsque les répliques sont liées aux actions de la scène et qu'il n'y a pas ou peu de choix dans la discussion. Car les conversations seront plutôt stockées dans des assets à part en reprenant le nom de l'interlocuteur (asset du personnage). Les phrases bateaux qui serviront à meubler les actions non prévues seront placées dans D_HERO.

Je précise que les assets D_CAFE et D_HERO sont préremplis pour ce tutoriel. Ouvrez quand même le premier asset pour remarquer la présence de plusieurs répliques à la racine. Le titre de la boîte spécifie le personnage de la réplique. Nous les utiliserons plus tard. En revanche dans D_HERO, la boîte possède trois répliques avec deux séparations. Cela permet de choisir aléatoirement une phrase différente en utilisant le même identifiant sans alourdir le code. Ouvrez l'asset D_REPORTER pour commencer l'écriture de notre mini conversation entre notre héros et le journaliste en reproduisant l'arborescence de la capture d'écran. **Figure 4**

La première boîte est vide. C'est une particularité qui permet de lancer le choix automatiquement (et d'atteindre les répliques sous-jacentes) s'il est placé avant les autres choix. La couleur est différente parce que ce choix est invisible au début de la partie et se débloque lorsque les deux répliques bleues ont été prononcées. Pour ce faire, sélectionnez le choix concerné, maintenez la touche ALT enfoncée et cliquez

sur la première boîte bleue puis la seconde. Vous verrez apparaître une nouvelle icône sur ces dernières avec les liens respectifs. Il est possible d'obtenir le même résultat par le code à l'aide de la fonction showChoice mais c'est forcément plus compliqué surtout lorsque les conditions sont multiples. Notez l'icône *interrupteur* dans la première et dernière boite permettant de quitter prématurément la conversation. Il s'agit de la propriété Exit. L'icône *roue* indique que la boîte possède du code, en l'occurrence l'instruction suivante pour changer l'animation du journaliste et le faire retourner après l'avoir dérangé : **anim O_REPORTER BUSY**. Le joueur pourra ainsi piquer sa carte de presse sans être vu. L'icône *puzzle* informe une validation d'énigme à ce moment précis. Pour faire apparaître cette icône, sélectionnez la boîte puis l'énigme #harassReport dans la propriété Success on say. Enfin, l'icône *œil* de la dernière boîte permet de rendre le choix toujours visible. Par défaut, les choix disparaissent de la liste après les avoir utilisés. Il s'agit de la propriété Never hide.

Les scènes et les événements

Une fois les objets avec les cellules connectées et les principaux dialogues en place, nous pouvons retourner dans l'éditeur de scènes pour ajouter les événements afin d'interagir avec le joueur. Il y existe évidemment plusieurs façons de procéder selon votre projet, tout est réalisable en se servant uniquement des événements et goals sans passer par l'éditeur de rôles. Étant donné que ce tutoriel a pour objectif d'aborder les principaux outils du logiciel, je vous propose de séparer les actions clés (qui débloquent les énigmes) des actions secondaires que nous allons mettre en place maintenant dans l'éditeur de scènes.

Sélectionnez le journaliste, faites un clic droit et choisissez l'événement SELECT. Dans le champ script, écrivez simplement **talk D_REPORTER**. La fonction talk permet de lancer une conversation. Puisque nous avons précédemment lié une case avec le journaliste pour l'événement SELECT, **AGE** se chargera de déplacer le héros vers cette case avant d'appeler l'événement. Si entre temps, le joueur décide de reprendre la main, le déplacement sera interrompu et aucun événement ne sera appelé. Il faudra alors recliquer sur le journaliste. Reproduisez la manip avec le garde mais cette fois-ci, écrivez le code **talk D_CAFE 16** pour lancer une seule réplique car nous ne souhaitons pas parler avec lui. Faîtes de même pour Botul en écrivant **talk D_CAFE 6**.

L'étape suivante consiste à sélectionner la carte de presse et gérer le cas où le joueur n'aurait pas encore dérangé le journaliste et ne peut donc pas récupérer la carte. Le code nécessite une condition :

```
if_not resolved #harassReporter
 talk D_CAFE 1
end
```

Enfin il faut gérer le cas où le joueur tenterait de dépasser le garde pour atteindre Botul. Nous avons déjà mis en place les cellules de la grille pour intercepter l'événement. Cliquez sur le bouton New pour ajouter l'événement CELL IN avec O_HERO en premier paramètre et GUARD en second paramètre (le nom de la case). Dans le script, écrivez le code suivant :

```
if_not resolved #passSecurity
 talk D_CAFE 8
 return 1
end
```

Note : *si le joueur n'a pas encore résolu l'énigme, nous devons lancer un dialogue et retourner la valeur 1 pour annuler le déplacement et repositionner le héros à la case précédente.*

Toutes les actions secondaires sont en place, vous pourrez en rajouter par la suite pour meubler davantage la scène. Avant de passer aux actions clés, faisons un petit tour dans l'éditeur de players.

Le player

Lorsqu'il faudra combiner des objets, nous devrons utiliser l'événement USE au lieu de SELECT. Contrairement à SELECT, les combinaisons peuvent être exponentielles à force de rajouter des objets dans le jeu. Il serait fastidieux de gérer tous les cas possibles afin de lancer toujours les mêmes répliques par défaut. Pour éviter cette corvée, cliquez sur le bouton New et ajoutez l'événement USE. Choisissez ensuite le premier élément de la liste (*) pour les deux paramètres.

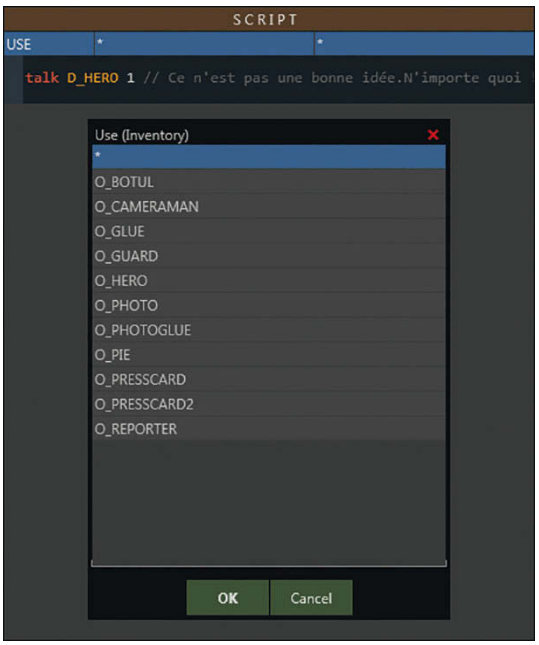


Figure 5

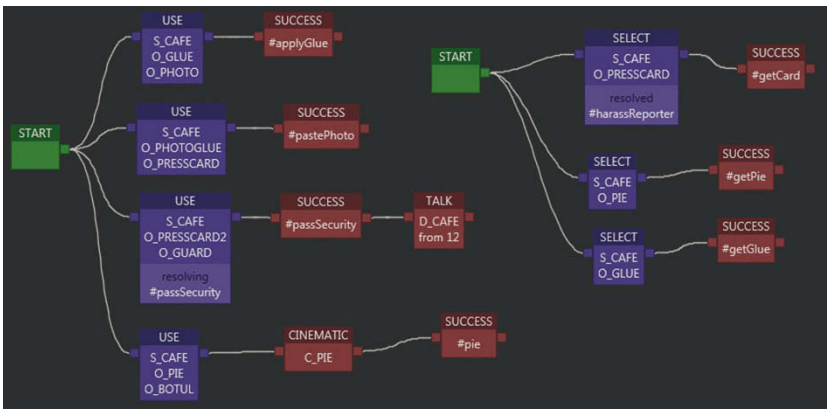


Figure 6

Figure 7

Puis dans le script, écrivez **talk D_HERO 1**. **Figure 5**
En utilisant les wildcards, Il est important de comprendre que cet événement ne sera pas appelé s'il existe un événement USE explicite ailleurs.

Les interactions via les rôles

Comme nous l'avons déjà évoqué et pour des raisons de clarté, nous allons réserver cet espace pour les actions qui ont un impact sur le déroulement de l'histoire. Ouvrez l'asset R_PIE. Bon à savoir : les boîtes START sont fusionnées à la création du jeu. **Figure 6**

Commençons par créer une boîte SELECT (l'équivalent de l'événement du même nom) et choisir la scène du jeu et l'objet O_PRESSCARD depuis les propriétés. Ajoutez une condition en spécifiant l'énigme #harassReporter à la ligne Resolved puzzle. Ainsi, l'événement ne sera appelé que si cette énigme a été résolue auparavant. Reliez l'entrée de cette boîte avec la sortie de la boîte START comme sur la capture d'écran. Ajoutez une boîte SUCCESS à la suite pour valider l'énigme #getCard. Faites de même pour la tarte et la colle.
Poursuivons avec la boîte USE en spécifiant les paramètres : S_CAFE, O_GLUE, O_PHOTO. En sortie, la boîte SUCCESS validera l'énigme #applyGlue. Faites de même pour la carte de presse.
La dernière énigme du jeu consiste à utiliser la tarte avec Botul pour l'entarter. Ajoutez alors une dernière boîte USE en spécifiant : S_CAFE, O_PIE et O_BOTUL. Pour jouer la cinématique d'entartage, ajoutez une boîte CINEMATIC en spécifiant l'asset C_PIE et terminez la branche avec une boîte SUCCESS de l'énigme #pie. Et voilà votre jeu est fini !

Les effets visuels

L'éditeur d'effets permet d'appliquer des effets visuels à l'écran en temps réel en sollicitant la carte graphique grâce aux shaders. Un effet peut être associé à une scène entière ou à un layer depuis l'éditeur de scènes ou à tout le jeu depuis les paramètres du projet. Un effet est constitué d'une liste de shaders parce qu'il est possible de les empiler pour créer une succession d'effets visuels. **Figure 7**

Créez un nouvel asset d'effet et ajoutez l'effet Curve à la liste. Ajustez ensuite les trois courbes pour modifier la luminosité, la saturation et la teinte du rendu à votre guise. Pour appliquer correctement l'effet à la scène, double-cliquez sur la propriété Mask du groupe Scène. Les fonds, les objets et les masques seront ainsi altérés par l'effet. Pour avoir un aperçu dans le logiciel, sélectionnez la scène du jeu dans le groupe Preview et jouez avec les différentes propriétés pour comparer le rendu. Lorsque vous êtes satisfait du résultat, il ne vous reste plus qu'à rajouter l'effet dans les propriétés du projet à la ligne Effects. Vous trouverez une propriété similaire mais prioritaire dans l'éditeur de scènes.

Conclusion

Ce tutoriel vous a donné une vue d'ensemble des principaux outils disponibles au sein du logiciel, n'hésitez pas à ajouter des énigmes pour approfondir vos connaissances et mieux comprendre les subtilités de **AGE**. Merci d'avoir suivi ce cours en espérant que cette première expérience vous a plu et vous donnera envie d'aller plus loin. Vous pouvez me joindre par email ou sur mon Discord si vous avez des questions, des remarques ou même des créations à partager.

Power Platform : de quoi parle-t-on ?

Antoine Galland
Service & Value design - Avanade

Passionné par l'apport de la technologie dans nos vies, je travaille depuis 7 ans dans la transformation numérique des entreprises. Alliant mon côté tech à un intérêt profond pour les métiers, je fais en sorte de proposer des alternatives numériques porteuses de sens et d'efficacité au monde de l'entreprise.

La Power Platform proposée par Microsoft est une boîte à outil, clé en main, permettant de matérialiser des solutions numériques dans l'environnement de travail de tout un chacun.

Cette boite à outil se compose de 4 briques, correspondant aux grandes composantes d'une solution applicative :

- Un moyen d'interaction à travers Power Apps pour la mise en place d'interfaces utilisateurs web ou mobile,
- Un moyen d'interaction avec un ChatBot via Power Virtual agent.
- La possibilité d'automatiser des processus grâce à travers Power Automate
- Un moyen d'analyse de données à travers Power BI

Figure 1

Vous l'aurez remarqué, la composante « Stockage de donnée » ne fait pas l'objet d'une brique à proprement parlé. Ceci est tout simplement dû au fait que la plateforme utilise des connecteurs afin de pouvoir exploiter un grand nombre de sources de données. Nous parlons ici de connecteurs vers des systèmes robustes type bases de données SQL, mais aussi plus inattendus, comme de simples fichiers Excel. Microsoft préconise l'utilisation de sa solution de stockage, le Dataverse, qui vient pleinement s'intégrer avec les possibilités de cette boîte à outil.
La plateforme permet de connecter une solution à de multiples sources de données à la fois, comprenant des données « On-Premise » par l'intermédiaire de passerelles (gateway).

Ces applications sont accessibles depuis un navigateur web (depuis le service Power Apps d'un compte Microsoft et sous forme d'IFrame sur n'importe quel site web après authentification) et depuis un périphérique mobile disposant de l'app Power Apps installée depuis le store. Sur les périphériques mobiles, il ne sera donc pas nécessaire de publier votre application dans un store, de passer les tests de sécurité et de payer le dit store. L'application Power Apps vous permet de naviguer et d'ouvrir les applications auxquelles vous avez accès avec votre compte.

Les types d'application

En termes d'application, Power Apps vous propose de créer des app de type « Zone de dessin » (Canvas) ou « Pilotés par un modèle » (Model Driven) ou des « Portails » (Portal).

Les app Canvas vous proposent une interface dans laquelle vous insérez des composants (des écrans, un bouton, une galerie) et venez configurer des propriétés en naviguant dans les menus de l'interface. Nous parlons ici de développement en mode WYSIWYG, très ludique pour débuter et donc accessible à tout un chacun. Voici une illustration de l'interface de conception d'application Canvas dans Power Apps :

Figure 2

L'autre type d'application, « Model Driven » se base obligatoirement sur de la donnée hébergée dans le Dataverse. Il s'agit de concevoir des applications dans une interface standardisée, composée de vues, formulaire et Tableaux de bords prêt à l'emploi. Nous viendrons ici définir l'expérience de navigation dans la donnée en ajoutant de la logique métier à l'aide de règles métier, processus métier, composants personnalisés.

Le dernier type d'application, les « Portails », permettent d'exposer de la donnée du Dataverse à des utilisateurs externes à votre organisation. Assez proche d'un éditeur de site web, vous pourrez permettre à des externes de visualiser vos données d'entreprise voir de leur proposer d'interagir avec.

Figure 1

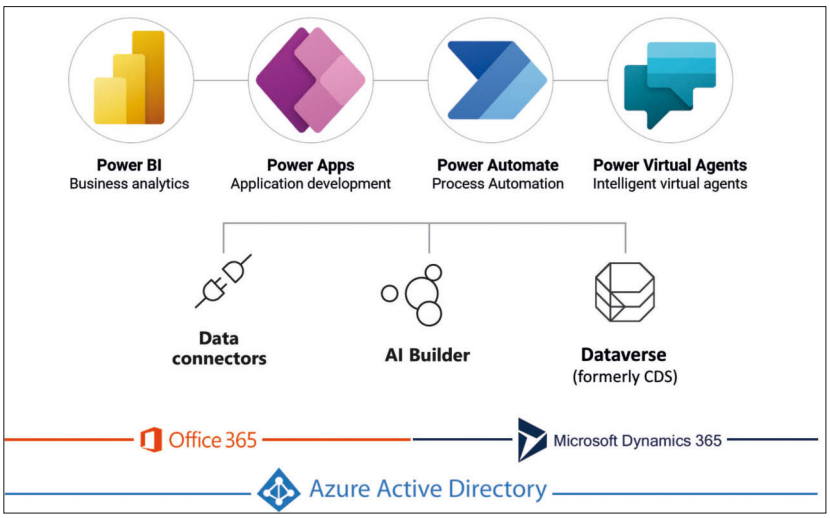
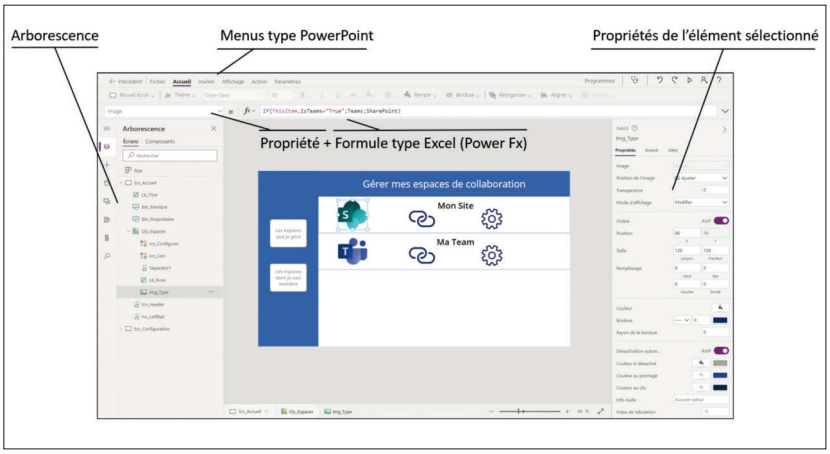


Figure 2



Il est important d'avoir en tête que la gestion des droits des sources de données est indépendante des droits de l'application. Concrètement, vous serez en mesure de concevoir une application et donc une expérience dont le contenu s'adaptera en fonction des droits utilisateurs sur les sources de données.

Développeur, en quoi je suis concerné ?

L'avènement de cette plateforme, comme outil légitime de développement, est une réponse naturelle à la multiplication des candidats à la digitalisation au regard des ressources disponibles et du délai attendu. Elle permet d'adresser le développement de solutions complexes et extensibles tout restant accessible à la conception de solutions très simples, directement par les utilisateurs finaux.

Le positionnement de la Power Platform dans l'environnement de travail des collaborateurs offre une possibilité d'expérimentation via des fonctionnalités standard pré-mâchées en constante évolution. Les interfaces de conception simplifiées réduisent drastiquement les compétences requises pour lancer un prototype, et accélèrent le temps nécessaire à la mise en production.

Cette plateforme représente un réel potentiel d'innovation, elle peut rapidement devenir un vivier à « bonnes idées » piloté par les métiers.

Le rôle de développeur s'inscrit dans la professionnalisation de ces « bonnes idées », en apportant expertise et robustesse aux solutions d'ampleur. Il s'agit ici de capitaliser sur la connaissance du besoin des métiers et la connaissance de la technique côté développeur.

La plateforme propose une grande variété de possibilités pour les développeurs, dans un rôle d'extensibilité, dès que l'on sort des fonctionnalités pré-mâchées proposées par les outils.
Vous aurez notamment la possibilité venir apporter à la plateforme des composants réutilisables à travers le « Power Apps Component Framework » ou concevoir des connecteurs personnalisés. L'intégration avec Azure permettra d'étendre les possibilités de ces solutions avec des fonctions Azure, la gestion d'identifiants de connexion avec Azure Key Vault, de la télémétrie avancée avec Azure App insights et de répondre ainsi aux besoins spécifiques d'une DSI.

Trop beau pour être vrai ?

Le réflexe naturel serait de trouver la plateforme trop contraignante car elle impose tout de même des principes de fonctionnement, certes complets, mais bien ancrés.

D'un autre côté, le fait de donner aux métiers la main pour concevoir des solutions, même simples, est un vrai sujet de gouvernance. En partant d'objectifs, d'une vision d'entreprise, il sera nécessaire de définir un certain nombre de règles, de renforcer la conformité et la sécurité du système d'information… Bref, il faudra planter un décor et l'adapter au fil du temps, sans quoi les bénéfices escomptés pourraient se transformer en véritable cauchemar.

Nous sommes ici dans du développement de solutions dans une infrastructure cloud, avec un effort de développement moindre que du développement traditionnel. En termes de personnalisation des interfaces, nous utilisons des contrôles qui sont mis à jour régulièrement. Les applications créées sont responsives nativement, l'intégration se fait au travers des connecteurs out-of-the-box. Pour ce qui est de l'extensibilité, vous capturez la logique métier sous forme d'API REST personnalisées et interagissez avec des pages web externes tout en conservant le contexte. Un apport de la plateforme s'axe autour de l'automatisation comme la capture de donnée, les notifications et l'approbation grâce à Power Automate. Pour ce qui est du reporting, là aussi Power BI viendra faciliter la digestion et visualisation de données.

Côté SLA, la disponibilité des services cloud Microsoft sont garanties à hauteur de 99,9%. Des spécificités en fonction des briques et fonctionnalités peuvent être consultés sur la documentation officielle : https://azure.microsoft.com/fr-fr/support/legal/sla/summary/

Testez par vous-même !

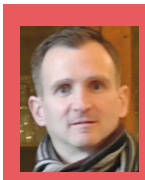
Envie de voir cela de plus près ? Sachez que vous pouvez tester un environnement de test pendant 30 jours gratuitement depuis le site Microsoft. Vous pourrez ainsi apprécier la simplicité des interfaces de conception, profiter d'un grand nombre d'exemples d'applications afin de les décortiquer et vous familiariser avec l'outil.

Pour ce faire rendez-vous sur **powerapps.microsoft.com** et créez un compte en sélectionnant « essayer gratuitement ». Une fois le compte créé, naviguez sur **make.powerapps.com**, sélectionnez créer dans la navigation de gauche puis choisissez le type d'application. Je vous conseille de partir sur une application Canvas dont l'interface est très intelligible. Vous pouvez rapidement importer de la donnée depuis un fichier Excel par exemple et insérer une galerie pour l'afficher dans l'application. Une fois l'interface prise en main, sauvegardez et publiez l'application. Elle sera alors instantanément disponible dans un navigateur web ainsi que depuis un périphérique mobile en installant l'application Power Apps depuis un store, en vous identifiant avec le compte créé précédemment. L'éditeur vous propose aussi des parcours d'apprentissage sur son site « Microsoft Learn » qui vous permettront de vous faire une idée globale de l'art du possible de cette plateforme Low-Code / No-Code.

Principales nouveautés détaillées de JavaScript de ES2015 (ES6) à ES2021 (ES12)

PARTIE 2.1 : LES NOUVEAUTÉS DE JAVASCRIPT 2016 ET 2017

Nous allons explorer au travers de cet article les principales évolutions du langage Java Script de 2015 à 2021 accompagnées d'exemples illustrant les nouveautés apportées par les différentes versions. JavaScript repose sur les standards d'ECMAScript qui lui se charge de définir un ensemble de normes concernant les langages de programmation de type script et standardisées par Ecma International dans le cadre de la spécification ECMA-262. Dans cette partie 2, nous allons explorer les nouveautés JavaScript 2016 et 2017.



Sylvain Cuenca

Je suis Architecte Système basé en région toulousaine et tout simplement passionné d'informatique depuis mon plus jeune âge, avec bientôt une vingtaine d'années d'expérience dans le monde du logiciel. Les thématiques autour de l'architecture logicielle en général, les performances, la fiabilité, la robustesse, la maintenabilité, l'amélioration continue et l'empreinte mémoire des solutions logicielles sont des sujets que j'aborde pleinement au quotidien. En parallèle, j'effectue beaucoup de veille technologique, m'occupe de la montée en compétence des équipiers et pratique les méthodes japonaises Kaizen sur l'amélioration continue autant sur le plan personnel que professionnel.

Principales nouveautés de la version ECMAScript 7 (ES7) ou JavaScript 2016

Ajout de la fonction "includes" sur les tableaux

Elle permet de vérifier si le tableau contient la valeur passée en argument. Elle retourne un booléen indiquant le résultat, c'est-à-dire « true » si le tableau contient la valeur, sinon « false ».

- **Syntaxe :**

  array.includes(itemToSearch)

  array.includes(itemToSearch, startIndex)

  Si l'indice de départ « startIndex » est strictement négatif, l'indice de départ effectif est la somme entre la taille du tableau et l'indice de départ « startIndex ». Si cette somme est toujours négative, le tableau est intégralement parcouru.

- **Exemples :**

Tests d'inclusion d'éléments existants dans un tableau avec et sans index de départ

```
const MY_NUMBERS = [1, 2, 3, NaN];

// Doit afficher true car 2 est inclus
// au niveau de l'index 1 (l'index d'un
// tableau démarre à 0)
console.log(MY_NUMBERS.includes(2));

// Doit afficher false car 2 est inclus
// au niveau de l'index 1 et on demande
//  à vérifier son existence à partir
// de l'index 2
console.log(MY_NUMBERS.includes(2, 2));

// Doit afficher true car NaN est inclus
// au niveau de l'index 3
console.log(MY_NUMBERS.includes(NaN));

// Doit afficher true car 1 est inclus
// au niveau de l'index 0 et on
// demande à vérifier son existence
// à partir de l'index -4 + 4(<=> taille du tableau) = 0
console.log(MY_NUMBERS.includes(1, -4));

// Doit afficher false car 1 est inclus
// au niveau de l'index 0 et on
// demande à vérifier son existence
// à partir de l'index -3 + 4(<=> taille du tableau) = 1
console.log(MY_NUMBERS.includes(1, -3));
```

Affichage dans la console :

```
true
false
true
true
false
```

Échec de la recherche d'un élément inexistant dans un tableau :

```
const ANIMALS = ['cat', 'dog', 'bat'];

// Doit afficher false car 'at' n'est pas
// inclus dans le tableau
console.log(ANIMALS.includes('at'));
```

Affichage dans la console :

```
false
```

Avis : *À utiliser en évitant de mettre un index de départ de recherche strictement négatif qui pourrait nuire à la compréhension du code.*

Simplification de l'exponentiation

ES7 ajoute un raccourci pour l'opérateur d'exponentiation : X ^ n s'écrit Math.pow(X, n) et aussi X ** n

- **Exemples :**

Calculs d'exponentiation avec le nouvel opérateur d'exponentiation « ** » et avec la fonction « Math.pow »

```
const BASE = 5;
const EXPONENT = 2;
```

```
const RESULT_WITH_EXPONENTIATION_OPERATOR = BASE ** EXPONENT;

console.log(`The result of ${BASE}^${EXPONENT}` +
  ` with exponentiation operator is: ${RESULT_WITH_EXPONENTIATION_OPERATOR}`);

const RESULT_WITH_MATH_POW_OPERATOR = Math.pow(BASE, EXPONENT);

console.log(`The result of ${BASE}^${EXPONENT}` +
  ` with \'Math.pow\' is: ${RESULT_WITH_MATH_POW_OPERATOR}`);
```

Affichage dans la console :

```
The result of 5^2 with exponentiation operator is: 25
The result of 5^2 with 'Math.pow' is: 25
```

Avis : *Les deux écritures (ancienne et nouvelle) conviennent bien à l'utilisation. C'est plus une question d'habitude d'utiliser la fonction statique « Math.pow » que l'opérateur d'exponentiation **, mais bon on peut s'habituer facilement à cette nouvelle écriture moins verbeuse.*

Principales nouveautés de la version ECMAScript 8 (ES8) ou JavaScript 2017

Définition de fonctions asynchrones

La déclaration « async function » définit une fonction asynchrone qui renvoie un objet « AsyncFunction ».

Pour rappel, une fonction asynchrone est une fonction qui s'exécute de façon asynchrone grâce à la boucle d'évènement en utilisant une promesse (Promise) comme valeur de retour. Cette promesse sera résolue avec la valeur renvoyée par la fonction asynchrone ou sera rompue s'il y a une exception non interceptée émise depuis la fonction asynchrone.

Une fonction asynchrone peut contenir une expression « await » qui met en stand-by l'exécution de la fonction asynchrone et attend la résolution de la promesse soumise à l'attente via « await ». Après cette attente de résolution, la fonction asynchrone reprend son exécution puis renvoie la valeur de résolution. Le mot-clé « await » est uniquement utilisé au sein de fonctions asynchrones. Si ce mot-clé est utilisé en dehors du corps d'une fonction asynchrone, cela provoquera une exception de type « SyntaxError ». Un cas d'usage du mot-clé « await » est pour séquencer des opérations asynchrones au sein d'une fonction asynchrone.

Les fonctions asynchrones permettent de définir des séquencements d'opérations de manière plus simple que par le biais de chaînage de promesses.

- **Syntaxes :**

  async function myFunction(param1, …, paramN) { … } ou
  const myFunction = async (param1, …, paramN) => { … } ;

- **Exemples :**

Création d'une fonction d'attente retournant une Promesse indiquant la fin d'attente seulement si tout s'est bien passé et de deux fonctions asynchrones qui attendent la résolution d'une promesse dans le corps de leurs fonctions

```
/**
* Allows to wait during a specific time defined in milliseconds.
*
* @param {*} timeToWaitInMS a time to wait in milliseconds.
*
* @returns a Promise.
*/
const wait = (timeToWaitInMS = 0) =>
  new Promise((resolve) => {
    setTimeout(() => {
      resolve(`Promise resolved after ${timeToWaitInMS}ms`);
    }, timeToWaitInMS);
  });

/**
* Asynchronous classic function allowing to wait a specific time.
* @param {*} timeToWaitInMS a time to wait in millseconds.
* @returns a Promise
*/
async function asyncWait(timeToWaitInMS) {
  // Ici on attend que la promesse d'attente soit résolue
  // via le mot-clé 'await' avant de retourner son résultat
  const result = await wait(timeToWaitInMS);

  return `Result for classic asynchronous waiting function: ${result}`;
}

/**
* Asynchronous Arrow function allowing to wait a specific time in milliseconds.
* @param {*} timeToWaitInMS a time to wait in milliseconds.
* @returns a Promise
*/
const asyncWaitWithArrowFunction = async (timeToWaitInMS) => {
  // Ici on attend que la promesse d'attente soit résolue
  // via le mot-clé "await" avant de retourner son résultat
  const result = await wait(timeToWaitInMS);

  return `Result for asynchronous waiting arrow function: ${result}`;
};

// Lancement d'une fonction asynchrone classique d'attente
// pour une durée de 10 secondes et affichage du résultat dans la console
asyncWait(10000).
  then((result) => console.log(result))
  .catch((error) => console.error(error));

// Lancement d'une fonction fléchée asynchrone d'attente
// pour une durée de 1 seconde et affichage du résultat dans la console
asyncWaitWithArrowFunction(1000).
  then((result) => console.log(result))
  .catch((error) => console.error(error));
```

Affichage dans la console :

```
Result for asynchronous waiting arrow function: Promise resolved after 1000ms
Result for classic asynchronous waiting function: Promise resolved after 10000ms
```

Dans l'exemple précédent, on peut s'apercevoir que l'exécution de la fonction fléchée asynchrone « asyncWaitWithArrow Function » se termine avant celle de la fonction classique « asyncWait », car le délai d'attente est plus court (on a paramétré 1 seconde d'attente pour la fonction « asyncWaitWith ArrowFunction » contre 10 secondes d'attente pour la fonction « asyncWait ».

Séquencement d'opérations asynchrones en chaînant les promesses

```
/**
 * Allows to wait during a specific time defined in milliseconds.
 *
 * @param {*} timeToWaitInMS a time to wait in milliseconds.
 *
 * @returns a Promise.
 */
const wait = (timeToWaitInMS = 0) =>
  new Promise((resolve) => {
    setTimeout(() => {
      resolve(`Promise resolved after ${timeToWaitInMS}ms`);
    }, timeToWaitInMS);
  });

/**
* Allows to wait sequentially many times
* by chaining Promises.
*/
const executeWaitingPromisesByChainingThem = () => wait(1000)
  .then(result => {
    console.log(result);

    return wait(3000)
  })
  .then((result) =>
    console.log(result))
  .catch((error) => console.error(error));

executeWaitingPromisesByChainingThem();
```

Affichage dans la console après une seconde d'attente :

```
Promise resolved after 1000ms
```

Affichage dans la console après 3 secondes supplémentaires :

```
Promise resolved after 3000ms
```

Plus on chaîne des promesses, plus il devient difficile d'avoir une vue globale de tout le séquencement.

Séquencement d'opérations asynchrones via « await » en conservant le séquencement défini dans le précédent exemple pour comparaison

```
/**
 * Allows to wait during a specific time defined in milliseconds.
 *
 * @param {*} timeToWaitInMS a time to wait in milliseconds.
 *
 * @returns a Promise.
 */
const wait = (timeToWaitInMS = 0) =>
  new Promise((resolve) => {
    setTimeout(() => {
      resolve(`Promise resolved after ${timeToWaitInMS}ms`);
    }, timeToWaitInMS);
  });

/**
* Allows to wait sequentially many times
* by not chaining Promises.
*/
const executeWaitingSequence = async () => {
  try {
    // Ici on attend durant 1 seconde que la promesse
    // d'attente soit résolue via le mot-clé "await"
    // avant de retourner son résultat
    const waitingOneSecondResult = await wait(1000);
    console.log(waitingOneSecondResult);

    // Ici on attend durant 3 secondes que la promesse
    // d'attente soit résolue via le mot-clé "await"
    // avant de retourner son résultat
    const waitingThreeSecondsResult = await wait(3000);
    console.log(waitingThreeSecondsResult);
  }
  catch (error) {
    console.error(error);
  }
};

executeWaitingSequence();
```

Affichage dans la console après une seconde d'attente :

```
Promise resolved after 1000ms
```

Affichage dans la console après 3 secondes supplémentaires :

```
Promise resolved after 3000ms
```

Avis : *Cette manière de séquencer les opérations apporte plus de visibilité sur l'ensemble des opérations séquencées via le mot-clé « await ». Les fonctions asynchrones sont très utiles lorsqu'on souhaite paralléliser des tâches au sein d'un traitement. À utiliser à bon escient tout de même et dès qu'on a un réel besoin.*

Ajout de la fonction Object#entries(myObject)

La fonction « Object.entries(myObject) » renvoie un tableau des propriétés propres énumérables d'un objet dont chaque élément du tableau est représenté par un tableau à deux éléments sous la forme de couples [clé, valeur], dans le même ordre qu'une boucle « for...in ».

L'ordre du tableau renvoyé par cette fonction ne dépend pas de la façon dont l'objet est défini. Afin de conserver un certain ordre, on pourra appeler la fonction « Array.sort() » qui permettra de trier comme on le souhaite les éléments du tableau.
Si l'argument « myObject » représente un type primitif, il sera converti en un objet avant l'opération se chargeant de lister les propriétés énumérables. Dans le cas où le type primitif n'a pas de propriété, par exemple un entier, le tableau retourné par la fonction sera vide.

- **Syntaxe :**
  Object#entries(myObject) qui retourne un tableau composé de tableaux à deux éléments sous forme de [Clé, Valeur]

- Le paramètre « myObject » définit l'objet dont on souhaite connaître les propriétés propres énumérables

• **Exemples :**

Affichage des propriétés d'un objet comportant seulement des propriétés énumérables :

```
const CAR = {
  brand: 'Toyota',
  model: 'Yaris',
  color: 'blue',
  year: 2021
};

console.log('The entries for CAR are:', Object.entries(CAR));
```

Affichage dans la console :

```
The entries for CAR are: [
 [ 'brand', 'Toyota' ],
 [ 'model', 'Yaris' ],
 [ 'color', 'blue' ],
 [ 'year', 2021 ]
]
```

Affichage des propriétés d'un objet comportant à la fois des propriétés énumérables et non énumérables :

```
const CAR = {
  brand: 'Toyota',
  model: 'Yaris',
  color: 'blue',
  year: 2021
};

Object.defineProperty(CAR, 'wheelNumber', { value: 4, enumerable: false });

console.log('Wheel number:', CAR.wheelNumber);

console.log('The entries for CAR are:', Object.entries(CAR));
```

Affichage dans la console :

```
Wheel number: 4
The entries for CAR are: [
 [ 'brand', 'Toyota' ],
 [ 'model', 'Yaris' ],
 [ 'color', 'blue' ],
 [ 'year', 2021 ]
]
```

Ici l'objet « CAR » comporte quatre propriétés énumérables : « brand », « model », « color » et « year » et une propriété non énumérable « wheelNumber ». Au travers de cet exemple, on s'aperçoit que la propriété non énumérable « wheelNumber » n'est pas listée lorsqu'on liste les entrées de l'objet « CAR ».

Définition d'un objet semblable à un tableau, c-à-d. dont les propriétés sont représentées sous forme d'index (volontairement, les propriétés de l'objet sont définies de manière désordonnée au niveau des index)

```
// Définition d'un objet semblable à un tableau
// c'est-à-dire dont ses propriétés sont
// représentées sous forme d'index
const MY_CAR = {
  2: 'Daihatsu',
  1: 'Toyota',
  0: 'Lexus',
  3: 'Scion'
}

console.log('The entries for MY_CAR are:', Object.entries(MY_CAR));
```

Affichage dans la console :

```
The entries for MY_CAR are: [
 [ '0', 'Lexus' ],
 [ '1', 'Toyota' ],
 [ '2', 'Daihatsu' ],
 [ '3', 'Scion' ]
]
```

On s'aperçoit que la fonction « Object#entries » a appliqué un tri par ordre croissant sur les propriétés définies dans l'objet « MY_CAR ».

Un argument de type primitif, ayant des propriétés, passé à la fonction « Object#entries » sera converti en un objet avec des propriétés, avant extraction des propriétés énumérables

```
// Un argument de type primitif ayant
// des propriétés sera converti en un
// objet via la fonction "Object#entries"
console.log(Object.entries('String example'));
```

Affichage dans la console :

```
[
 [ '0', 'S' ], [ '1', 't' ],
 [ '2', 'r' ], [ '3', 'i' ],
 [ '4', 'n' ], [ '5', 'g' ],
 [ '6', ' ' ], [ '7', 'e' ],
 [ '8', 'x' ], [ '9', 'a' ],
 [ '10', 'm' ], [ '11', 'p' ],
 [ '12', 'l' ], [ '13', 'e' ]
]
```

Un argument de type primitif, n'ayant pas de propriété, passé à la fonction « Object#entries » sera converti en un objet vide, avant extraction des propriétés énumérables

```
// Un argument de type primitif n'ayant
// pas de propriété sera converti en un
// objet vide via la fonction "Object#entries"
console.log(Object.entries(1000));
```

Affichage dans la console :

```
[]
```

Extraction des propriétés d'un objet vide

```
console.log(Object.entries({}));
```

Affichage dans la console :

```
[]
```

Extraction des propriétés d'un tableau non vide

```
console.log(Object.entries(['Toyota', 'Lexus', 'Daihatsu', 'Scion']));
```

Affichage dans la console :

```
[
 [ '0', 'Toyota' ],
 [ '1', 'Lexus' ],
 [ '2', 'Daihatsu' ],
 [ '3', 'Scion' ]
]
```

Extraction des propriétés d'un tableau vide

```
console.log(Object.entries([]));
```

Affichage dans la console :

```
[]
```

Parcours des couples de propriétés/valeur d'un objet

```
const MY_CAR = {
  brand: 'Acura',
  model: 'NSX',
  color: 'yellow',
  year: 1992
};

// Parcours des couples de propriété/valeur
// d'un objet via l'opérateur for..of
console.log('Property/Value pairs from MY_CAR object',
  'browsed by \'for..of\' operator are :');
for (const [property, value] of Object.entries(MY_CAR)) {
  console.log(`${property}: ${value}`);
}

// Parcours des couples de propriété/valeur
// d'un objet via l'opérateur forEach
console.log('Property/Value pairs from MY_CAR object',
  'browsed by \'forEach\' operator are :');
Object.entries(MY_CAR).forEach(
  ([property, value]) => console.log(`${property}: ${value}`)
);
```

Affichage dans la console :

```
Property/Value pairs from MY_CAR object browsed by 'for..of' operator are :
brand: Acura
model: NSX
color: yellow
year: 1992
Property/Value pairs from MY_CAR object browsed by 'forEach' operator are :
brand: Acura
model: NSX
color: yellow
year: 1992
```

Avis : *Très pratique lorsqu'on souhaite récupérer les propriétés d'un objet sous forme de couple clé/valeur.*

Ajout de la possibilité de convertir un objet en passant par « Map »

Pour rappel, l'objet « Map » définit un dictionnaire composé de clés/valeurs. Les clés et les valeurs peuvent être de n'importe quel type (objets, types primitifs).

Lorsqu'on ajoute un couple clé/valeur dans une Map, l'ordre d'insertion est conservé et ainsi le parcours de la Map respecte cet ordre.

On a désormais possibilité de convertir simplement un objet en un objet « Map » via la fonction « Object#entries ».

- **Rappel de la syntaxe d'une Map** :
  new Map([iterable])
  - Le paramètre « iterable » définit un tableau ou tout autre objet itérable dont ses éléments sont des couples de clé/valeur et cela va nous intéresser pour créer une « Map » à partir du résultat de la fonction « Object#entries » appliquée à un objet.

- **Exemples :**

Transformation d'un objet sous forme d'une map

```
const MY_CAR = {
  brand: 'Acura',
  model: 'NSX',
  color: 'yellow',
  year: 1992
};

// Conversion de l'objet MY_CAR en Map
const MY_CAR_MAP = new Map(Object.entries(MY_CAR));

// Affichage de la map
console.log(MY_CAR_MAP, '\r\n');

// Affichage des couples clé/valeur
// de la map
console.log('Map content:');
for (const [key, value] of MY_CAR_MAP.entries()) {
  console.log(`${key}: ${value}`);
}
```

Affichage dans la console :

```
Map(4) {
 'brand' => 'Acura',
 'model' => 'NSX',
 'color' => 'yellow',
 'year' => 1992
}

Map content:
brand: Acura
model: NSX
color: yellow
year: 1992
```

Avis : *Combinée avec la fonction « Object.entries », on peut aisément transformer un Objet en « Map ». À utiliser sans modération dès que le besoin s'en fait ressentir.*

La suite de cet article dans PROGRAMMEZ! 254.

Automatiser ses Velux avec .NET nanoFramework et Home Assistant

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

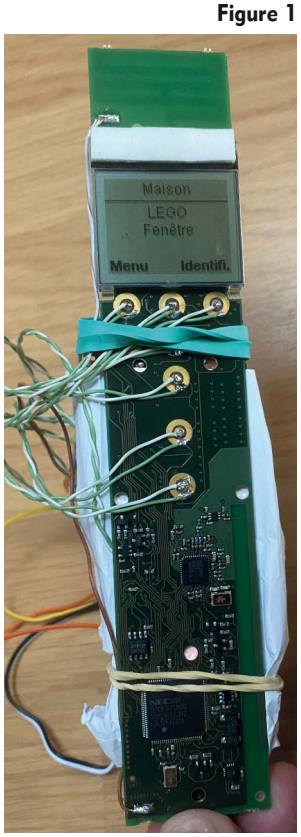
Voilà plusieurs années que j'utilise Home Assistant (https://www.home-assistant.io/) pour intégrer et automatiser des tâches chez moi. Un des avantages est le tableau de bord de Home Assistant qui permet de visualiser à peu près n'importe quoi de façon graphique, soit sous forme de graph, soit sous forme de bouton, d'icônes ou de visuels divers.

J'ai également un serveur MQTT (https://mqtt.org). MQTT est un standard fonctionnant comme un bus de message où l'on souscrit (sub) et on publie (pub) des messages. Le protocole est léger et évite toute la complexité d'autres protocoles tels que HTTP. Il y a également une notion de qualité de service permettant de choisir comment publier ses messages, avec ou sans garantie de réception, avec ou sans réessayer. Le tout de façon sécurisée en pouvant utiliser des messages cryptés en TLS (Transport Layer Security). Cela en fait un élément très intéressant pour faire communiquer des objets dans le monde de l'Internet of Things (IoT). J'utilise Mosquitto (https://www.mosquitto.org/) qui fonctionne parfaitement bien et offre une grande flexibilité.

Je possède 4 Velux et je voulais les intégrer dans mon Home Assistant. Ces Velux sont pilotables avec une télécommande. Je me suis donc dit qu'il y aurait un moyen simple de les intégrer. En cherchant un peu, je me suis aperçu qu'il fallait acheter une box spécifique coûtant environ 200€ et que son intégration était tout sauf simple. Velux utilise en effet un système propriétaire de communication avec ses éléments depuis les télécommandes et les éléments entre eux. La partie de cryptage est intégrée directement dans les chipsets ce qui rend la tâche de reverse engineering assez compliquée.

Je me suis dit qu'une bonne idée serait certainement de réutiliser une télécommande et de la hacker. Mon idée est simple : piloter la télécommande en simulant l'appui sur les touches pour ouvrir et fermer mes Velux. Et pour appuyer sur les boutons, un peu d'électronique suffit associée à un ESP32. Voilà à quoi ressemble le résultat final : **Figure 1**

Figure 1



Nous allons bien sûr détailler tout cela et voir comment réaliser tout cela étape par étape.

ESP32 et .NET nanoFramework

Tout d'abord, le choix d'un ESP32 est assez simple : tous les ESP32 supportent .NET nanoFramework (https://docs.nanoframework.net/). .NET nanoFramework est une implémentation open source de .NET sur des micro contrôleurs (Micro Controller Unit – MCU). Cela permet d'avoir le support de C#, d'utiliser Visual Studio, de pouvoir mettre des points d'arrêts dans son code, de pouvoir débugger ce type de MCU avec de vrais outils et de façon efficace.

.NET nanoFramework n'est pas compatible avec tout le framework .NET. C'est un sous-ensemble avec des limitations tel que le non-support des génériques, pas de linq. Mais il y a une certaine compatibilité de code, qui permet de réutiliser du code notamment à travers des projets partagés. .NET nanoFramework possède tout ce qu'il faut pour aller très loin tel que la réflexion, des classes pour le réseau, se connecter au wifi, des classes de haut niveau pour utiliser Azure IoT et même AWS IoT Core. Bien entendu il y a aussi un accès facile aux GPIO, I2C, SPI, PWM et plus généralement les autres protocoles hardwares. À noter également de très nombreux drivers pour des capteurs et autres périphériques, Le tout aligné avec .NET IoT (https://github.com/ dotnet/iot/) permettant là aussi une réutilisation de code. Tout est disponible à travers des nugets comme dans n'importe quel projet .NET plus classique.

Ce dont nous aurons besoin c'est de l'accès aux GPIO. Nous utiliserons pour piloter les boutons de la télécommande. Nous utiliserons aussi les capacités de connexion au wifi pour s'interfacer avec le serveur MQTT. Et en bonus, nous ajoutons un capteur d'humidité et de température, un AM2320 connecté en I2C. Ce dernier me permettra d'ajouter des scénarios avancés dans mon Home Assistant comme l'ouverture et la fermeture automatique des Velux en fonction de la température intérieure et extérieure.

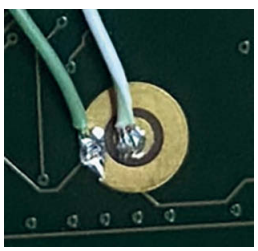
Hardware hack d'une télécommande

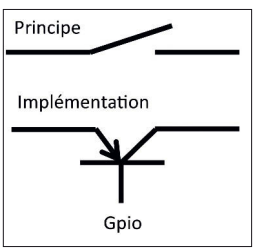
La télécommande originale est celle-ci : **Figure 2**

Pour pouvoir piloter l'appui sur les boutons, il faut d'abord démonter toute la télécommande, enlever les éléments en plastique, les connecteurs de batterie. Suivant les télécommandes, cette opération peut être plus ou moins délicate. Si comme ici vous avez un écran, faites bien attention de ne pas l'abîmer, les connectiques sont souvent assez fragiles.

Le principe d'un bouton de télécommande est assez simple : vous avez une partie déformable qui contient un élément conducteur (souvent du graphite) qui vient faire contact entre 2 portions de circuit imprimé.

Pour piloter cela, nous utiliserons un transistor qui sera piloté par un pin de l'ESP32. Dans le cas de ma télécommande, les contacts sont assez simples et il est assez facile de souder des fils dessus :





Cela peut être un peu plus compliqué dans certains cas. Il est

Figure 2

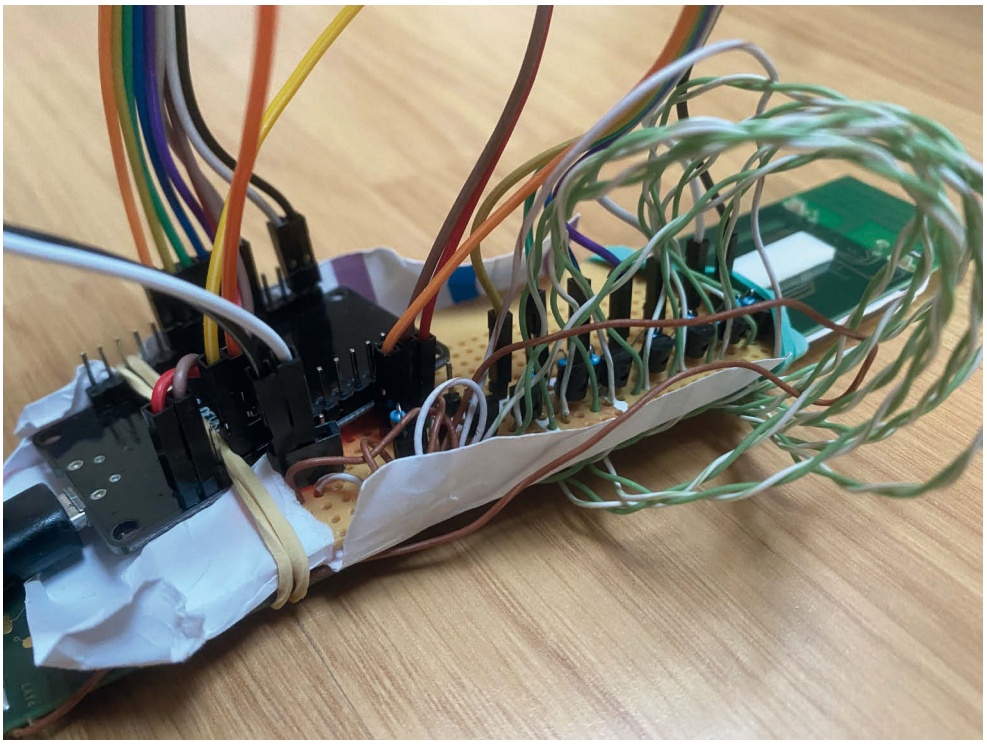


Figure 3

nécessaire d'utiliser un transistor que l'on utilisera en mode saturé de façon à créer le contact et fermer le circuit électrique. Vu les voltages impliqués, de simples transistors font parfaitement bien l'affaire. Attention à bien vérifier où se trouve la tension positive et négative pour bien positionner le transistor. On ajoutera entre le pin de l'ESP32 et le collecteur du transistor une petite résistance pour limiter le courant de sortie. On utilisera le même principe pour allumer la télécommande. Ici, nous avons de la chance, le voltage d'alimentation est de 5 volts. Nous prendrons donc l'alimentation USB pour la télécommande comme pour l'ESP32.
Et comme vous pouvez le voir sur la photo, le plus compliqué, c'est de faire les soudures proprement sur la télécommande et aussi le peu d'électronique qu'il y a. **Figure 3**

Pilotage des pins avec .NET nanoFramework

Comme indiqué, nous allons piloter les boutons. Pour cela, il faudra effectuer plusieurs tâches. La première, ouvrir le pin, la seconde changer l'état du pin. Tout cela se fait extrêmement simplement en .NET nanoFramework :

```
_controller = new GpioController();
_controller.OpenPin(ButtonMiddleDown, PinMode.Output);
_controller.Write(ButtonMiddleDown, PinValue.Low);
```

La première ligne permet de créer un contrôleur. Un seul suffit pour tout le projet. Ensuite, on ouvre le pin. Et il est possible d'écrire, donc de changer la valeur du pin. Ici, low qui représente le niveau bas, donc l'état 0. Ce pin est connecté à un bouton de la télécommande. Il est connecté au transistor, nous aurons donc un transistor fermé, le courant ne passera pas. *ButtonMiddleDown* est une constante qui contient le numéro du pin. On parle de numéro de GPIO, dans mon cas le 19. Attention lors des branchements, ce qui compte c'est bien le numéro de GPIO, pas le numéro physique de la broche. C'est quelque chose à quoi il faut faire attention et qui porte souvent à confusion chez les débutants.
Tous les pins de toute la télécommande sont initialisés de la même façon. On effectue ensuite un reset de la télécommande, cela permet de la mettre dans un état connu.

```
_controller.Write(ButtonReset, PinValue.Low);
Thread.Sleep(500);
_controller.Write(ButtonReset, PinValue.High);
```

Un transistor gère également l'alimentation. Le niveau bas ne permet pas le passage du courant dans le transistor, la télécommande sera donc arrêtée. On attend alors 500 ms qui semblent suffisantes pour la réinitialiser. Et on remet l'état haut du transistor, la télécommande s'allume alors et arrive dans un état connu.

Wifi, et connexion MQTT

Avec .NET nanoFramework et les MCU qui le supportent tels que les ESP32, il est possible de se connecter en Wifi ou Ethernet. Il y a des helpers qui facilitent la vie. Par exemple, voici le modèle pour se connecter en DHCP quand vous stocker les éléments de votre wifi dans votre code :

```
// Nous allons utiliser TLS, nous devons avoir une date & heure valides
// Nous allons laisser 1 minute pour se connecter ce qui est largement suffisant
var success = WiFiNetworkHelper.ConnectDhcp("Ssid", "Password", requiresDateTime :
 true, token: new CancellationTokenSource(60000).Token);
if (!success)
{
  if (WiFiNetworkHelper.HelperException != null)
  {
    Debug.WriteLine($"{WiFiNetworkHelper.HelperException.Message}");
  }
```

```
}
Debug.WriteLine($"Date and time is now {DateTime.UtcNow}");
```

Le helper permet juste de donner ses éléments de connexion et offre des options telles qu'être sûr d'avoir une date et heure valide. C'est important dans le cas où vous vous connectez en SSL/TLS sinon vos certificats ne seront pas validés correctement.
Il y a également un client MQTT qui s'installe lui aussi à travers un nuget comme tous les éléments de .NET nanoFramework. Le support de la librairie permet d'utiliser MQTT 3.1, 3.1.1 et 5.0. L'ensemble des scénarios sont supportés, connexion avec ou sans certificats. Le tout permettant des scénarios avancés.
Pour faire simple, nous allons ici créer une connexion à un serveur local, sans nom d'utilisateur ni mot de passe et sans utiliser TLS. Dans la vraie vie, vous prendrez soin de sécuriser cette connexion avec TLS (et donc un certificat) d'une part et un nom d'utilisateur et mot de passe (ou clé privée) d'autre part. Le tout étant bien sûr supporté en .NET nanoFramework.

```
_mqtt = new("192.168.1.2", 1883, false, null, null, MqttSslProtocols.None);
TryReconnectMqtt();
_mqtt.ConnectionClosed += MqttConnectionClosed;
_mqtt.MqttMsgPublishReceived += MqttMsgPublishReceived;

private static void TryReconnectMqtt()
{
  // Try to reconnect right away
  MqttReasonCode ret;
  int retry = 0;
Reconnect:
  ret = _mqtt.Connect(DeviceName);
  if (ret != MqttReasonCode.Success)
  {
    // Give it quite some retry
    if (retry++ > 10)
    {
      SetupDeepSleepAndRetry();
    }

    // Wait 30 seconds
    Thread.Sleep(30000);
    goto Reconnect;
  }

  SubscribeTopics();
}

private const string TopicSubscribe = "velux/action";

private static void SubscribeTopics()
{
  _mqtt.Subscribe(new string[] {
    $"{TopicSubscribe}0",
    $"{TopicSubscribe}1",
    $"{TopicSubscribe}2",
    $"{TopicSubscribe}3",
    $"{TopicSubscribe}4",
  }, new MqttQoSLevel[] {
    MqttQoSLevel.AtMostOnce,
    MqttQoSLevel.AtMostOnce,
    MqttQoSLevel.AtMostOnce,
    MqttQoSLevel.AtMostOnce,
    MqttQoSLevel.AtMostOnce,
  });
}
```

Une fois le client MQTT créé, nous allons essayer de nous connecter. Ici, j'essaie de me connecter plusieurs fois en cas de problème, j'attends 30 seconds et réessaie. Si vraiment il y a un gros problème, le MCU ira en sommeil profond pour quelques secondes. Cela aura le bienfait de l'équivalent d'un reboot.

```
private static void SetupDeepSleepAndRetry()
{
  // Sleep for 2 seconds and retry
  Sleep.EnableWakeupByTimer(TimeSpan.FromSeconds(2));
  Sleep.StartDeepSleep();
}
```

L'avantage des MCU est cette capacité de pouvoir dormir pour diminuer leur consommation quand ils ne font rien et de pouvoir se réveiller. Dans mon scénario, j'ai besoin qu'il soit tout le temps connecté.
La dernière partie du code est la souscription aux messages MQTT. Je souscris donc à tous les messages pour velux/action finissant par 0 à 4 représentants chacun un velux, le numéro 4 représentant l'ensemble des Velux. Nous verrons cela dans les scénarios un peu plus loin.
À noter que je souscris également à deux événements. Le premier concerne la fermeture de la connexion. Dans ce cas, j'essaierai de me reconnecter comme expliqué précédemment. La connexion dans un réseau local peut être perdue si le serveur MQTT reboote ou a un problème, si votre connexion wifi a un problème. Réessayer de se connecter règle en général le problème rapidement.

```
private const string MessageMqttOpen = "ON";
private const string MessageMqttClosed = "OFF";

private static void MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
{
  string message = Encoding.UTF8.GetString(e.Message, 0, e.Message.Length);

  int number = e.Topic[TopicSubscribe.Length] - '0';
  if ((number < 0) || (number > NumberOfWindows))
  {
    Debug.WriteLine($"Windows number not correct: {number}");
    return;
  }

  Debug.WriteLine($"Message from {number}: {message}");
  if (message == MessageMqttOpen)
  {
    if (!LastAction[number])
    {
      while (_isBuzy)
```

```
        {
            Thread.Sleep(1000);
        }

        _isBuzy = true;
        OpenWindow(number);
    }
  }
  else
  {
    if (LastAction[number])
    {
        while (_isBuzy)
        {
            Thread.Sleep(1000);
        }

        _isBuzy = true;
        CloseWindow(number);
    }
  }
}
```

La fonction permettant de gérer les messages reçus est relativement simple. D'abord on extrait le numéro de la fenêtre. Comme nous n'avons souscrit qu'à ces messages, nous sommes sûrs que le dernier élément du sujet est le numéro de la fenêtre.
Ensuite, nous regardons si l'ordre est d'ouvrir ou de fermer. Comme les actions peuvent être envoyées les unes après les autres, mais que la télécommande le permet que de piloter 1 fenêtre à la fois, il y a l'introduction de *_isBuzy* qui permettra d'attendre que les actions précédentes soient effectuées.
Il reste encore une action autour de MQTT que je n'ai pas encore présentée : envoyer une information, que l'on appelle publication.

```
private const string VeluxState = "velux/state";
private const string TopicVelux = "velux/percent";

private static void PublishState(int number, int state)
{
  if ((_mqtt == null) || !_mqtt.IsConnected)
  {
    return;
  }

  string topic = $"{TopicVelux}{number}";
  _mqtt.Publish(topic, Encoding.UTF8.GetBytes($"{state}"));
  topic = $"{VeluxState}{number}";
  if (state > 10)
  {
    _mqtt.Publish(topic, Encoding.UTF8.GetBytes(MessageMqttOpen));
  }
  else
  {
    _mqtt.Publish(topic, Encoding.UTF8.GetBytes(MessageMqttClosed));
  }
}
```

Ici, je vais en envoyer plusieurs, une pour reporter le statut et l'autre le pourcentage d'ouverture. Je peux en effet choisir le pourcentage d'ouverture. Et je considère que si la fenêtre est ouverte à plus de 10%, elle est dans l'état ouverte.
Cette fonction sera appelée dans les fonctions permettant d'ouvrir et de fermer les fenêtres.

Piloter la télécommande

On a vu comment ouvrir et changer le statut d'un pin. Maintenant, pour piloter la télécommande il va falloir enchaîner ces actions. Ce qui est relativement compliqué c'est que l'on va imiter un humain qui appuie sur la télécommande. Il va donc falloir mesurer le temps que prennent les différentes actions de navigation dans les menus de la télécommande. Cliquer va prendre quelques millisecondes, il va falloir attendre qu'une nouvelle fenêtre s'affiche, cela va prendre aussi un peu de temps. Ce qui est compliqué dans cette démarche c'est surtout qu'il va falloir trouver des valeurs qui fonctionnent à tous les coups. Et aussi optimiser un peu le temps. On peut bien sûr attendre 1 seconde entre chaque appui de touche, mais s'il faut 20 appuis pour ouvrir une fenêtre, on préférera optimiser pour trouver une valeur plus faible qui fonctionne quand même à chaque fois. Ici, pas de miracle, le test manuel est nécessaire. Cela revient à simuler des clicks avec du vrai code et mesurer ce qui se passe, vérifier que tout sera ok une fois plein d'actions effectuées.
Autre élément sur ma télécommande, je peux désactiver la lumière de fond et faire en sorte que l'écran ne se mette jamais en veille.
J'ai donc créé des fonctions de base permettant de naviguer. Par exemple celle qui me permet de sélectionner une fenêtre :

```
private const int ClickLength = 300;

private static void ButtonSelectWindow(bool down)
{
  if (down)
  {
    _controller.Write(ButtonUpMiddleDown, PinValue.High);
    Thread.Sleep(ClickLength);
    _controller.Write(ButtonUpMiddleDown, PinValue.Low);
  }
  else
  {
    _controller.Write(ButtonUpMiddleUp, PinValue.High);
    Thread.Sleep(ClickLength);
    _controller.Write(ButtonUpMiddleUp, PinValue.Low);
  }
}
```

De façon assez simple, comme pour la remise à zéro, on change le statut. Ici l'état initial est bas, le mettre à haut permet d'alimenter le transistor qui simule l'appui puis le remettre à bas coupe l'alimentation du transistor et donc de l'appui. Une valeur de 300 ms fonctionnait parfaitement pour moi. Un des boutons permet de naviguer en bas dans les menus, l'autre en haut. C'est ce qui est fait ici.
Pour ouvrir une fenêtre, voici la fonction :

```
private const int TimeToProcessOperation = 28000;

private static void OpenWindow(int number)
{
  Debug.WriteLine($"Opening velux {number}");
  LastAction[number] = true;
  if (number == AllWindows)
  {
    for (int i = 0; i < NumberOfWindows; i++)
    {
      LastAction[i] = true;
    }
  }
  SelectWindows(number);
  _controller.Write(ButtonMiddleUp, PinValue.High);
  Thread.Sleep(ClickLength);
  _controller.Write(ButtonMiddleUp, PinValue.Low);
  PublishState(number, 100);
  Thread.Sleep(TimeToProcessOperation);
  _isBuzy = false;
}
```

Le tableau contenant le statut des fenêtres est mis à jour. Une des options de la télécommande permet d'ouvrir toutes les fenêtres à la fois. La sélection de la fenêtre est faite, ensuite on simule le clic sur le bouton permettant d'ouvrir la fenêtre.

Une fois cela effectué, on met à jour les informations en publiant les statuts dans le serveur MQTT comme expliqué plus haut. À noter que l'ouverture complète d'une fenêtre prend 28 secondes ! On bloque donc les opérations de la télécommande pendant ce temps. La télécommande étant en contact avec la fenêtre pendant tout le temps. Elle ne rend pas la possibilité d'être piloté à nouveau avant que l'opération soit terminée ou annulée. Dans cette version, je ne donne pas cette possibilité qui permettrait de piloter plus finement le pourcentage. C'est quelque chose d'assez simple à faire, mais je me suis rendu compte que je les ouvrais ou les fermais simplement.

La solution est disponible avec l'ensemble des éléments sur mon GitHub. Mais tous les points importants ont été discutés ici. La question que vous pouvez me poser est : mais que se passe-t-il s'il pleut ? Et bien la bonne nouvelle c'est que les fenêtres vont se fermer de façon automatique. Cependant, je n'ai pas de moyen de le savoir. Il me faudra donc prendre cela en compte dans mon Home Assistant.

Home Assistant : intégrons notre télécommande

Home Assistant (https://www.home-assistant.io/) est une plateforme open source de domotique qui est basé sur des intégrations (plug-ins). Home Assistant offre la possibilité de créer très facilement des tableaux de bord et d'intégrer à peu près tout ce que vous avez la maison, même le fait maison ! Il y a des milliers d'intégrations disponibles, souvent communautaires qui permettent même d'intégrer des éléments très franco-français comme votre consommation de gaz ou votre consommation d'eau.

J'utilise cette plateforme depuis quelques années maintenant. Home Assistant fourni son propre OS installable sur une Raspberry Pi ou basé sur des containers. J'ai opté pour cette dernière option avec un SSD à la place de la carte SD. Home Assistant passe son temps à écrire et lire des données. Cela réduit très rapidement la durée de vie des cartes SD. Autre aspect : les performances ! Incomparable, environ 20 à 100 fois plus rapide avec le SSD.

J'ai beaucoup de choses intégrées, du Zigbee, du Zwave, beaucoup de fait maison avec du .NET nanoFramework et beaucoup de choses connectées au server MQTT. Dans notre cas de Velux, comme vu précédemment, l'intégration se fait à travers le serveur MQTT.

Sans rentrer dans les détails de Home Assistant, vous avez la notion de *sensor* (capteur), de *switch* (interrupteur), de *platform* (plateforme) et de quelques autres. Ici, ce sont les 3 qui vont nous intéresser.

J'ai donc ma plateforme MQTT de définie, suivez la documentation, elle est très bien faite. Ensuite, je vais définir un capteur par Velux. Ici un exemple :

```
# velux
- platform: mqtt
  state_topic: "velux/percent0"
  name: "Velux Salle de bain"
  icon: mdi:window-open
  qos: 0
  unit_of_measurement: "%"
```

Vous reconnaîtrez du code expliqué précédemment notre topic et la valeur entre 0 (fermé) et 100 (ouvert). Quant à l'action pour ouvrir ou fermer, la voici :

```
- platform: mqtt
  name: "Velux Salle de bain"
  command_topic: "velux/action0"
  state_topic: "velux/state0"
  payload_on: "ON"
  payload_off: "OFF"
  state_on: "ON"
  state_off: "OFF"
  optimistic: false
  retain: true
  qos: 0
```

Là aussi, vous reconnaîtrez notre topic et nos mots clés ON et OFF. Il est possible d'ajuster tout cela, Home Assistant est très flexible pour cela. Juste un point d'attention ici, vous vous souvenez que je reviens toujours dans un mode connu quand ma télécommande vient à rebooter. Ici, avec l'option *retain* à true, on fait en sorte que l'ordre d'ouverture ou de fermeture reste tout le temps disponible. C'est une des options de MQTT. Cela permet ainsi de revenir toujours au mode que j'ai choisi quelque soit celui de la télécommande après un reboote par exemple.

Une fois le tout intégré dans le tableau de bord, cela me donne cela : **Figure 4**

Lorsque je commande l'ouverture d'un Velux à travers l'interrupteur, l'ordre est envoyé au serveur MQTT, l'ESP32 le

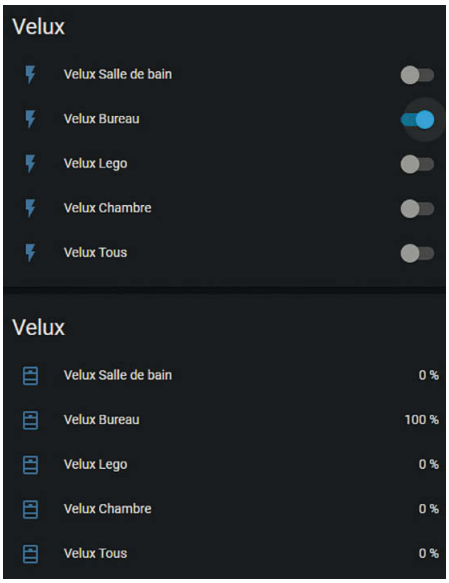


Figure 4

reçoit et pilote la télécommande pour suivre l'ordre puis reporte l'avancement, ici à 100%.

Par où commencer ?

Il vous faut d'abord un périphérique tel qu'un ESP32, installer une des dernières versions de Visual Studio, l'extension .NET nanoFramework qui va avec, flasher l'ESP32 et écrire votre « hello world » en C#, mettre un point d'arrêt et appuyer sur F5 Le reste viendra tout seul ! Et évidemment, nous avons tout prévu pour vous aider avec un guide détaillé pour commencer (https://docs.nanoframework.net/content/getting-started-guides/getting-started-managed.html).

Ensuite, pour commencer à apprendre comment l'utiliser avec .NET nanoFramework, vous avez de très nombreux exemples dans le repository à cet effet (https://github.com/nanoframework/Samples), ils vous permettront de rapidement prendre en main .NET nanoFramework et d'en apprécier toute sa magie !

Le code complet est lui disponible ici : https://github.com/Ellerbach/OpenVeluxMqtt.

Pour la partie hardware et électronique, vous trouverez assez facilement sur Internet des éléments qui vous permettront d'avancer. **Attention, ne jouez pas avec des hautes tensions et des fortes intensités ! Et attention aussi, la « fumée sacrée » n'est jamais loin ! Vérifiez et revérifiez bien vos montages. Testez petit à petit !**

Côté Home Assistant, suivez le guide depuis https://www.home-assistant.io/. Au début, n'hésitez pas à tester plein de choses quitte à supprimer et tout recréer proprement avec un peu d'expérience. Cela vient assez vite, la documentation est plutôt bien faite. Et si une intégration ne fait pas exactement ce que vous voulez, tout est open source en Python, vous pouvez toujours l'étendre, la modifier et l'utiliser à la place de l'officielle.

Et pour le serveur MQTT, je vous recommande Mosquitto : https://www.mosquitto.org/. Il y a beaucoup de tutoriels et d'aide pour l'installation. Suivant l'installation de Home Assistant que vous choisirez, vous pourrez l'intégrer de façon plus ou moins directe.

Bon hack !

Formater son code Java avec Prettier-java

Ling-Chun SO

Dès le début de sa carrière, Ling-Chun a pu découvrir l'open-source et les formidables communautés qui le composent et plus particulièrement au sein de l'organisation JHipster. Il a pu y apporter sa brique sur le générateur de code JHipster et a participé au développement du formateur de code Prettier-java. Aujourd'hui, il travaille en tant que consultant chez Ippon Technologies.

https://fr.ippon.tech/

ippon
POSITIVE TECHNOLOGY

Vous êtes fraîchement arrivé(e) sur un nouveau projet, bien motivé(e), vous décidez de vous promener dans le code source de l'application. Cependant, vous commencez à sourciller, du code mal indenté, des enchaînements d'appel de méthode sur une ligne à n'en pas finir, trop de retours à la ligne et bien sûr les problèmes sont toujours différents d'un fichier à l'autre ! Pas facile de lire le code, mais ne vous inquiétez pas, Prettier-Java arrive à la rescousse ! Le but de cet article est de présenter ce formateur de code d'un point de vue technique ainsi que son utilisation.

Prettier-java

Prettier-Java est un formateur de code Java open source. Il ingère du code en entrée et le transforme pour le rendre subjectivement plus agréable à lire et il uniformise aussi le style d'écriture au sein d'un projet. Cela a donc pour but de garder une certaine cohérence de forme mais aussi de régler certains types de problèmes tels que les soucis d'indentation par tabulation ou par espaces. Prenons par exemple le code ci-dessous non formaté :

```
public class Prettier {

 public boolean prettierIsAwesome(int myFirstArgument, int mySecond
Argument, int myThirdArgument, int myFourthArgument, int myFifth
Argument) {
   if(myFirstArgument == 1 && mySecondArgument == 2 && myThird
Argument == 3 && myFourthArgument == 4 && myFifthArgument
 == 5) {
   return true;
   }

   // if you do not want to format the next expression
   // prettier-ignore
   if(myFirstArgument == 1 && mySecondArgument == 1 && myThird
Argument == 1 && myFourthArgument == 1 && myFifthArgument
 == 1) {
   return true;
   }

   MyObject mo = new MyObject();
   return mo.very().very().very().very().very().very().very().very().very().
very().very().very().very().longMethod();
 }
}
```

Appliquons le formatage par Prettier-Java et observons le résultat. Il s'agit là d'un exemple volontairement exagéré. Il est à noter que dans le cas où une expression n'est pas trop longue, elle ne va pas être "cassée" sur plusieurs lignes. De manière générale, Prettier-Java suit dans les grandes lignes le Google Java style guide mais prend aussi des libertés. Elle se réserve le droit d'appliquer ses propres règles de formatage qui peuvent être dues à la volonté des développeurs ou des retours effectués par les utilisateurs.

```
public class Prettier {

 public boolean prettierIsAwesome(
   int myFirstArgument,
   int mySecondArgument,
   int myThirdArgument,
   int myFourthArgument,
   int myFifthArgument
 ) {
   if (
   myFirstArgument == 1 &&
   mySecondArgument == 2 &&
   myThirdArgument == 3 &&
   myFourthArgument == 4 &&
   myFifthArgument == 5
   ) {
   return true;
   }

   // if do not want to format the next expression
   // prettier-ignore
   if(myFirstArgument == 1 && mySecondArgument == 1 && myThird
Argument == 1 && myFourthArgument == 1 && myFifthArgument
 == 1) {
   return true;
   }

   MyObject mo = new MyObject();
   return mo
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .very()
   .longMethod();
 }
}
```

Fonctionnement

Prettier-Java n'est pas un formateur de code en soi mais un plugin d'un outil appelé Prettier. Il s'agit d'une application connue qui formate de nombreux langages et formats de données (JavaScript, TypeScript, JSON...) et Prettier-java est officiellement recensé comme projet communautaire qui ajoute le support de Java dans Prettier.

Le plugin correspond à l'association de deux projets, Java-Parser et Prettier-plugin-java. En effet, deux grandes étapes sont nécessaires pour formater le code Java. Dans un premier temps, le code doit être transformé en une donnée structurée par analyse syntaxique. Pour ce faire, il génère un arbre syntaxique qui est ensuite visité par Prettier-plugin-java, qui, lui, est chargé de rendre le code sous une nouvelle forme. Le fait d'avoir découpé le projet de cette manière permet de mettre à disposition un analyseur syntaxique Java pour d'autres cas d'usage.

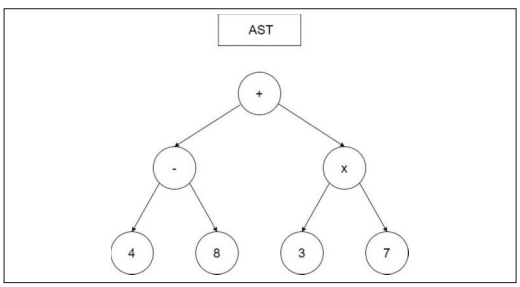
Petite histoire

A l'origine, Thorben von Hacht a été le premier à travailler sur le projet et a ensuite été repéré par Julien Dubois qui est notamment fondateur de JHipster et Java Champion. Prettier-Java, a été intégré dans l'organisation JHipster afin que cette communauté puisse contribuer activement au développement du projet.

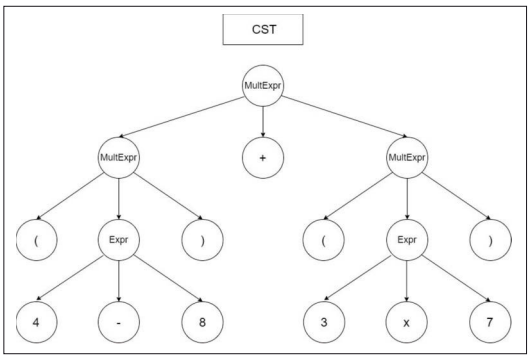
Il est à noter que Prettier-Java se base sur la librairie Chevrotain, il s'agit d'un outil permettant d'analyser du texte et de construire un arbre syntaxique associé à ce texte au même titre que ANTLR pour ne citer que celui-ci. A l'instar de ce dernier, Chevrotain est capable de faire le même travail sans générer de code et de manière beaucoup plus performante. Cette librairie a été créée par Shahar Soel et celui-ci a notamment pris la direction du projet Prettier-Java permettant ainsi d'exploiter tout le potentiel de Chevrotain au profit de Prettier-Java. Aujourd'hui le projet est disponible via npm et est maintenu par la communauté JHipster.

Java-Parser

La librairie Java-Parser est un analyseur syntaxique (parser) de code Java écrit en JavaScript mais à la différence de la plupart des parsers existants, celui-ci génère un Arbre Syntaxique Concret (Concrete Syntax Tree ou CST) et non un Arbre Syntaxique Abstrait (Abstract Syntax Tree ou AST) comme on aurait l'habitude d'en voir dans un compilateur ou un interpréteur. Voici un exemple de représentation d'un arbre syntaxique abstrait pour l'expression (4 - 8) + (3 x 7) :



Celui-ci est un exemple de représentation d'un arbre syntaxique concret pour la même expression :



Exemple d'un CST pour l'expression (4 - 8) + (3 x 7)

La différence entre un CST et un AST est qu'un CST est une représentation fidèle et même "littéral" en arbre de la structure syntaxique du code texte Java. À partir d'un CST, il est possible de recréer le code texte Java exact et sans ambiguïté. Un AST est quant à lui le résultat de la simplification d'un CST par réduction de l'information, pour n'en conserver que le nécessaire afin de donner un sens sémantique au code Java. C'est pour cela que les compilateurs et interpréteurs dans une écrasante majorité des cas construisent un AST. Prenons l'exemple précédent de l'expression (4 - 8) + (3 x 7). Les parenthèses ne sont pas présentes dans l'AST car elles n'ont pas d'utilités, bien qu'elles permettent de donner les groupements et priorités de calculs. La structure de l'arbre en lui-même permet de retrouver ces priorités de calculs en faisant un parcours en profondeur.

Prettier-java-plugin

La librairie Prettier-plugin-java correspond au plugin de Prettier qui permet de passer d'un arbre de syntaxe concret généré à l'aide du parser précédemment présenté à un code formaté via l'API de Prettier. En effet, Prettier offre un éventail d'outils permettant de formater relativement simplement le code, gérer l'indentation, la concaténation, etc. Comme dans l'écrasante majorité des interpréteurs et analyseurs, celui-ci se base sur le design pattern visitor afin de parcourir l'arbre et d'effectuer les traitements de formatage pour chaque nœud.

Utilisation

Installation et utilisation

Étant donné que Prettier-java est un plugin de Prettier écrit en Javascript, il requiert Node.js, il faut donc préalablement l'installer et utiliser une version supérieure ou égale à 12.

Pour installer prettier-java-plugin, il suffit d'exécuter cette commande :

```
npm install prettier-plugin-java
```

Il faut ensuite exécuter la commande suivante afin de formater tous les fichiers contenus dans le dossier courant (et sous dossiers) :

```
npx prettier —write "**/*.java"
```

Si vous souhaitez pouvoir l'utiliser non seulement dans vos projets nodes mais aussi dans n'importe quel répertoire, vous pouvez aussi l'installer de manière globale et exécuter prettier-java-plugin comme ceci :

```
npm install -g prettier prettier-plugin-java
prettier —write "**/*.java"
```

Prettier-ignore

Dans certains cas, Prettier-Java peut donner un résultat moins pertinent ou tout simplement pas satisfaisant. Il est possible d'ignorer le formatage de certaines parties de code avec les commentaires :

- **"// prettier-ignore"** qui ignore l'expression en dessous du commentaire
- **"// @formatter:on"** et **"// @formatter:off"** permettant d'activer/désactiver le formatage sur un bloc de code

Voici un exemple d'utilisation avec le code ci-dessous :

```
public class Prettier {

 public boolean prettierIsAwesome(int myFirstArgument, int mySecond
Argument, int myThirdArgument, int myFourthArgument, int myFifthArgument) {
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();
    // prettier-ignore
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();

    // @formatter:off
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();
    // @formatter:on
 }
}
```

Nous avons en sortie le code suivant :

```
public class Prettier {

 public boolean prettierIsAwesome(
    int myFirstArgument,
    int mySecondArgument,
    int myThirdArgument,
    int myFourthArgument,
    int myFifthArgument
 ) {
    mo
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .very()
    .longMethod();
    // prettier-ignore
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();
    // @formatter:off
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();
    mo.very().very().very().very().very().very().very().very().very().very().very()
.very().very().longMethod();
    // @formatter:on
 }
}
```

IntelliJ

Bien que passer par la CLI soit simple, cela reste fastidieux et nous utilisons très souvent des IDE pour travailler, notamment IntelliJ. Il est possible d'appliquer automatiquement Prettier-Java à chaque changement dans cet IDE. Il faut d'abord installer le plugin **Intellij File Watchers** et aller dans les paramètres **Tools/File Watchers** et configurer le watcher suivant :

Name: Prettier-java
File type: Java
Program: Le chemin vers l'exécutable Prettier
Arguments: —write $FilePathRelativeToProjectRoot$
Output paths to refresh: $FilePathRelativeToProjectRoot
Auto-save edited files to trigger the watcher: coché
Trigger the watcher on external changes: coché

Git Hook

Pour ceux n'utilisant pas d'IDE et/ou souhaitant appliquer automatiquement Prettier-Java avant de committer, il est possible de le faire avec Husky et Pretty-quick. Voici un exemple complet de la mise en place de Prettier-Java via Git Hook :
Code complet sur programmez.com & github

CI/CD

Il est possible d'intégrer dans un pipeline CI/CD la vérification que tous les fichiers ont bien été formatés par Prettier-Java. En effet, le CLI de prettier possède une option de "check" qui retourne 0 si les fichiers ont été préalablement formatés et des valeurs supérieures à 0 pour les fichiers qui ne sont pas formatés ou en cas d'erreur. Nous allons ici utiliser Github Actions pour vérifier que tous les fichiers Java ont été formatés et voici ce que l'on peut obtenir :
Code complet sur programmez.com & github
Cette configuration lance un job à chaque fois que l'on pousse ou ouvre une Pull Request sur la branche main et vérifie que tous les fichiers Java ont été formatés. Dans le cas contraire, le job se termine en erreur.

Conclusion

Prettier-java est donc un outil simple d'utilisation (une simple commande à lancer) qui permet donc de formater du code en forçant un certain style d'écriture. Cela permet notamment d'améliorer la lisibilité du code et de ramener un certain standard au sein d'un projet. Cependant la nécessité de Node.js peut en rebuter certains et le formatage imposé n'est pas forcément au goût de tous. Il est intéressant de noter qu'il existe d'autres alternatives comme google-java-format ou le plugin Save Actions pour ceux qui utilisent IntelliJ. Chacun ayant ses avantages et inconvénients mais la grande force de Prettier-Java réside dans sa polyvalence. Il s'adapte à de nombreux cas d'usage que ce soit dans l'utilisation d'un IDE, la mise en place de git hooks afin de l'exécuter avant un commit ou encore dans l'intégration dans un pipeline de CI/CD.

Sudoku : un problème récursif ?

La programmation récursive est souvent présentée en prenant comme exemples les calculs du Factoriel ou des termes de la suite de Fibonacci, faciles à comprendre. Or de nombreux problèmes possèdent une solution récursive. C'est notamment le cas du célèbre jeu du Sudoku…

Thierry LERICHE
Architecte et tech lead
Twitter : @ThierryLeriche
Linked'in :
https://www.linkedin.com/in/thierryler/

Qui n'a jamais griffonné son journal pour résoudre un Sudoku ? Pour rappel, le but du jeu est de remplir une grille avec des caractères en respectant trois règles simples :

- Chaque caractère est présent une et une seule fois sur chaque ligne ;
- Chaque caractère est présent une et une seule fois sur chaque colonne ;
- Chaque caractère est présent une et une seule fois sur chaque bloc.

Les grilles qu'on trouve dans les quotidiens sont préremplies et généralement constituées de 9 lignes par 9 colonnes. Les blocs sont alors des carrés de 3 par 3. Enfin il s'agit de placer les chiffres de 1 à 9. Il existe des variantes, sur la taille ou sur le jeu de caractères, mais le fonctionnement reste globalement le même. Pour simplifier, la suite de ce billet se concentrera sur la version classique. **Figure 1**

Pour en savoir plus sur le Sudoku, le mieux est de consulter la page Wikipedia dédiée, dont la grille d'illustration, est issue : https://fr.wikipedia.org/wiki/Sudoku

Note : *l'objectif de cet article est d'illustrer la programmation récursive avec un exemple ludique. La méthode proposée est une résolution de type « force brute ». Les spécialistes du Sudoku pourront proposer des algorithmes plus performants.*

Mise en place

Les méthodes de résolution (dont la solution récursive présentée ici) répondent toutes à un problème commun. Celui-ci peut donc être détaillé dans une interface :

```
public interface SudokuSolver {
 char[] POSSIBLE_CHARACTERS = { '1', '2', '3', '4', '5', '6', '7', '8', '9' };

 char[][] solve(final char[][] board) throws NotResolvableException;
```

L'interface *SudokuSolver* définit les caractères utilisables, ici limités aux chiffres de 1 à 9 pour simplifier. Elle définit également la méthode *solve* qui prend une grille incomplète en entrée et renvoie la grille complétée. Si la résolution est impossible, une exception est lancée. C'est un choix arbitraire et d'autres modes de fonctionnement sont possibles.
Sans surprise, le nom de l'implémentation d'illustration indique une programmation récursive :

```
public class RecursiveSudokuSolver implements SudokuSolver {

 @Override
 public char[][] solve(final char[][] board) throws NotResolvableException {
  ...
 }
```

Ce découpage sera naturellement reproduit dans les tests unitaires (non présentés ici) écrits pour l'occasion.

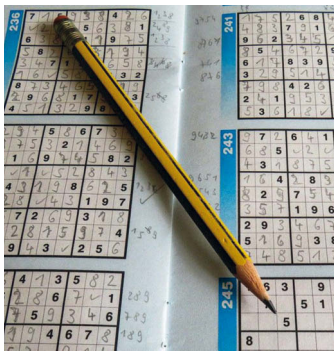
Règles

Les contraintes pour placer les caractères sur la grille sont les mêmes, quelle que soit l'implémentation du jeu. Le plus simple est donc de les définir comme méthodes par défaut au niveau de l'interface. Vérifier qu'un caractère n'est pas déjà sur une ligne ou une colonne est trivial :

```
default boolean isValueInRow(final char value,
                             final int rowIndex,
                             final char[][] board) {
 for (int j = 0; j < 9; j++) {
  if (board[rowIndex][j] == value) {
   return true;
  }
 }
 return false;
}

default boolean isValueInColumn(final char value,
                                final int columnIndex,
                                final char[][] board) {
 for (int i = 0; i < 9; i++) {
  if (board[i][columnIndex] == value) {
   return true;
  }
 }
 return false;
}
```

Ces deux méthodes sont relativement symétriques. Pour vérifier que la valeur n'est pas déjà dans le bloc, le plus simple est de boucler à partir de la cellule en haut à gauche du bloc, d'où les modulos :

```
default boolean isValueInBlock(final char value,
                               final int rowIndex,
                               final int columnIndex,
                               final char[][] board) {
 final int blockTopRowIndex = rowIndex - rowIndex % 3;
 final int blockLeftColumnIndex = columnIndex - columnIndex % 3;

 for (int i = blockTopRowIndex; i < blockTopRowIndex + 3; i++) {
  for (int j = blockLeftColumnIndex; j < blockLeftColumnIndex + 3;
j++) {
   if (board[i][j] == value) {
    return true;
   }
  }
```



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 1

```
 }
 return false;
}
```

Enfin, une dernière méthode permet de valider qu'une cellule peut recevoir un caractère donné :

```
default boolean isCorrectAt(final char value,
                            final int rowIndex,
                            final int columnIndex,
                            final char[][] board) {
 return !isValueInRow(value, rowIndex, board)
    && !isValueInColumn(value, columnIndex, board)
    && !isValueInBlock(value, rowIndex, columnIndex, board);
}
```

Affichage

Pour que le résultat soit plus facile à lire, il suffit de le formater dans la console. cf. code sur github.

Résolution

L'algorithme est relativement simple, mais provoquera un mal de tête en première lecture. La grille en entrée n'est qu'une étape. Chaque cellule remplie rapproche soit d'une solution soit d'un échec :

```
public class RecursiveSudokuSolver implements SudokuSolver {

 @Override
 public char[][] solve(final char[][] board) throws
NotResolvableException {

  final char[][] copy = copyOfBoard(board);

  for (int i = 0; i < 9; i++) {
   for (int j = 0; j < 9; j++) {
    if (copy[i][j] == ' ') {
     for (int idx = 0; idx < 9; idx++) {
      final char value = POSSIBLE_CHARACTERS[idx];
      if (isCorrectAt(value, i, j, copy)) {
       copy[i][j] = value;

       try {
        return solve(copy);
       } catch (final NotResolvableException e) {
        copy[i][j] = ' ';
       }
      }

     }
     throw new NotResolvableException("The for loop is over!");
    }
   }
  }

  return copy;
}
```

L'idée principale est de naviguer dans la grille à la recherche d'une case vide puis d'y placer un des caractères autorisés, sous réserve que ce caractère satisfasse aux règles. Cela donne donc une grille un peu plus remplie qu'à l'itération précédente. La méthode est alors appelée de nouveau, de manière récursive, avec la grille complétée d'une case.
Une exception est lancée lorsqu'une case ne peut accepter aucun caractère, éliminant ainsi une branche de l'arbre des solutions. L'algorithme s'arrête dès qu'une solution a été trouvée (la première faisant l'affaire) ou s'il n'en existe aucune. Pourquoi faire une copie de la grille ? Parce que c'est (toujours) mal de modifier les paramètres.

Exécution

Il ne reste plus qu'à assembler tout cela :

```
public static void main(String[] args) {
 System.out.println("SUDOKU SOLVER");

 final char[][] board = { //
   {'5', '3', ' ', ' ', '7', ' ', ' ', ' ', ' '},
   {'6', ' ', ' ', '1', '9', '5', ' ', ' ', ' '},
   {' ', '9', '8', ' ', ' ', ' ', ' ', '6', ' '},
   {'8', ' ', ' ', ' ', '6', ' ', ' ', ' ', '3'},
   {'4', ' ', ' ', '8', ' ', '3', ' ', ' ', '1'},
   {'7', ' ', ' ', ' ', '2', ' ', ' ', ' ', '6'},
   {' ', '6', ' ', ' ', ' ', ' ', '2', '8', ' '},
   {' ', ' ', ' ', '4', '1', '9', ' ', ' ', '5'},
   {' ', ' ', ' ', ' ', '8', ' ', ' ', '7', '9'},
 };

 final SudokuSolver solver = new RecursiveSudokuSolver();

 System.out.println("Tableau de depart :");
 solver.printBoard(board);

 try {
  final char[][] solvedBoard = solver.solve(board);
  System.out.println();
  System.out.println("Tableau de solution :");
  solver.printBoard(solvedBoard);

 } catch (NotResolvableException e) {
  System.out.println("This board can not be solved!");
 }
}
```

Pour une grille de 9x9, le calcul est quasi instantané, même pour des grilles compliquées : **Figure 2**

```
Tableau de solution :
-------------
|534|678|912|
|672|195|348|
|198|342|567|
-------------
|859|761|423|
|426|853|791|
|713|924|856|
-------------
|961|537|284|
|287|419|635|
|345|286|179|
-------------
```

Figure 2

Conclusion

La plupart des problèmes admettent une solution récursive, souvent simple et élégante. Le plus compliqué est alors d'identifier la récursion et ses conditions d'arrêt. Attention toutefois à ne pas en abuser dans un environnement sous-dimensionné ou en Java, car ce langage n'est pas celui qui s'y prête le mieux. Une autre fois, on parlera de programmation fonctionnelle. Pour aller plus loin… Bien sûr, écrire les tests, en prenant en compte des cas complexes, multiples, mais aussi des grilles incorrectes. Adapter le point d'entrée pour que la saisie de la grille soit plus aisée, par exemple à partir de fichiers. Et puis essayer d'autres algorithmes…
Retrouvez le code utilisé dans cet article sur GitHub : https://github.com/thierryler/article-sudoku

Créer son site e-commerce avec Gatsby Jamstack

PARTIE 2

Cynthia Henaff

Cynthia est développeuse front-end autodidacte depuis 2018. Auparavant infirmière de bloc opératoire, elle s'est prise de passion pour le développement et a décidé d'en faire son quotidien. Membre active de la communauté React lilloise, elle est actuellement lead front-end chez Tymate.

https://cynthiahenaff.com
https://twitter.com/monsieur_riz

La mise en place d'un site e-commerce est souvent vue comme laborieuse, mais avec l'écosystème de la Jamstack, nous pouvons créer un site marchand très facilement. C'est ce que je vais vous démontrer ici en créant un site e-commerce fictif qui proposera des romans graphiques inspirés de l'univers de la musique, nommé PaperJam. Mon répertoire GitHub est disponible sur https://github.com/cynthiahenaff/paperjam et le site démo disponible à cette adresse https://paperjam.henaff.io.

J'ai choisi d'utiliser la pile technique suivante :
- Gatsby en générateur de site statique
- DatoCMS pour le contenu
- Snipcart pour la partie e-commerce
- Vercel pour l'hébergement CDN

Pour bien débuter, nous allons créer un compte sur chaque SaaS que nous allons utiliser : DatoCMS, Snipcart et Vercel.

Initialisation du projet avec Gatsby

Gatsby est un générateur de sites statiques basé sur React et GraphQL. L'initialisation d'un projet avec Gatsby est assez simple et accessible à tous, il suffit de taper la commande suivante dans notre terminal :

```
npm init gatsby
```

Nous pouvons nous laisser guider dans les différentes étapes de la CLI jusqu'à la question suivante : *Will you be using a CMS?* . *On sélectionne alors la réponse DatoCMS*

À la question *Would you like to install a styling system?* répondons non, pour nous concentrer sur les autres aspects de la conception. Poursuivons avec la question *Would you like to install additional features with other plug-ins?* , sélectionnons les options ci-dessous :
- Add responsive images
- Add page meta tags with React Helmet
- Add an automatic sitemap
- Generate a manifest file

Pour finir, entrons notre clé API pour DatoCMS en réponse à la question *Configure the DatoCMS plugin. Cette clé se trouve sous le nom de Read-only API token dans la rubrique Settings/API tokens de notre projet*, sur le site DatoCMS. Elle *est protégée en écriture et ne permet que la lecture seule de la donnée.*

Pour lancer le serveur du projet, il suffit d'utiliser la commande suivante à sa racine.

```
npm start
```

Maintenant que notre serveur est lancé, la page d'accueil de notre application web est accessible à l'adresse http://localhost:8000. Le code de cette page se trouve dans le fichier src/pages/index.js. C'est à l'intérieur du dossier pages que les différentes pages de notre application web sont déterminées. Par exemple, il est courant sur un site e-commerce d'avoir une page « À propos » qui parle soit de notre société, soit de notre produit en détail. Nous allons donc créer cette page sur notre application. Comme nous souhaitons créer la route /about, nous allons créer un fichier about.js dans le dossier pages, comme ci-dessous :

```
|– src
 |– pages
  |— index.js
  |— about.js
```

```
import React from 'react';

const AboutPage = () => {
 return (
  <div>
   <h1>Ma page « À propos »</h1>
  </div>
 );
};

export default AboutPage;
```

La route http://localhost:8000/about est alors automatiquement créée.

Pour un site e-commerce, au-delà de la performance du site, il est primordial pour un bon référencement d'ajouter les balises de métadonnées dans les fichiers HTML. Pour ajouter ces éléments dans la balise head de nos pages, la plus simple des méthodes est d'utiliser la librairie Helmet (https://www.npmjs.com/package/react-helmet).

Nous allons créer un composant Layout dans notre dossier components que nous allons appeler sur nos différentes pages. Ce composant va également nous permettre de pouvoir utiliser la même mise en page sur tout notre site. Nous allons donc inclure le header et footer de notre projet dans ce composant.

```
import React from 'react';
import { Helmet } from 'react-helmet';

const Layout = ({ children, title, description, image }) => {
 return (
  <div>
   <Helmet
   defaultTitle="PaperJam"
   defer={false}
   htmlAttributes={{
       lang: 'en',
```

```
    }}
   >
    <title>{title}</title>
    <meta name="description" content={description} />
    <meta property="og:title" content={title} />
    <meta property="og:description" content={description} />
    <meta property="og:image" content={image} />
    <meta property="og:type" content="website" />
    <meta property="og:locale" content="en_GB" />
    <meta property="twitter:card" content="summary_large_image" />
    <meta property="twitter:creator" content="@monsieur_riz" />
   </Helmet>

   <div class="content">
    <header />
    <main>{children}</main>
    <footer />
   </div>
  </div>
 );
};

export default Layout;
```

Voici à quoi ressemble notre fichier pages/about.js après l'ajout de notre composant Layout :

```
import React from 'react';
import Layout from '../components/Layout';

const AboutPage = () => {
 return (
  <Layout
   title="Ma page 'À propos'"
   description="Découvrez pour quelle raison j'ai décidé de créer ce site e-commerce"
  >
   <div>
    <h1>Ma page "À propos"</h1>
   </div>
  </Layout>
 );
};

export default AboutPage;
```

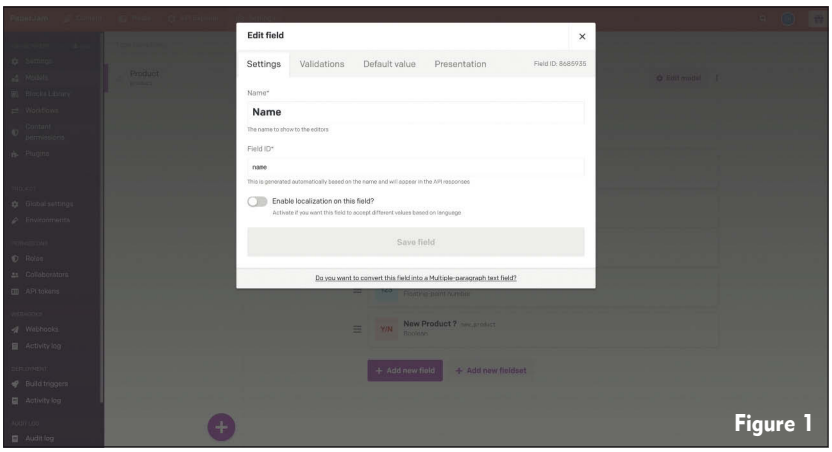


Figure 1

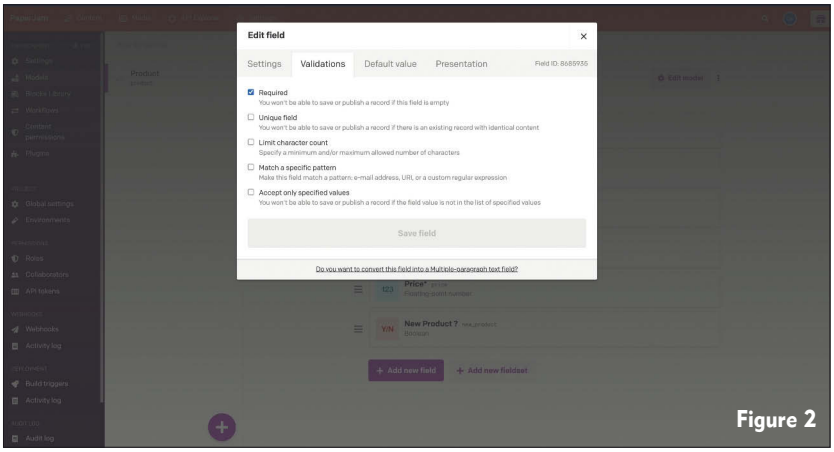


Figure 2

Création du contenu avec DatoCMS

Maintenant que notre projet est prêt, nous allons passer à la création du modèle de données d'un produit avec DatoCMS. DatoCMS a pour avantage d'être simple d'utilisation et ergonomique. Il permet de rendre facilement des éléments obligatoires tels que les noms de produits ou encore les textes alternatifs sur les images. Une galerie est disponible, ce qui permet de stocker les images, mais aussi de les retoucher directement. Il est également possible de gérer la traduction de notre site jusqu'à 5 langues différentes.

Pour cela, rendons-nous dans la rubrique Settings/Models du projet dans DatoCMS.

Nous allons nommer notre premier modèle de données Product; il contiendra les éléments suivants :

- **Name** - *Text — Single-line string* Nous allons rendre cet élément obligatoire, car il est essentiel pour le bon affichage d'un produit (**Figure 1 et 2**)
- **Slug** - *type SEO — Slug* Le slug est une partie de l'adresse qui va nous permettre d'identifier de manière unique une page. Il doit être dans un format facilement lisible pour les utilisateurs et les moteurs de recherche dans un souci UX et SEO.

DatoCMS permet d'utiliser une valeur de référence comme le nom de notre produit afin de le générer automatiquement. Il aura les validations suivantes : **Figure 3**

- **Thumbnail** - *type Media — Single asset* en termes de validation, nous allons rendre obligatoire le texte alternatif avec l'option Require alt and/or title car celle-ci est essentielle en matière d'accessibilité et de SEO.
- **Price** - *type Number – Floating-point number* Cet élément sera également obligatoire.

Durant les développements, après une modification d'un modèle de données, il est nécessaire de couper le serveur puis de taper la commande suivante dans le terminal avant de le relancer.

```
npm clean
```

Celle-ci va permettre d'effacer le cache de notre site et d'aller la rechercher sur DatoCMS

Maintenant que notre modèle est créé, nous pouvons revenir dans la partie Content et créer notre premier produit.

Création de la page d'accueil

Nous allons maintenant afficher nos produits sur la page d'accueil. Pour cela, nous allons récupérer les produits enregistrés dans DatoCMS dans le fichier index.js

La première chose à faire est d'importer graphql depuis Gatsby en ajoutant cette ligne en haut du fichier JavaScript.

```
import { graphql } from 'gatsby';
```

Sous la déclaration du composant HomePage, exportons une nouvelle constante appelée query. Le nom de la constante n'est pas important, car Gatsby recherche une chaîne GraphQL exportée du fichier plutôt qu'une variable spécifique. Il est important de retenir que nous ne pouvons avoir qu'une seule requête GraphQL par fichier.

La première partie de l'écriture d'une requête GraphQL consiste à définir le type d'opération (dans ce cas query) ainsi qu'un nom.

Gatsby intègre l'outil GraphiQL, qui est un environnement de développement disponible sur l'adresse http://localhost:8000/___graphql

Il permet de construire des requêtes GraphQL et de les copier/coller directement dans notre code. **Figure 4**

Maintenant que la requête est prête, nous pouvons directement récupérer l'objet data dans les props du composant comme ci-dessous :

Code complet sur programmez.com & github

Intégration de Snipcart

Notre page d'accueil contient maintenant une liste d'articles, nous allons pouvoir nous pencher sur l'intégration de Snipcart pour la partie e-commerce. Snipcart est un gestionnaire de panier d'achat très simple à intégrer dans une application web. Un simple ajout d'un bout de code JavaScript permet d'intégrer les ressources, fichiers JavaScript et CSS dans le code source. C'est similaire à l'ajout d'un script tiers comme Google Analytics. Gatsby ayant une très grande communauté, il existe déjà un plugin qui va permettre d'ajouter les scripts nécessaires à Snipcart. Nous allons donc installer la librairie gatsby-plugin-snipcart-advanced

```
npm install gatsby-plugin-snipcart-advanced
```

Une fois l'installation terminée, nous allons ajouter la configuration nécessaire au bon fonctionnement du plugin dans le fichier gatsby-config.js

```
module.exports = {
 plug-ins: [
  {
   resolve: `gatsby-plugin-snipcart-advanced`,
   options: {
    publicApiKey: "YOUR_SNIPCART_APIKEY",
    currency: 'eur',
    openCartOnAdd: true,
    useSideCart: true,
    // be careful with this mode cart. The cart in this mode has a bug of scroll in  firefox
   },
  },
 ],
};
```

La clé API nécessaire au fonctionnement de Snipcart est disponible dans la zone configuration de notre tableau de bord. Une fois connecté•e, cette zone est accessible en cliquant sur l'icône de l'utilisateur dans le coin supérieur droit de l'écran. Elle permet de configurer tous les paramètres administratifs tels que les passerelles de paiement, la livraison, etc.

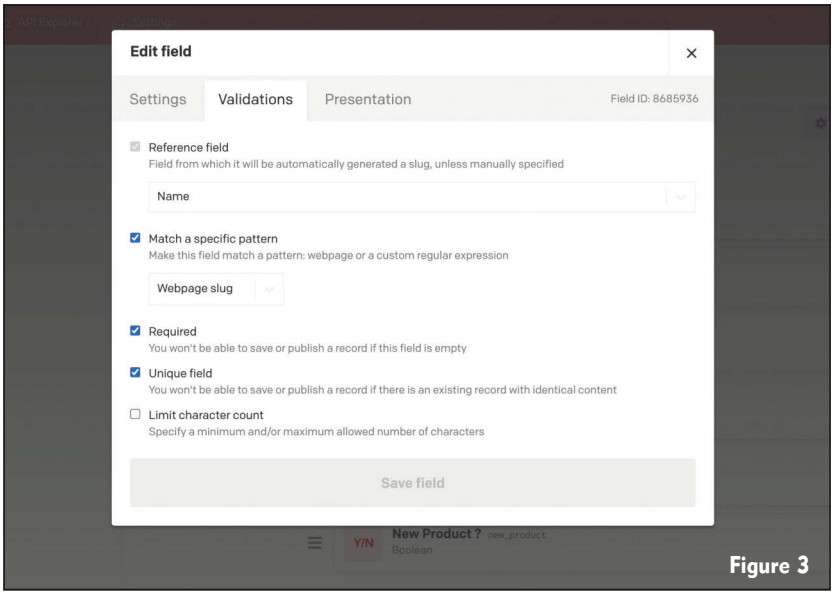


Figure 3

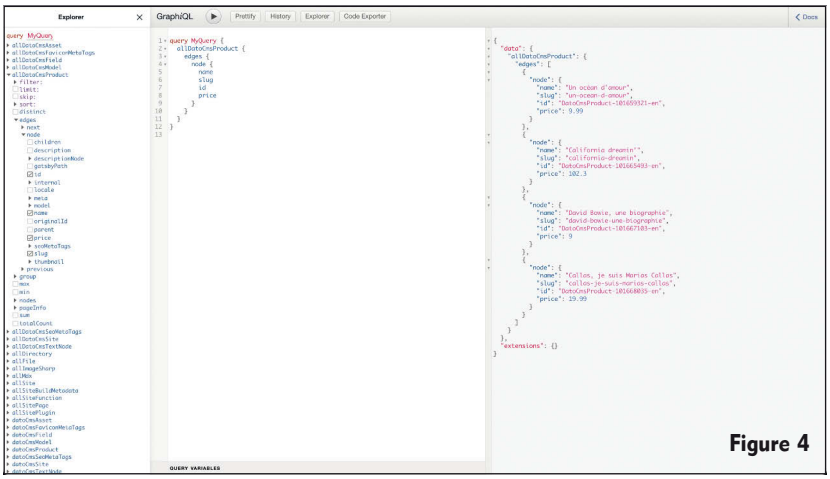


Figure 4

La clé API est accessible dans la zone API KEYS sous le nom "public test API key" (**Figure 5**)

Il est également nécessaire de personnaliser nos paramètres régionaux (ils se situent dans l'onglet Regional Settings) afin de configurer la devise prise en charge, le fuseau horaire et les pays dans lesquels il sera possible de passer commande.

Figure 6

Snipcart propose de nombreuses fonctionnalités qui sont paramétrables dans cette zone, telles que la gestion de l'inventaire, la livraison, la mise en page des mails automatiques ou encore la passerelle de paiement.

De base, Snipcart prend en charge de nombreuses passerelles de paiement comme Stripe, Paypal ou encore Square. Pour ce guide, nous ne configurerons pas la passerelle de paiement et resterons dans le mode « test » de Snipcart.

Ajout au panier d'un article

Afin que Snipcart réagisse aux évènements d'ajout au panier, nous devons ajouter la class snipcart-add-item sur les boutons d'ajout au panier de chaque produit, ainsi que les attributs suivants :

- data-item-id : Identifiant unique de notre produit
- data-item-price : Prix du produit
- data-item-image : URL de l'image du produit
- data-item-name : Nom du produit

```
import React from 'react';
import { graphql } from 'gatsby';

const IndexPage = ({ data }) => {
 const products = data?.allDatoCmsProduct?.edges ?? [];

  return (
   <div>
    <h2>Last product</h2>
    <div>
     {products.map(({ node }) => (
     <div key={node?.id}>
      <figure>
       <img src={node?.thumbnail?.url} alt={node?.thumbnail?.alt}/>
      </figure>
      <div>
       <h3>{node?.name}</h3>
       <button
        class="snipcart-add-item"
        data-item-id={node?.slug}
        data-item-price={node?.price}
        data-item-image={node?.thumbnail?.url}
        data-item-name={node?.name}
       >
        Ajouter au panier
       </button>
      </div>
     </div>
    ))}
   </div>
  </div>
 );
};

export const query = graphql`
 query getProducts {
  allDatoCmsProduct {
   edges {
    node {
     id
     slug
     name
     price
     thumbnail {
      url
      alt
     }
    }
   }
  }
 }
`;

export default IndexPage;
```

Figure 5

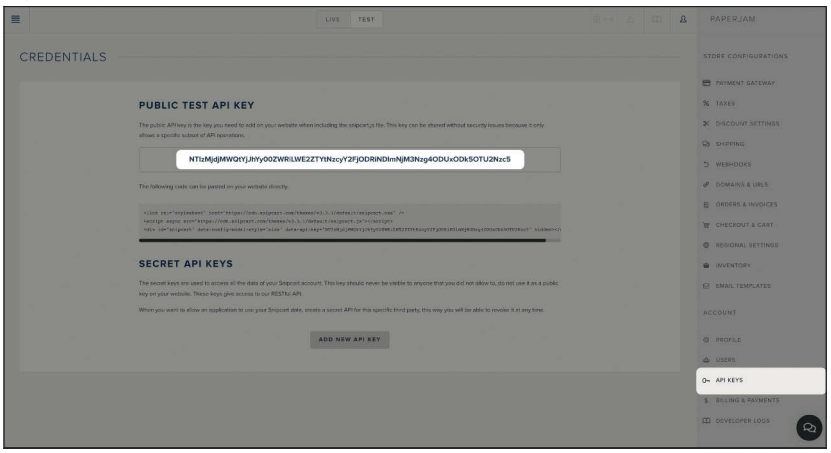
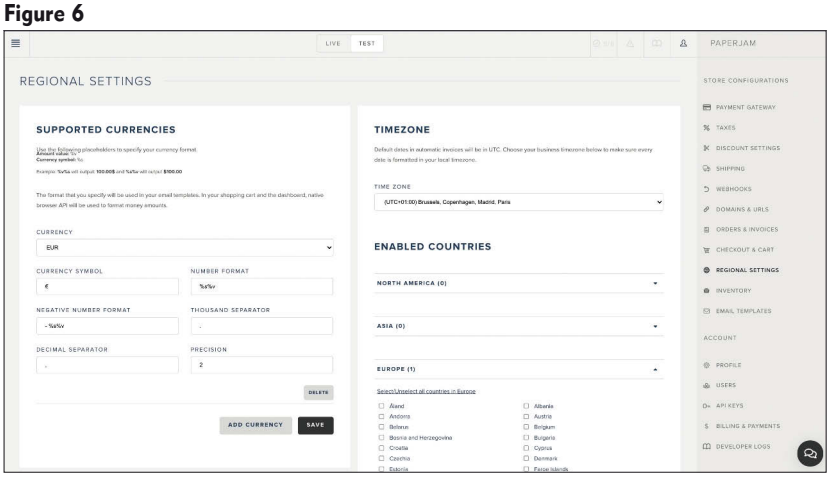


Figure 6



Visualisation du panier

Un résumé du panier permet aux clients d'y accéder rapidement et leur rappelle son contenu.

Nous avons la possibilité d'ajouter des éléments avec des classes CSS spécifiques sur notre application web qui permet d'interagir avec Snipcart directement.

Pour ajouter un bouton qui permet d'ouvrir le panier, il faut ajouter la classe snipcart-checkout. Il est également possible d'afficher le nombre d'articles dans le panier ou le prix total en ajoutant la classe snipcart-items-count ou snipcart-total-price sur un élément JSX.

Nous allons rajouter ce bouton à l'intérieur de notre élément header dans notre composant Layout pour afficher ces informations et donner la possibilité à notre utilisateur d'aller consulter son panier en détail.

```
...
<header>
   <button class="snipcart-checkout">
    Panier
    <span class="snipcart-total-price"></span>
    (<span class="snipcart-items-count"></span>)
   </button>
</header>
```

Notre tunnel d'achat est maintenant prêt à être testé sur le serveur.

Mettre en ligne son site avec Vercel

Nous allons utiliser Vercel qui est un des gros acteurs du déploiement automatisé côté front-end.

Vercel nous permet, en plus de simplifier le déploiement, de déployer notre site sur des serveurs aux 4 coins du globe. C'est ce que l'on appelle un CDN (Content Delivery Network). En plus du prix imbattable et de la certitude que le site résistera à n'importe quelle charge, il permet de charger le contenu de la page web depuis le serveur le plus proche de l'utilisateur et donc de réduire le temps de chargement.

Il est primordial pour utiliser Vercel, ou un autre CDN, d'héberger son code comme GitHub ou GitLab.

Vercel permet d'importer facilement un répertoire Git une fois authentifié avec la plateforme Git choisie. **Figure 7**

Au moment d'importer notre projet, Vercel détecte quelle librairie celui-ci utilise et préremplit la configuration nécessaire. Une bonne pratique en termes de sécurité est d'ajouter, nos variables d'environnements sur Vercel et de ne pas les exposer sur votre GitHub.

Il est temps maintenant de déployer notre site. Vercel va automatiquement lui attribuer une URL.

Maintenant que notre site e-commerce est disponible en ligne, il nous reste encore quelques étapes avant d'avoir un site fonctionnel. Commençons par la configuration sur le tableau de bord de Snipcart.

Dans la section Domains & URLS, nous allons ajouter le domaine que nous venons de créer avec Vercel. Cela permet de déterminer les domaines et sous-domaines sur lesquels notre boutique Snipcart peut fonctionner. **Figure 8**

Pour finir, nous allons importer tous les produits disponibles sur notre site dans Snipcart. Pour cela, dans la section Products / Fetch, il suffit d'entrer l'adresse de la page de notre application web sur laquelle se trouve notre liste de produits; il s'agit ici de la page d'accueil.

Snipcart va automatiquement scrapper notre page afin de récupérer les articles qui portent les classes que nous avons ajoutées précédemment.

Nous pouvons maintenant aller au bout de notre tunnel d'achat et valider notre toute première commande. Snipcart gère automatiquement l'envoi du mail récapitulatif au client.

Pour finir, notre site étant pré-construit sur notre serveur, il nous manque une fonction de mise à jour du site pour prendre en compte les données actualisées sur DatoCMS. Pour cela, il est nécessaire d'ajouter un build trigger dans les paramètres du projet DatoCMS. Celui-ci est directement connecté à Vercel et configuré. **Figure 9**

La "build method" permet d'ajouter automatiquement l'intégration DatoCMS sur Vercel, ce qui ajoute un bouton de redéploiement du site sur toutes les pages de DatoCMS. **Figure 10**

Notre site e-commerce est maintenant en ligne, accessible rapidement depuis partout dans le monde et prêt à recevoir nos premiers visiteurs !

SEO = Performance, mais pas que

Grâce à Gatsby, notre site e-commerce est performant et rapide. Mais si la rapidité est primordiale pour le référencement, cela ne fait pas tout. En effet, il ne faut pas pour autant mettre de côté les bases du SEO qui sont entre autres :

- Une bonne structure des pages en utilisant les balises appropriées (h1, h2, …)
- Un contenu unique et de qualité
- Des images optimisées afin de réduire leur temps de chargement
- Du texte alternatif sur toutes les images qui en nécessitent
- Un bon maillage interne (liens d'une page vers l'autre)
- Un grand nombre de liens entrants vers le site

Vous avez maintenant toutes les cartes en main pour lancer et optimiser votre site e-commerce, cependant n'oubliez pas votre cible : il faut toujours être utilisateur de son propre produit afin de comprendre ses clients et de les satisfaire au maximum.

Bon Jamstack !

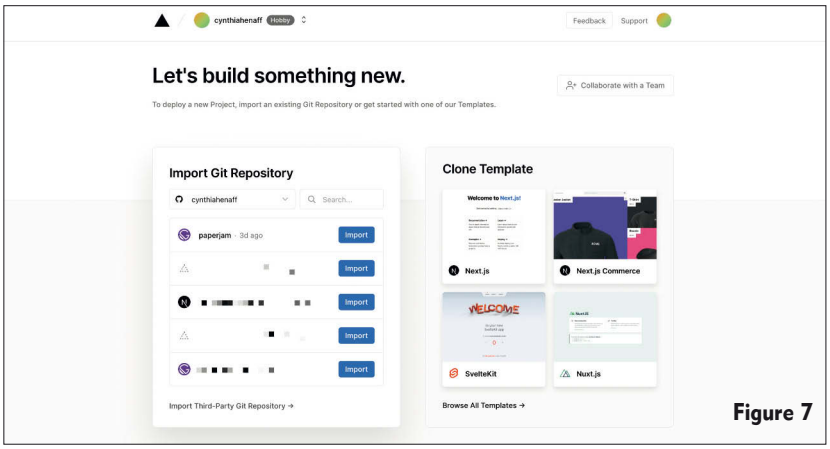


Figure 7

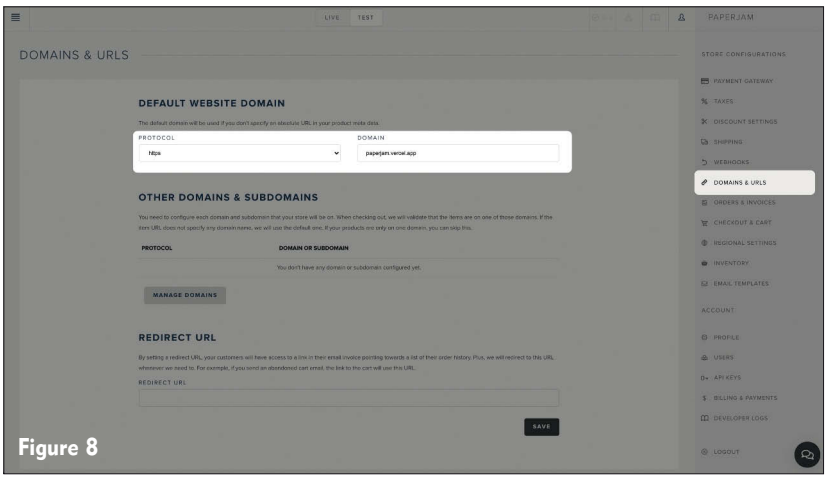


Figure 8

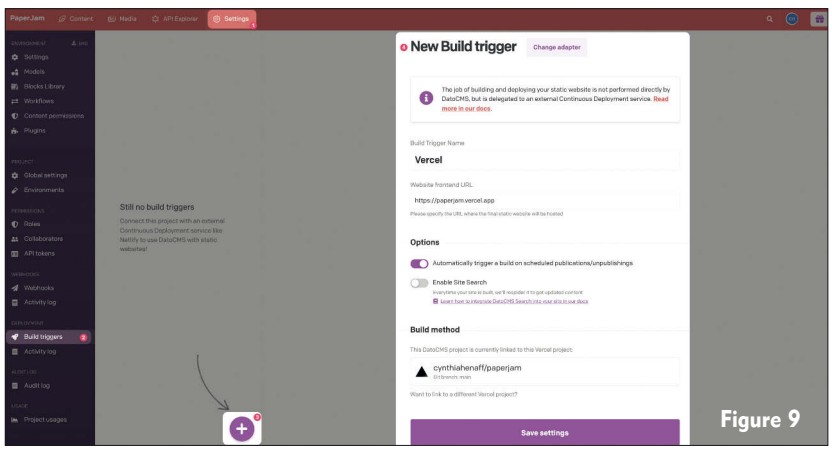


Figure 9

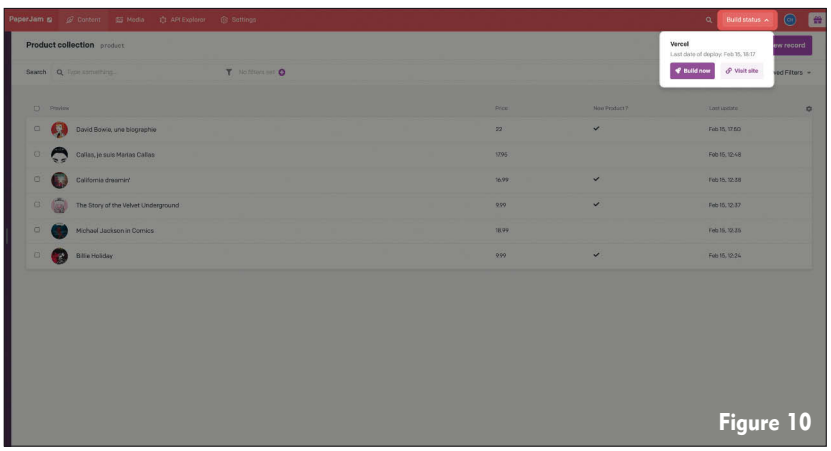


Figure 10

Java : le projet Valhalla

Le projet Valhalla est un projet d'OpenJDK démarré en 2014 dont le but est d'apporter à la JVM une manière plus flexible et performante de définir des types aplatis pour exprimer des données pures (flattened data types).

Loic Mathieu
Consultant Formateur à Zenika Lille.
GCP Google Developer Expert.
Contributeur Quarkus.
https://www.loicmathieu.fr
https://twitter.com/loicmathieu

zenika

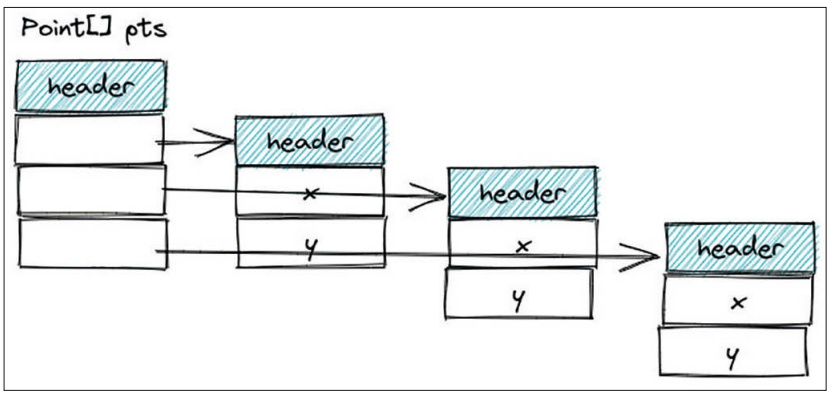
Le but est d'aligner le fonctionnement de la JVM avec les caractéristiques des hardwares modernes. Pour cela il définit de nouveaux types Java qui permettent de "coder comme une classe mais fonctionner comme un int".

Problématique

Le système de type de la JVM comprend huit types primitifs (int, long, etc.), des objets (agrégats hétérogènes avec identité) et des tableaux (agrégats homogènes avec identité).

Des données qui ne peuvent pas être codées via un type primitif (par exemple un point avec coordonnées x et y) seront codées via un objet. Or les objets sont stockés dans la heap, ont un header et doivent être référencés via une indirection en mémoire.

Prenons l'exemple d'un tableau de Point, nous aurons l'agencement suivant en mémoire :



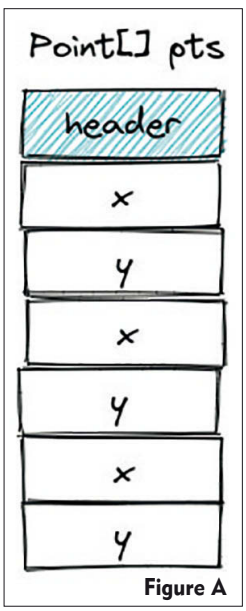


Figure A

Pour parcourir un tableau de 3 points nous devrons lire 4 headers d'objet et réaliser 3 indirections en mémoire. Les indirections en mémoire impliquent que la donnée n'est pas forcément localisée au même endroit. Les architectures mémoire modernes étant optimisées via leurs nombreux caches pour de la lecture co-localisée, la performance de lecture peut s'en trouver grandement impactée.

Le but du projet Valhalla est de pouvoir avoir un agencement en mémoire compact et sans indirection pour un tableau de point, comme on en a pour un tableau d'int. C'est un agencement aplati (flattened) et dense car sans header autre que celui du tableau. **Figure A**

Pour réaliser cela, le projet Valhalla va proposer deux nouveaux types qui sont Value Class et Primitive Class. Et comme corollaire l'unification des génériques entre primitifs et classes.

Value Class

Une Value Class est un nouveau type de classe qui est immuable (tous ses champs sont final), final, et n'a pas d'identité.

Un objet qui n'a pas d'identité ne peut être utilisé comme moniteur pour la synchronisation. La JVM (et le JIT) ***pourra*** donc instancier ces objets sans header, et les allouer en ligne (inline) si possible, pour éviter les indirections en mémoire.

Une instance de Value Class peut être nulle.
Comparer des instances de Value Class avec == compare la valeur de leurs champs.
Exemple de définition d'une Value Class :

```
public value class Point {
  int x;
  int y;
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
}
```

Un record pourra aussi être une Value Class.

Une Value Class étend java.lang.ValueObject qui lui-même étend java.lang.Object, les classes ayant une identité (donc celles sans le modifier value) étendront elles java.lang.IdentityObject qui lui-même étend java.lang.Object.

Plus d'information dans la draft JEP :
https://openjdk.java.net/jeps/8277163

Primitive Class

Une Primitive Class est un type de Value Class qui permet la définition de nouveaux types primitifs.

Comme une Value Class, elle est immuable (tous ses champs sont « final »), final, et n'a pas d'identité. Restriction supplémentaire, il ne peut avoir de champs d'un type qui dépend de lui-même (donc d'un type de sa hiérarchie).

Cette restriction permet pour les Primitive Class d'être représentées en mémoire en ligne (inline) sans indirection car leur forme (layout) est fixe, sans cycle.

Une instance d'une Primitive Class, comme une instance d'un type primitif, ne peut être nulle. Ce qui veut dire qu'elle a une valeur initiale, qui est une instance spéciale dont tous

les champs ont comme valeur leur défaut : nulle pour les références ou la valeur par défaut de chaque type primitif (0 ou false).
Comparer des instances de Primitive Class avec == compare la valeur de leurs champs.

Exemple de définition d'une Primitive Class :

```
public primitive class Point {
  int x;
  int y;
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
}
```

Les Primitive Class sont monomorphiques, elles appartiennent à un unique type à la compilation et au runtime. Mais comme elles peuvent participer à un polymorphisme via une interface, il est parfois nécessaire de manipuler une référence de ces primitifs.

Chaque Primitive Class a donc une Reference Class associée (qui est une Value Class) qui peut être utilisée via Point.ref dans notre exemple. Lors de l'utilisation de cette Reference Class, l'instance peut être nulle. Cette Reference Class permet aussi d'utiliser des Primitive Class dans du code qui n'a pas été prévu pour les supporter. Par exemple, si vous appelez une méthode qui s'attend à avoir en paramètre une classe ayant une référence, vous pourrez quand même utiliser votre Primitive Class et sa référence sera automatiquement utilisée. On voit ici l'approche compatibilité amont de Java.

La JEP 402 (https://openjdk.java.net/jeps/402) propose de transformer les wrappers existants des types primitifs (Integer, Long, Boolean, …) en Primitive Class.

Plus d'information dans la JEP 401 :
https://openjdk.java.net/jeps/401

Unified Generic

Avec l'unification des types référence et primitifs grâce aux Primitive Class, le fait de ne pouvoir utiliser de type primitif dans les types génériques devient de plus en plus limitant car ces nouvelles Primitive Class ne pourraient être utilisées comme un type paramétré pour une classe ou une méthode générique.
De plus, cette limitation actuelle a entraîné une prolifération de classe spécialisée pour un type primitif en sus de la classe générique (OptionalInt, IntStream, …) ou la prolifération de méthodes spécifiques (Arrays.binarySearch(Object []a, Object key), Arrays.binarySearch(int []a, int key), …).

L'unification de la généricité a pour but de permettre d'utiliser un type primitif comme type paramétré pour une classe ou une méthode générique. La JEP ne mentionne pour l'instant que l'utilisation des Primitive Class et pas des primitives existantes.

Plus d'information dans la draft JEP :
https://openjdk.java.net/jeps/8261529

Specialised Generic

Cette fonctionnalité est pour l'instant rarement évoquée et n'a pas encore de définition claire. L'idée est de pouvoir avoir un type générique qui a certaines méthodes ***spécialisées*** pour une certaines valeur du type paramétré.

Imaginons la classe suivante:

```
public class Container<T> {
  public add(T another) {
    // do something clever here
  }
}
```

La spécialisation d'un générique est la possibilité de définir une implémentation spécifique de la méthode add pour certaines valeurs du type paramétré T de manière transparente pour l'utilisateur de la classe Container.

Conclusion et spéculation

Le projet Valhalla, en permettant une optimisation de l'empreinte mémoire et un accès plus rapide à de la donnée, est un des projets les plus innovants et attendus d'OpenJDK.

La nouvelle API Vector (voir la JEP 417 :
https://openjdk.java.net/jeps/417) permettant du calcul matriciel via instruction SIMD (Single Instruction Multiple Data) devrait en tirer grandement parti, et grâce à ces deux fonctionnalités réunies, Java va devenir pertinent dans des situations où le C était encore traditionnellement utilisé (machine learning, deep learning, …).

Spéculation : j'aimerais voir arriver les premières JEP (Value Class et Inline Class) pour Java 19 (au jour d'écriture de l'article, une seule JEP est prévue pour Java 19), croisons les doigts ! Quoi qu'il en soit, ces JEP seront intégrées tout d'abord en preview feature, et nous ne les verrons certainement pas en version finale avant la prochaine LTS.

Introduction à l'Internet des Objets, un capteur hygrométrique pour jardin.

Nous allons voir dans cet article comment utiliser l'Internet des Objets (IoT) pour une réalisation simple telle qu'un capteur d'hygrométrie pour jardin. Cette création est née de la volonté de mon fils de créer un petit jardin potager. Ce jardin doit être arrosé régulièrement, mais comment ne pas oublier cette action primordiale ?

Paul Pinault

Passionné par l'Internet des Objets, auteur du blog disk91.com, référence sur ces technologies, je réalise des objets et services connectés à titre professionnel comme par curiosité. Ceci alimente mes connaissances que je partage ensuite sur ma chaîne YouTube au travers de MooC et sur mon blog.

Contact : @disk_91 / www.disk91.com

Pour ce faire, j'ai choisi d'utiliser le réseau LoRaWan Helium et initialement des modules RAK Wireless Wisblock, mais je vous présenterai dans cet article une solution basée sur une carte LoRa radio node qui est plus accessible aux débutants, puisque 100% basé sur Arduino.

L'IoT, des objets autonomes, capables de communiquer des années avec très peu d'énergie

L'Internet des Objets peut avoir de nombreuses définitions et intégrer un très large panel d'applications et de technologies. Pour ma part, j'aime définir l'IoT comme étant une catégorie d'objets capables de communiquer de façon totalement autonome (sans le relai d'un téléphone ou d'une box), sur de très longues durées, ceci pouvant être plusieurs mois à plusieurs années grâce à une capacité à communiquer très peu énergivore. Cette caractéristique innovante a permis l'émergence du terme IoT à partir de 2012 environ auprès du grand public et de l'industrie. De nombreux objets (ou plutôt machines) avaient préalablement des capacités de communications reposant sur des technologies conventionnelles impliquant en général une autonomie faible (technologies type GSM, LTE…) ou le recours à une passerelle locale (technologies BLE, Zigbee, WiFi…) Ces technologies étaient qualifiées de Machine to Machine (M2M).

Si cette classification peut être tout à fait discutée et si personnellement je trouve le périmètre de l'IoT particulièrement flou, au point que l'on puisse y consacrer des ouvrages entiers, je vous propose de l'accepter le temps de cet article pour comprendre que ce dont nous allons parler est bien de l'IoT tel que défini ci-dessus.

Des réseaux dédiés proposant un très bas coût de déploiement et d'usage

Il existe plusieurs technologies : LoRaWan, Sigfox, Kineis, Wize… pour ne citer que celles qui sont nées en France. Mais il ne faut pas oublier non plus NB-IoT, la déclinaison IoT des technologies issues du 3GPP, les technologies déployées en même temps que les réseaux mobiles (4G, 5G).

Ces technologies, pour que les objets puissent communiquer de façon autonome, doivent reposer sur la présence d'un réseau capable de capter leurs communications. Dans le cas de Kineis, par exemple, ce réseau est composé d'une flotte de satellites. Pour les autres, il s'agit de réseaux de communication terrestre. Ces réseaux doivent répondre à deux critères importants : une large couverture territoriale et un coût de déploiement faible. À titre de comparaison, un réseau 5G avec 9000 antennes couvre 41% de la population française et chaque antenne va coûter entre 50 000 et 150 000 € pour être installée, sans compter l'achat des bandes de fréquences nécessaires.

Des réseaux terrestres comme Sigfox ou LoRaWan vont nécessiter, respectivement, entre 1500 et 5000 antennes pour une couverture de plus de 95% de la population, chacune coûtant seulement quelques milliers d'€ et sans achat préalable de fréquences.

Cette approche à bas coût permet alors de bénéficier d'un coût de connectivité très faible pour chaque objet.

LoRaWan, des réseaux dédiés à l'IoT, accessibles au grand public

Dans les technologies citées, une grande partie s'est tournée vers des marchés professionnels, c'est cas de Wize, Kineis qui ne sont pas réellement accessible aux particuliers. Sigfox propose une accessibilité meilleure auprès des makers mais reste très orienté professionnel. Toutes ces technologies sont proposées par un unique opérateur qui va privilégier un modèle et une cible commerciale.

Les réseaux basés sur la technologie LoRa et les protocoles LoRaWan vont eux permettre plus de diversité d'offre, car cette technologie, propriété d'un fondeur de chip (Semtech), n'est pas associée à un opérateur.

Ainsi, s'il existe plusieurs offres commerciales en France (quasi-exception mondiale) avec Orange et Objenious adressant un marché professionnel avant tout. Mais il existe aussi des offres communautaires (TheThingsNetwork et Helium) accessibles à tous.

TheThingsNetwork (TTN), quand l'esprit de l'Open-Source s'empare des télécoms

Lancé en 2015 alors que la spécification de LoRaWan n'est pas encore aboutie, TheThingsNetwork reprend les standards de l'Open source pour créer une offre nouvelle : un réseau gratuit, déployé par des contributeurs volontaires, partout dans le monde, adossé à une offre payante associée à une meilleure qualité de service.

Ce réseau est un succès au point de devancer les offres des autres opérateurs en atteignant fin 2019 60% de part de marché dans le trafic LoRaWan dans les zones couvertes par le réseau. Fort de 20 000 antennes déployées par des particuliers, associations et entreprises, il est entre 2015 et 2020

le premier réseau LoRaWan mondial. Utiliser ce réseau est gratuit, s'il n'est pas présent dans votre localité, l'étendre par vous-même et votre voisinage, ne vous coûtera qu'une petite centaine d'euros et quelques minutes de configuration.

Helium, l'uberisation des télécoms est en route

Lancé en 2020, Helium, un réseau LoRaWan, lui aussi basé sur le déploiement de l'infrastructure par des particuliers est devenu en moins d'un an le premier réseau LoRaWan à l'échelle de la planète. Avec plus de 850 000 antennes déployées en un peu plus d'une année et des perspectives à 1000000 antennes à horizon 6 mois, il est déjà plus gros que la somme de tous les réseaux publics existants sur LoRaWan.
Son secret ? L'adjonction d'un fonctionnement décentralisé piloté par une blockchain, adossée à un Token qui crée un encouragement économique auprès de ceux qui déploient le réseau.
La couverture française est un peu en retrait vis à vis du reste de l'Europe et des USA mais couvre tout de même très bien la plupart des grandes villes. Là où le réseau est déployé; il est en général plus dense que les réseaux concurrents offrant ainsi une meilleure qualité de service. Helium n'est pas un réseau gratuit, mais comme 10000 messages sont offerts et que 100000 messages supplémentaires ne coûtent que $1 il en est tout comme.

Ce réseau étant nouveau et à l'avenir prometteur, c'est sur celui-ci que je vais m'appuyer pour la suite de cet article.

LoRa et LoRaWan

Avant de rentrer dans le dur du sujet, faisons un dernier point sur LoRa : il s'agit d'une technologie radio permettant de couvrir de longues distances avec peu d'énergie. Pour cela, elle offre une très grande résilience au bruit radio grâce à un principe de déplacement de fréquence. Grosso modo, les valeurs à transmettre vont l'être par une émission sur une fréquence qui va croître ou décroître. Cette croissance / décroissance qui constitue un mouvement de fréquence va permettre de coder la valeur émise.
Grâce à cela, un message, émis avec une puissance de 25mW (par comparaison WiFi est de 100mW) va pouvoir être reçu à une distance de plus de 10km (jusqu'à 50km dans de très bonnes conditions, 400km de façon exceptionnelle).

Pour arriver à cet exploit, il faut toutefois faire quelques compromis, ici, c'est celui de la vitesse de transmission. LoRa permet, dans ce que nous allons utiliser, des débits entre 250 bits / s et 5400 bits / s. Pas de quoi tout faire donc.

LoRaWan est une couche de protocole permettant d'utiliser la technologie radio LoRa dans le cadre de réseaux multiples. Ce protocole qui doit donc être implémenté dans l'objet définit les interactions entre celui-ci et un cœur de réseau appelé LoRa Network Server (LNS). Il existe plusieurs implémentations du protocole LoRaWan. Nous allons utiliser la version LMIC produite initialement par IBM, mieux adaptée aux environnements légers comme Arduino que la stack Semtech plus complète, mais plus gourmande.

Rak Wisblock une sorte de Légo pour l'IoT

Comme je vous l'ai dit, j'ai initialement réalisé ce projet avec un kit de développement RAK Wireless Wisblock. Ce kit à, l'avantage de ne pas requérir de soudure, d'avoir un processeur performant et de nativement fournir une solution d'harvesting solaire dans un boîtier étanche qui correspond tout à fait à ce cas d'usage. Vous trouverez tous les détails de cette solution sur mon blog.
Cette solution reste très adaptée au projet, mais sont prix avec les taxes d'importation dépasse les 100€. Je vais donc vous proposer une solution à moindre coût basé sur une carte LoRa radio node.

LoRa Radio Node, tout en un pour petits projets IoT

Cette carte que l'on peut trouver sur les sites classiques fournissant du matériel à bas coût vaut environ 10 à 15€ et possède tout ce qu'il faut pour un petit objet connecté.
Elle réunit sur une même carte une Arduino type Pro Mini avec un module de communication LoRa RFM95. Il vous faudra choisir un modèle 868MHz qui correspond à la fréquence utilisée en Europe. Elle possède en outre des connecteurs d'extension pour lui ajouter des capteurs supplémentaires.
Pour être programmé, il faudra lui adjoindre un adaptateur USB-UART communément appelé câble FTDI. Le Qr-code ci-joint vous renvoie vers un de mes articles qui détaille comment connecter cette carte à votre ordinateur pour la programmer en utilisant le logiciel Arduino.
Cette carte peut être alimentée de différente façon : la plus pratique consiste à utiliser une batterie à placer sur le support. Malheureusement, la communication radio sur cette carte demande un pic de courant autour de 120mA et beaucoup de piles ne vont pas supporter cela. Si vous souhaitez utiliser une batterie rechargeable dans le support, il faudra opter pour des LiFESo4 qui n'intègrent pas une grande capacité d'énergie. Pour cette raison, j'utilise souvent 2 piles Lithium au format ½ AA de 3.6V ou 3V offrant alors une tension totale de 6V à 7.2V et 1000 à 1200mAh.
Une fois l'Arduino programmé pour réduire sa consommation d'énergie entre deux mesures, cet objet pourra fonctionner des mois/années durant sans changement des piles.
Cette carte a toutefois certaines limites, comme nous l'avons vu, la pile de protocole LoRaWan est gourmande en ressources et sur le petit 328P avec seulement 32Ko de mémoire flash et 2Ko de mémoire RAM, il ne restera de la place que pour un programme utilisateur simple. La carte est donc tout à fait adaptée à capter d'hygrométrie d'un sol, la température / pression externe, une présence… mais elle ne permettra pas beaucoup plus. Pour aller au-delà, les kits comme Wisblock offrent une plus grande liberté.

En premier lieu, nous allons devoir légèrement modifier le kit LoRa Radio Node pour lui permettre de fonctionner sur LoRaWan. En effet, il faut réaliser un pont de soudure pour permettre au signal d'interruption de réception de message d'être reçu par le microcontrôleur sur l'un de ses ports d'entrée / sortie.

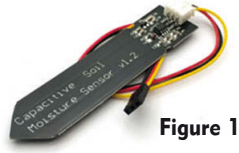
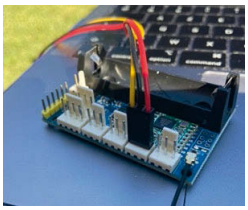


Figure 1

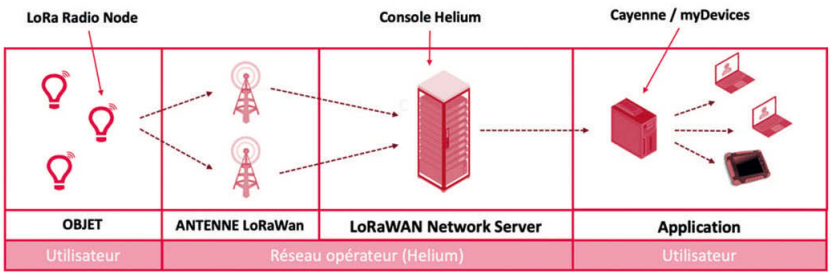


Capter l'humidité du sol

Il est possible de réaliser un capteur très simple avec deux clous comme je n'ai pas illustré dans l'article présent sur mon blog, mais nous pouvons aussi nous simplifier les choses avec un capteur capacitif que l'on trouvera facilement pour 1 ou 2 € tel que **figure 1**. Le connecteur est presque adapté à notre carte, il faudra toutefois veiller à inverser le fil rouge et le fil noir correspondant à l'alimentation pour que tout fonctionne correctement. Pour cela avec une aiguille ou une pince fine, il faut relever la lame plastique qui retient les pattes.

Le principe de ce capteur est de détecter l'humidité du sol par l'impact que cela a sur la capacité du capteur. Il transforme cette information en une tension variable sur sa patte analogique que nous allons pouvoir lire par la suite sur l'Arduino.
Ce capteur nécessite un courant de 700uA qui sera significatif dans la consommation et donc l'autonomie de la solution. Il serait donc opportun de piloter sont alimentation, lors des mesures uniquement, en utilisant un GPIO. Toutefois, pour une première version simplifiée, nous allons l'alimenter en continu en le branchant directement sur l'un des ports de la carte comme ci-contre. Nous choisissons le connecteur permettant d'accéder au port analogique A0.

Les différents composants de notre solution



Nous allons donc réaliser la programmation d'un objet qui va émettre une donnée, de façon régulière, vers un réseau LoRaWan. Pour cela, nous aurons besoin de déclarer cet objet auprès de cet opérateur, au niveau du network server, ici la console Helium. Enfin nous souhaitons visualiser ces données dans un rapport graphique accessible depuis Internet. Pour cela, nous allons utiliser la solution Cayenne de myDevices qui permet une intégration rapide et gratuite des données de notre capteur.
Ceci compose les 3 niveaux d'une solution IoT : l'objet, le réseau et l'application qui traite les données.

Installation de l'environnement Arduino

La carte nécessite l'installation de l'environnement Arduino (disponible sur www.arduino.cc) et utilise une carte « Arduino Pro ou Pro mini » avec comme sous-type « Atmega 328P (3,3V, 8MH)z ». Vous aurez besoin en complément d'installer la librairie permettant de communiquer sur LoRaWan. Pour cela, aller dans le menu « outil » puis le sous-menu « Gérer les bibliothèques ». Recherchez la bibliothèque « MCCI LoRaWAN LMIC library » écrite par IBM, Matthijs Kooijman… J'utilise pour cet exemple la version 3.3
L'environnement est prêt. Nous allons aussi utiliser la bibliothèque « Arduino Low Power » qui nous permettra d'endormir l'Arduino entre deux mesures et donc économiser fortement la batterie. Installez cette bibliothèque.

Fonctionnement de la LMIC

Pour utiliser la bibliothèque LMIC, il est nécessaire de configurer la zone géographique dans laquelle nous utilisons LoRaWAN. En effet la fréquence des communications en Europe est de 868MHz alors que les USA utilisent du 915MHz qui est la valeur par défaut.
Pour se faire, nous allons éditer le fichier *Documents/arduino/libraries/MCCI_LoRaWAN_LMIC_library/project_config/lmic_project_config.h*
Il doit contenir uniquement les lignes suivantes :

```
#define CFG_eu868 1        // configuration Europe 868MHz
#define CFG_sx1276_radio 1 // composant utilisé pour la communication
#define LMIC_LORAWAN_SPEC_VERSION  LMIC_LORAWAN_SPEC_VERSION_1_0_2
#define DISABLE_BEACONS    // suppression de code inutile
#define DISABLE_PING       // suppression de code inutile
```

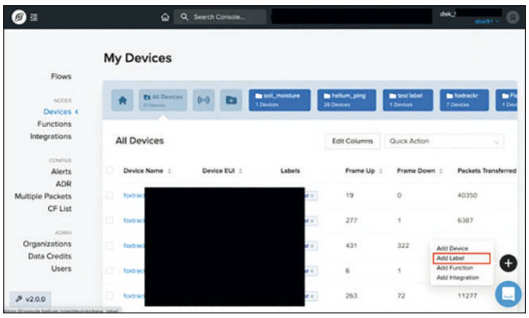
Cette configuration faite, la bibliothèque est utilisable.
La bibliothèque LMIC a un fonctionnement événementiel qui peut être un peu perturbant, car les opérations sont exécutées de façon asynchrone par l'appel, dans la boucle principale d'une fonction « os_runloop_once() » qui déclenchera les opérations nécessaires.
Nous allons donc pouvoir écrire un premier programme permettant d'envoyer un message chaque 5 minutes. Mais avant de commencer il faut que nous ayons des identifiants utilisables sur le réseau.

Configuration du cœur de réseau pour recevoir et traiter les messages de notre objet

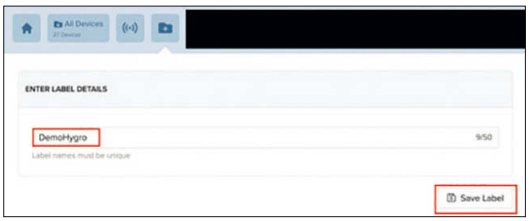
Nous allons maintenant inscrire notre objet sur le LoRaWan Network Server (LNS) Helium qui dans la terminologie Helium s'appelle un « routeur » et qui s'utilise avec un composant « console » que l'on accède sur https://console.helium.com
Il faudra dans un premier temps créer un compte sur cette console. C'est gratuit et vous obtenez 10 000 crédits de communication. Chaque crédit correspond, pour faire simple, à l'émission d'un message. Bref pour notre exemple ça va être plus d'un an de communication offert. Si vous souhaitez prolonger au-delà, sachez que 10 000 messages correspondent à une valeur de $0,1.
Pour configurer notre objet, il y a quelques concepts à connaître :

- Un « device » correspond donc à un objet et chacun a des clefs DEVEUI et DEVKEY qui lui sont propres et utilisées pour signer et chiffrer les messages.
- Un « label » correspond à un groupe d'objet pour lesquels nous allons avoir des opérations communes comme les intégrations. À un label correspond un APPEUI qui sera commun aux objets.
- Une « intégration » décrit comment transmettre les données reçues par le router vers l'application (pour nous cayenne myDevices)
- Un « flow » décrit les opérations à effectuer et les liens entre un « label » et une « intégration »

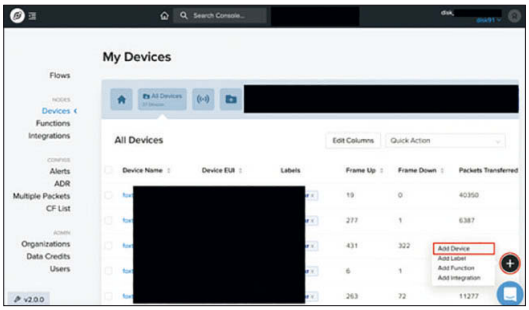
Pour commencer, il faut créer le « label »



Il faut donner un nom à ce « label », ici DemoHygro. Vous pouvez mettre autre chose, ceci n'a pas d'importance tant qu'il est réutilisé par la suite.



Alors, nous pouvons créer un « device » pour obtenir les clefs nécessaires à notre objet.

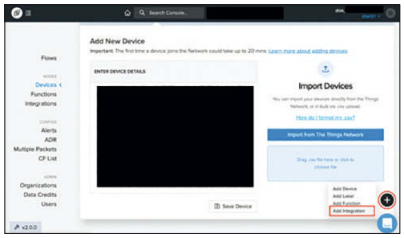


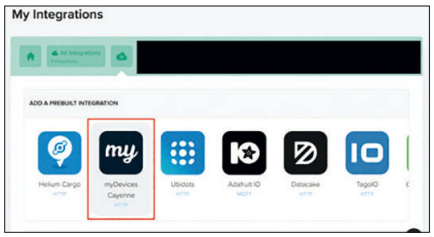
Nous donnons à l'objet le nom de notre choix (ici myDevice) les Dev EUI / App EUI / App Key sont automatiquement générés. Pour l'instant nous ne nous en occupons pas.

Il est important de rattacher cet objet au « label » que nous avons précédemment créé en le recherchant et en le sélectionnant pour le faire apparaître. Une fois ceci fait, nous pouvons enregistrer la création.

Nous allons maintenant pouvoir créer l'intégration vers le service « cayenne myDevices » qui nous permettra de consulter facilement les données et dont nous verrons les détails par la suite. Sélectionner myDevices Cayenne puis entrez simplement un nom pour cette intégration et validez.

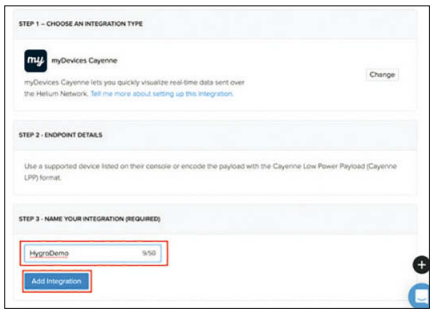
L'avantage d'utiliser une intégration prédéfinie et qu'il n'est pas nécessaire de rentrer les détails techniques relatifs à cette intégration.

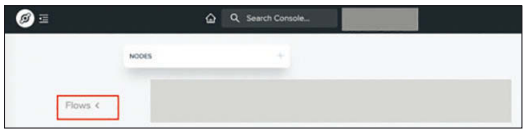




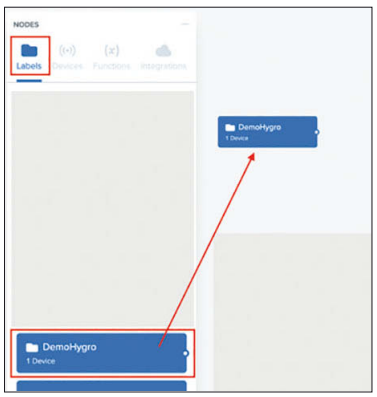
Nous allons maintenant créer le « flow » qui permet de relier cette intégration au label que nous avons créé. Ainsi, chaque message reçu par un « device » attaché au « label » HygroDemo exécutera l'intégration Hygro Demo (puisque nous les avons nommés de la même façon).

Sélectionner « Flows » dans le menu principal.

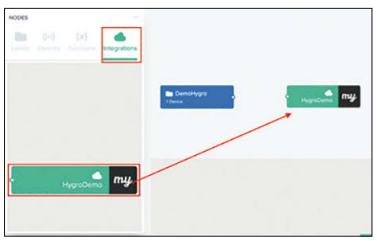




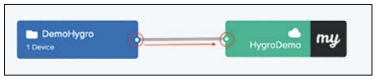
Puis cliquer sur « Nodes » et sélectionner les « Labels ». Choisir celui que nous venons de créer et le faire glisser (drag & drop) sur la zone de construction du « flow ».



Faire de même avec « l'intégration » après avoir sélectionné « intégration dans la barre de menu en haut ».



Ainsi nous avons l'ensemble des éléments que nous devons maintenant relier. Pour cela, relier les connecteurs des deux blocs en cliquant et étirant à l'aide de la souris.



Pensez bien à « sauvegarder » le « flow » créé en cliquant sur le bouton en bas de page.

Récupération des identifiants réseau

Lors de la configuration précédente, nous avons créé un

Device et avons aperçu ses identifiants. Nous allons devoir l'injecter dans le programme ci-dessous. Pour accéder aux identifiants, dans la console Helium, il faut sélectionner le Device pour afficher ses informations.

Nous allons cliquer sur l'icône avec les doubles flèches pour afficher les identifiants sous une forme facilement copie/collable sur Arduino et ensuite utiliser l'icône de copie pour mettre cette partie de l'identifiant dans le presse-papier. Il faut alors mettre cela dans le programme ci-dessous.

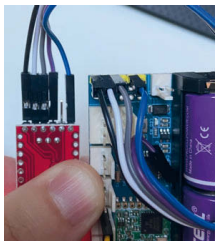
Programme de captation hygrométrie

Nous pouvons écrire notre programme qui va émettre la donnée lue sur le capteur d'hygrométrie toutes les 5 minutes sur le réseau Helium

```
#include <lmic.h>
#include <hal/hal.h>
#include <LowPower.h>

const lmic_pinmap lmic_pins = {
  .nss = 10,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 9,
  .dio = {2, 5, LMIC_UNUSED_PIN},
};

#define TXPERIOD  (5*60) // 5 minutes

static const u1_t PROGMEM APPEUI[8] = {0x60, 0x81, 0xF9, 0xXX, 0xXX, 0xXX,
 0xXX, 0xXX};
static const u1_t PROGMEM DEVEUI[8] = {0x60, 0x81, 0xF9, 0xXX, 0xXX, 0xXX,
 0xXX, 0xXX};
static const u1_t PROGMEM APPKEY[16] = {0xC3, 0x6F, 0xAE, 0x48, 0xDD,
 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX};

void os_getArtEui (u1_t* buf) {
 for ( int i = 0 ; i < 8 ; i++ ) buf[7-i] = APPEUI[i];
}
void os_getDevEui (u1_t* buf) {
 for ( int i = 0 ; i < 8 ; i++ ) buf[7-i] = DEVEUI[i];
}
void os_getDevKey (u1_t* buf) {
 memcpy_P(buf, APPKEY, 16);
}

void setup() {

  os_init();   // Intialisation de la bibliothèque
  LMIC_reset();
  LMIC_setClockError(MAX_CLOCK_ERROR * 10 / 100); // agrandit la
 fenetre de reception
  LMIC_setAdrMode(0);  // desactive le mode ADR
  // Configure les canaux utilisables pour les communications
  LMIC_setupChannel(0, 868 100 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(1, 868 300 000, DR_RANGE_MAP(DR_SF12, DR_
SF7B), BAND_CENTI);
  LMIC_setupChannel(2, 868 500 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(3, 867 100 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(4, 867 300 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(5, 867 500 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(6, 867 700 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(7, 867 900 000, DR_RANGE_MAP(DR_SF12, DR_
SF7),  BAND_CENTI);
  LMIC_setupChannel(8, 868 800 000, DR_RANGE_MAP(DR_FSK,  DR_
FSK),  BAND_MILLI);
  // Configure la vitesse et la puissance de transmission
  LMIC.dn2Dr = SF9;
  LMIC_setDrTxpow(DR_SF9,14);
  LMIC_setLinkCheckMode(0);
}

boolean canSleep = true;
uint32_t temps = TXPERIOD;  // transmettre au demarrage
uint8_t data[] = {
 0x01, 0x68, 0x00,    // canal 1, type de donnée Hygrometrie 1 ut = 0,5%
};

void loop() {
 if ( temps >= TXPERIOD ) {
    canSleep = false;
    // lecture de la donnée du capteur
    int16_t v = analogRead(A0); // 100 % = 313 | 50% = 678
    data[2] = map(v, 636, 450, 0, 100) * 2;
    // emission de la valeur
    LMIC_setDrTxpow(DR_SF9,14);
    lmic_tx_error_t err = LMIC_setTxData2(1, data, sizeof(data), 0);
    temps=0;
 }
 // put your main code here, to run repeatedly:
 os_runloop_once();
 if ( canSleep ) {
  LowPower.powerDown(SLEEP_8S, ADC_OFF,BOD_OFF);
  temps += 8;
 }
}

void onEvent (ev_t ev) {
  switch(ev) {
    case EV_JOINED:
      LMIC_setLinkCheckMode(0);
      break;
    case EV_TXCOMPLETE:
      canSleep = true;
      break;
    default:
```

```
        canSleep = false;
        break;
    }
}
```

Programmation de la carte



Pour programmer la carte, il est nécessaire d'utiliser un câble ou une carte FTDI qui permet de faire communiquer un microcontrôleur utilisant une liaison série avec un ordinateur via son port USB. Ces cartes sont faciles à trouver sur les sites marchant pour quelques euros. Il faudra relier les pins du programmeur avec celle de la carte comme indiqué ci-contre. Nous allons relier les masses (GND) ensemble, croiser RX et TX et relier DTR ensemble. L'alimentation VCC n'est pas reliée du côté de notre carte, car celle-ci sera fournie par les piles. Il est donc important lors de la programmation de s'assurer que les piles sont insérées dans la carte et fournissent une tension supérieure à 3,2V.

Vérification

Nous pouvons maintenant consulter la console Helium pour visualiser les messages transmis par l'objet. Il faut pour cela sélectionner l'objet dans la partie « Device » et descendre dans la page pour visualiser les communications. Si aucune communication n'apparaît, il est alors nécessaire de vérifier quelques points :

- Êtes-vous bien couvert par le réseau Helium ? (voir mappers.helium.com)
- Avez-vous bien entré les identifiants de l'objet dans le programme ?
- L'objet est-il bien inscrit sur le réseau (il y a un temps d'attente entre la création et son inscription qui peut atteindre 15 minutes)

Normalement vous devriez voir les communications suivantes dans l'écran de l'objet :

Le message JOIN REQUEST est envoyé par l'objet pour rejoindre le réseau. Ceci est la première étape d'échange des clefs permettant l'ouverture d'une session de communication chiffrée. Le message JOIN ACCEPT est envoyé en réponse par le réseau à l'objet pour ouvrir cette session. Cet échange n'aura lieu qu'une seule fois lors de la première communication. Il peut arriver que plusieurs communications comme celles-ci s'enchaînent dans le cas où l'objet n'a pas réussi à recevoir la réponse du réseau. Dans un tel cas :

- Attendre d'autres essais
- Si l'antenne du réseau est à quelques mètres de l'objet, il faut éloigner l'objet (le signal est trop fort)
- Si l'antenne du réseau est loin, il faut peut-être placer l'objet à l'extérieur pour lui offrir une meilleure chance de recevoir la réponse du réseau.

Suivent ensuite les communications UPLINK émises de façon régulière par l'objet pour communiquer les nouvelles valeurs d'hygrométrie du sol.

Création d'un tableau de bord avec Cayenne / myDevices

Il nous reste une dernière étape pour que notre solution soit complète : créer un tableau de bord permettant de visualiser les données de notre capteur. Nous allons utiliser la solution Cayenne de myDevices qui permet de gratuitement constituer ce type de tableau de bord.

MyDevice possède une autre force : celle de décoder automatiquement le contenu des messages émis grâce à un format de données spécifique. C'est pour cette raison que dans notre programme, nous envoyons 4 octets pour une valeur de 16 bits signée. Le premier octet que nous envoyons indique un numéro de capteur, ici 1 puisqu'il n'y a qu'un seul capteur, celui d'hygrométrie. Ensuite le second octet de valeur 2 indique que le format est un entier signé sur 16 bits. Il y a différents types que Cayenne va reconnaître et standardiser : température, pression, position GPS… De cette façon, le tableau de bord va pouvoir se constituer automatiquement avec les bonnes unités et les bonnes représentations.

Nous devons donc créer un compte sur **https://cayenne.mydevices.com/** et vous pourrez alors ajouter un nouvel objet.

Pour cela, cliquer sur « Add new » puis choisir « Device ».

Ceci va ouvrir une nouvelle fenêtre dans la zone de droite.

Dans cette partie de l'écran, reproduit ci-contre, il faudra sélectionner la technologie ici « LoRa » puis le réseau, ici « Helium ». Ensuite, il faudra utiliser le champ de recherche pour trouver « Cayenne LPP ».

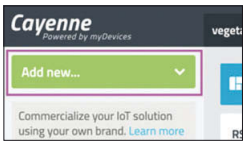
Une fois ceci fait, vous pourrez renseigner les informations identifiant l'objet. Il faut entrer dans le champ DEVEUI l'identifiant de l'objet que nous avons précédemment obtenu depuis la console et utilisé dans le programme.

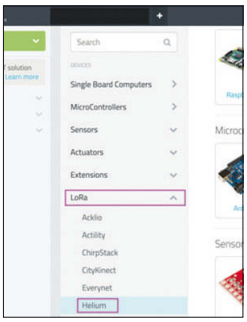
Nous pouvons laisser les autres informations telles quelles et valider la création de l'objet.

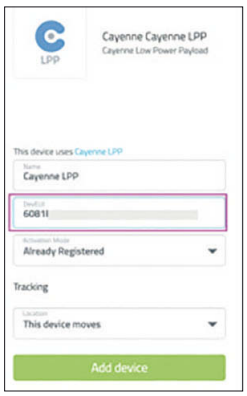
Nous pouvons maintenant visualiser les données dans le tableau de bord comme représenté ci-contre. Il sera possible de transformer celui-ci pour l'adapter à notre besoin : une gauge correspondant au niveau d'hygrométrie. Nous pourrons aussi placer une alerte par email en dessous d'un seuil.

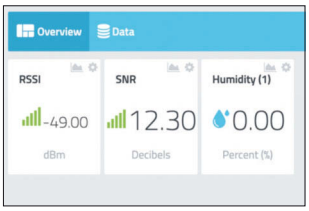
Finalisation

Une fois les tests terminés, il faudra fixer les composants pour rendre le système utilisable dans un pot de fleurs ou une solution étanche s'il est utilisé en extérieur. Vous pourrez aussi augmenter la période entre chaque communication, l'hygrométrie du sol évoluant à faible vitesse, 1 donnée chaque heure sera suffisante. La ligne `#define TXPERIOD` pourra donc être passée à la valeur (60*60). Vous pourrez aussi configurer avec Cayenne un « trigger » pour recevoir un email dès lors que le capteur retournera une valeur inférieure à une limite. Pour aller un peu plus loin, vous pourrez piloter l'alimentation du capteur par un GPIO de sorte à augmenter l'autonomie de la solution. Ces modifications faites, la consommation en veille sera inférieure à 50uWh environ ce qui devrait permettre une autonomie de plusieurs années avec les piles utilisées.









Jamais content

Directives de compilation

PROGRAMMEZ!

Programmez! n°253 - Juillet - Août 2022

Directeur des rédactions :
François Tonic - ftonic@programmez.com

Review : **Dorra Bartaguiz**

Contacter la rédaction : redaction@programmez.com

Ont collaboré à ce numéro : **Louis Adam (Zdnet)**
CommitStrip

Les contributeurs techniques

Sylvain Arboudie
Marc Gueury
Rosalie Zandona
Nicolas Haag
Jean-Christophe Quetin
Eric Sérandour
Louis Henrionnet
Pascal Engélibert
Lomig Perrotin
Philippe Boulanger
Sylvain Seccia
Antoine Galland
Sylvain Cuenca
Laurent Ellerbach
Ling-Chun SO
Thierry Leriche
Cynthia Henaff
Loïc Mathieu
Paul Pinault

Maquette
Pierre Sandré

Marketing – promotion des ventes
Agence BOCONSEIL - Analyse Media Etude
Directeur : Otto BORSCHA
oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 09 67 32 09 34

Publicité
Nefer-IT
Tél. : 09 86 73 61 08
ftonic@programmez.com

Impression
SIB Imprimerie, France

Dépôt légal
A parution

Commission paritaire
1225K78366

ISSN
1627-0908

Abonnement
Abonnement (tarifs France) : 55 € pour 1 an, 90 € pour 2 ans. Etudiants : 45 €.
Europe et Suisse : 65 € - Algérie, Maroc, Tunisie : 64 € - Canada : 80 € -
Tom - Dom : voir www.programmez.com.
Autres pays : consultez les tarifs sur www.programmez.com.

Pour toute question sur l'abonnement :
abonnements@programmez.com

Abonnement PDF
monde entier : 45 € pour 1 an.
Accès aux archives : 20 €.

Nefer-IT
57 rue de Gisors, 95300 Pontoise France
redaction@programmez.com
Tél. : 09 86 73 61 08

softfluent

Rejoignez l'ESN créée par des développeurs pour les Développeurs

Vous possédez une ou plusieurs de ces compétences ?

- Azure DevOps
- Cloud
- SQL Server
- Angular
- React
- C#
- CI/CD
- .NET Core
- ASP.NET MVC
- Microservices

Contactez nous !

Retrouvez-nous sur notre page carrière: softfluent.fr/nous-rejoindre

Ou contactez **Sylvie**, en charge du recrutement chez SoftFluent :

06 67 14 01 39

sylvie.benjelloun@softfluent.com

- Conseil
- Expertise
- Partage

softfluent.fr