F : 15,50 €

elektor 488 * SINCE 1961 *
créer > partager > vendre

MARS & AVRIL 2021
ELEKTORMAGAZINE.FR

n° 488 — mars & avril 2021

Elektor Helps
L'électronique aide quand ça va mal

découvrez MicroMod
p. 10

Rédaction invitée pour ce numéro :

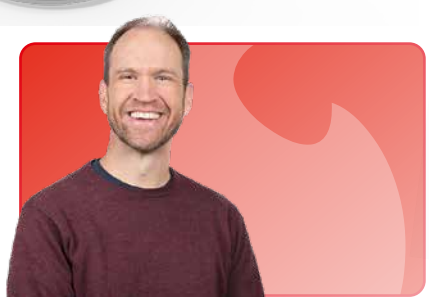
sparkfun

dans ce numéro :

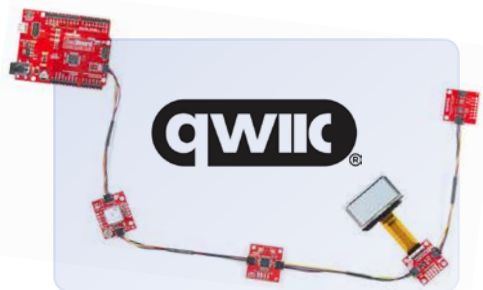
et bien davantage

L 19624 - 488 - F: 15,50 € - RD

PCB À La Carte
Conception et fabrication sur mesure
p. 62



Passion d'un ingénieur
Entretien avec Nathan Seidle, fondateur de SparkFun
p. 6



Écosystème Qwiic
Prototypage rapide avec I²C
p. 73

ISSN 0181-7450 • (B) 16,50 € • (CH) 29,00 FS • (CAN) 20.99 $ca • (And) 15,50 € • DOM surface 16,50 € • DOM avion 16,90 € • N Cal/S 2000 cfp

NOTRE GAMME PAR DES TECHNICIENS POUR DES TECHNICIENS

The best part of your project:
www.reichelt.com

Uniquement le meilleur pour vous - provenant de plus de 900 marques.

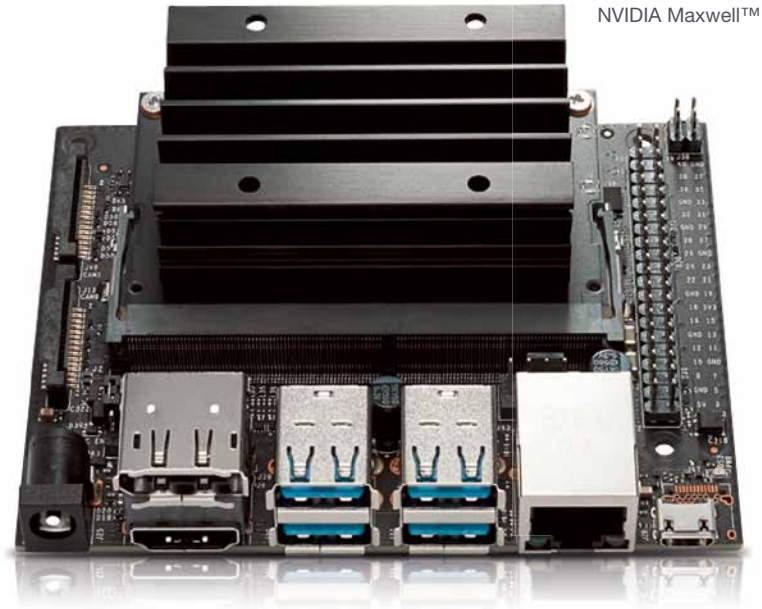
Nos responsables produits sont employés par Reichelt depuis de nombreuses années et connaissent les exigences de nos clients. Ils rassemblent une large gamme de produits de qualité, à la fois parfaits pour les besoins dans les domaines de la recherche et du développement, la maintenance, l´infrastructure informatique et la production en petites séries et adaptés pour les fabricants.

IA MODERNE POUR LA RECONNAISSANCE D'OBJETS ET LE TRAITEMENT DU LANGAGE

JETSON NANO™ – KIT DE DÉVELOPPEMENT

Calcul haute performance avec 472 GFLOP de 5 à 10 watts seulement.

NVIDIA.



GPU
NVIDIA Maxwell™

RAM
4 Go 64 bits LPDDR4

CPU
ARM Cortex-A57

N° de commande : **JETSON NANO KIT**

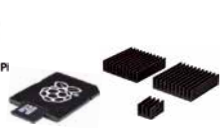
110,88 (92,40)

Camera
2x MIPI CSI-2 DPHY

Plus d'informations sur reichelt.com/ki

Plus de 1000 accessoires de A pour adaptateurs à Z pour ZIGBEE.

Découvrez maintenant ▶
www.reichelt.com/conseils-de-developpement



Types de paiement : Sofort. VISA MasterCard Maestro PayPal amazon pay

PRIX DU JOUR! Prix à la date du: 3. 2. 2021

- Excellent rapport qualité prix
- Plus de 110 000 produits sélectionnés
- Livraison fiable - depuis l'Allemagne dans le monde entier

www.reichelt.com

Assistance téléphonique: +33 97 518 03 04

reichelt elektronik – Tirer le meilleur parti de votre projet

Édition spéciale

rédaction invitée : sparkfun START SOMETHING

44ème année
n° 488 – mars/avril 2021

ISSN 0181-7450
Dépôt légal : janvier 2021
CPPAP 1120 T 83713
Directeur de la publication : Donatus Akkermans

Elektor est édité par :
PUBLITRONIC SARL
c/o Regus Roissy CDG
1, rue de la Haye
BP 12910
FR - 95731 Roissy CDG Cedex

Ont participé à cette édition : C.J. Abate, Mathias Claussen, Megan Hemmings, Luc Lemmens, Chris McCarty, Juan Peña, Justin Rajewski, Rob Reynolds, Derek Runberg, Glenn Samala, Avra Saslow, Nathan Seidle, Marcus Stevenson, Alex Wende
Maquette : Harmen Heida, Patrick Wielders
Version française : Yves Georges, Denis Lafourcade, Helmut Müller
Coordination : Denis Meyer

Pour vos questions : service@elektor.fr
www.elektor.fr | www.elektormagazine.fr

Banque ABN AMRO : Paris
IBAN : FR76 1873 9000 0100 2007 9702 603
BIC : ABNAFRPP

Publicité :
Raoul Morreau
Tél. : +31 6 4403 9907
raoul.morreau@elektor.com

Imprimé aux Pays-Bas par
Pijper Media – Groningen
Distribué en France par M.L.P.
et en Belgique par A.M.P.

ENTRE U.E. et É-U : le **partage du plaisir** de l'électronique

Ce numéro d'Elektor a quelque chose de particulier : nous avons proposé à *SparkFun Electronics*, nos amis américains de Boulder, dans le Colorado, d'être la rédaction invitée de cette édition. Cette idée est née en mai 2019 lors de notre participation à la Maker Faire à San Mateo, en Californie. Une rencontre autour d'un espresso entre Don Akkermans, P.-D.G. d'Elektor, de C. J. Abate, directeur éditorial d'Elektor et de Jahnell Pereira, responsable du développement stratégique de SparkFun, a jeté les bases d'une possible coopération de cette ampleur.

Ce jour-là, les participants ont rapidement compris que les deux entreprises avaient beaucoup en commun : des équipes d'ingénieurs talentueux, une forte présence en ligne, une passion durable pour la pratique de l'électronique et des communautés en pleine expansion d'électroniciens ingénieux et curieux. Peu après cette première rencontre, Elektor proposait déjà aux visiteurs de sa boutique en ligne de nouveaux produits SparkFun. Puis a mûri le plan d'une collaboration sur une édition complète d'Elektor. Et aujourd'hui, après des mois de cogitation, d'édition et de traduction, tu tiens ce numéro entre tes mains !

Tu constateras que l'équipe d'Elektor a travaillé en étroite collaboration avec SparkFun pour sélectionner, préparer et éditer des projets, des tutoriels, des entretiens et des bancs d'essai intéressants pour ce numéro spécial, à l'occasion de la 60e année d'Elektor. Une célébration qui te réserve encore d'autres surprises ! Nos commerciaux se sont efforcés de faciliter l'accès à de nombreuses offres combinées et outils dont tu as besoin pour tes propres projets innovants, que tu sois bidouilleur à Amsterdam, étudiant à Paris, informaticien à Cambridge ou ingénieur à Munich. Amuse-toi bien !

C.J Abate (Directeur du contenu, Elektor) et Jan Buiting (Rédacteur en chef, Elektor)

Tout d'abord, merci à Elektor pour son invitation à créer ensemble un numéro de cet incroyable magazine ! Nous sommes très heureux de partager avec la communauté Elektor des entretiens, des projets et des articles sur SparkFun. Nous avons apprécié la coopération avec l'équipe d'Elektor pour produire cette édition, ce sont des professionnels talentueux.

SparkFun Electronics vient de célébrer son 18e anniversaire. Cette date ponctue un long voyage, dont nous relatons pour les lecteurs d'Elektor quelques péripéties marquantes. Il y a dix-huit ans, Nathan créait l'entreprise dans sa chambre d'étudiant. SparkFun est passé d'un seul employé à plus de cent, d'une piaule d'étudiant à un grand siège commercial. Dix-huit années au cours desquelles SparkFun n'a cessé de faciliter la pratique de l'électronique de pointe, dans le jeu, le prototypage et l'expérimentation, en proposant des cartes de liaison faciles à utiliser, en élaborant de bons tutoriels et en partageant des projets passionnants. Ce numéro d'Elektor reflète à la fois ce qu'était SparkFun et ce vers quoi il va. Plongez dans notre univers grâce à des entretiens, et écoutez ce que nos ingénieurs disent de leur espace de travail et de leurs outils préférés. Notre équipe partage ici certains de ses anciens projets favoris, ainsi que les projets récents comme le robot quadrupède de Rob. Explorez les ressources d'Artemis, de Qwiic et de MicroMod et découvrez comment nous encourageons le prototypage rapide, la conception de produits et l'électronique de loisir.

SparkFun Electronics se réjouit de sa coopération avec Elektor et nous espérons que le résultat vous plaira. Amusez-vous bien !

Nathan Seidle (fondateur) et Glenn Sanala (PDG) de SparkFun

DANS CE NUMÉRO

rédaction invitée : SparkFun (États-Unis)

Rubriques

Articles de fond

Réalisations

Clock-Clock :
horloge récursive 40

Extra !

bientôt dans ces pages

Le numéro de mai & juin 2021 d'Elektor
Vous retrouverez dans le prochain magazine Elektor l'habituel mélange stimulant de réalisations originales, de circuits soigneusement étudiés, d'articles de fond, de sujets nouveaux, de trucs et d'astuces pour les électroniciens actifs.

Quelques-uns des points forts :

- Elektor fête ses 60 ans !
- Pico de Raspberry Pi & RP2040
- Alimentation rechargeable à batterie Li-Po
- Commutateur WiFi
- Java sur Raspberry Pi
- Caméra thermographique Seek Shot Pro
- Station de soudage facile à construire
- Bouton IoT de suivi chronologique

Et bien davantage !

Cette édition paraîtra le 6 mai.

Q&A avec Nathan Seidle

(Fondateur de SparkFun)

l'étincelle du plaisir
vision & passion d'un ingénieur

C. J. Abate (Elektor)

En 2003, quelques mois seulement après l'éclatement de la bulle internet, Nathan Seidle - alors étudiant en génie électrique à Boulder, dans le Colorado, a, initiative surprenante, lancé un site pour se mettre à vendre des composants électroniques sur l'internet. Près de 20 ans après, l'entreprise de Nathan Seidle prospère et prépare de nouveaux produits passionnants.

C. J. Abate : J'ai mené récemment des entretiens avec plusieurs ingénieurs de renom, Eben Upton de Raspberry Pi et Ryan Cousins de krtkl, au sujet de l'impact de la COVID-19 sur leurs entreprises, leur manière de concevoir et d'innover. Comment a-t-elle affecté le processus de conception créative chez vous?

Nathan Seidle : Je pense que la friction est le piment de la vie. La COVID a eu un impact terrible sur de nombreuses personnes, et cela a été difficile sur le plan personnel, mais j'ai trouvé les changements vraiment intéressants. J'ai appris que l'effet *coconnage* du travail à distance et en solitaire peut favoriser la concentration sur des défis techniques difficiles. En outre, le fait d'être réduit aux réunions par vidéo m'a montré à quel point il m'est possible de bien travailler à l'échelle mondiale. Mon équipe s'est étendue à trois continents. Sans la pandémie de 2020, je n'aurais pas été aussi à l'aise ou n'aurais pas fait ce pas aussi rapidement. J'observe avec fascination les processus créatifs des autres. L'innovation dans les entreprises locales de ma ville natale de Boulder a été incroyable parce qu'elles n'avaient pas le choix. Aussi ravageur que puisse être un gigantesque incendie de forêt – et ce sont des ravages que fait la COVID dans bien des domaines – je reste optimiste en voyant naître des entreprises qui prospèrent comme des fleurs sauvages.

Abate : Revenons à vos antécédents et à vos intérêts techniques. Quand avez-vous commencé à vous intéresser à l'électronique ? Avez-vous été inspiré par un ami, un parent ou un professeur ? Ou avez-vous découvert votre passion par vous-même ?

Seidle : Pendant des années, je dépiautais tout circuit de récup sur lequel je pouvais mettre la main. Comme *Rosie Revere, engineer* (livre illustré pour les enfants), mon bonheur était dans les poubelles, au grand dam de mes parents. Je me sentais un peu différent, aussi des autres explorateurs de poubelles, dans la mesure où je cherchais aussi à vendre des choses à mes camarades. Je confectionnais des câbles de programmation pour calculatrices TI pour y télécharger des jeux. Quand un tel câble de TI coûtait 100 $, je vendais mon câble bidouillé à mes amis pour 25 $. Dès 1994 j'ai lancé un BBS dont j'ai fait payer l'accès afin de pouvoir rembourser à mes parents la lourde facture téléphonique. J'ai économisé 500 $ et ai pu me payer un enregistreur de CD en 1995 (les CD-R coûtaient alors 8 $ pièce !) pour pouvoir vendre des copies (oui, des copies pirates) de logiciels et de musique à mes amis. Personne ne m'a jamais vraiment inspiré, je n'ai eu aucun mentor. J'ai toujours cherché à combiner ce que j'aimais avec mon travail.

Abate : Pouvez-vous nous parler de votre premier projet basé sur un microcontrôleur ?

Seidle : En 2002, j'allais obtenir un diplôme d'ingénieur, mais mes cours étaient purement théoriques. Je savais composer un diagramme de Bode et calculer la combinaison RC requise pour un filtre coupe-bande, mais de là à construire quelque chose moi-même… ? Non, rien de cela, dans aucun de mes cours.
Or, je voulais construire un appareil capable de lire quelques capteurs et d'amplifier la voix. C'était pour la pratique de l'aviron. C'est ainsi que je me suis intéressé au monde des microcontrôleurs. Ma première étape : le *Board of Education* de *Parallax*.
Le Stamp a été révolutionnaire pour le monde des microcontrôleurs et celui de l'apprentissage. Ah, l'extase de la première LED que j'ai fait clignoter en BASIC. Je commandais au monde physique, tout devenait possible ! J'ai tenté de lire mon capteur et d'en convertir le résultat. Les exigences du codage des opérations en virgule flottante épuisaient toutes les ressources de mon Stamp devenu de ce fait incapable d'assurer aucune des autres fonctions nécessaires pour mon rameur.

Abate : Qu'est-ce qui vous a conduit à lancer la société en 2003 ?

Seidle : Vers l'automne 2002, j'aidais un ami à construire une télécommande pour un robot rampant, conçu pour inspecter l'intérieur de tuyaux d'acier verticaux. Il fallait gérer quelques manettes pour la direction et un curseur pour la vitesse. J'ai utilisé un PIC pour la conversion analogique-numérique et la conversion en signaux numériques pour le pilote du moteur pas à pas.
Lors d'une passionnante séance de travail sur ce projet, mon PIC gérait les manettes et le moteur tournait, tandis que je déplaçais des objets pour faire un peu de place sur ma table, quand le programmateur sous tension est entré en contact avec une vis, peut-être un résidu de broche de LED ou un bout de fil, peu importe. Il y a eu une étincelle, un peu de fumée, et mon programmateur à 150 $ était mort.
Il m'en fallait un nouveau, et j'ai donc sauté sur l'internet pour trouver le programmateur PIC le moins cher possible. Le choix était vaste, mais mon attention a été attirée par la fréquence de la marque Olimex dans les résultats de recherche : bon matériel, prix très bas, mais site effrayant sans possibilité de payer en ligne. Après quelques échanges par courriel pour passer ma commande, j'ai compris qu'ils étaient basés en Bulgarie. Je suis curieux de tout, mais je n'avais pas de passeport et, ignorant tout de la Bulgarie, je ne voyais pas comment leur envoyer de l'argent.
La procédure de commande, lourde et difficile, m'a donné l'idée, plutôt que de ne commander qu'un seul programmateur pour remplacer le mien, d'en commander plusieurs avec quelques accessoires proposés par Olimex, que je revendrais à des amis à la fac. Je pourrais peut-être financer de cette manière ma propre dépendance à l'électronique et couvrir les coûts de ma soif d'apprendre !
Autour de Noël 2002, j'ai créé un site avec un logiciel de commerce électronique prêt à l'emploi, j'ai commencé à photographier les programmateurs, les câbles et les autres pièces à mesure de leur entrée en stock et j'ai rempli les papiers pour régulariser mes activités auprès de l'État du Colorado. Je n'oublie pas ce jour où mon programmateur est mort, ce moment précis où ont jailli l'étincelle (*spark*) et l'aiguillon du plaisir (*fun*). Le nom de domaine *SparkFun.com* était disponible, et c'est ainsi que naquit *SparkFun Electronics*. Le 3 janvier 2003 j'avais mes papiers pour les taxes sur les ventes au détail de l'État du Colorado, tout était en règle et j'étais prêt à faire des affaires. En quelques heures, j'ai reçu ma première commande, et j'ai réalisé qu'il me faudrait un carton pour l'expédier. Peu après est arrivée la première commande internationale de SparkFun et j'ai dû apprendre à expédier à l'étranger.

> *Ma passion est la conception de nouveaux produits pour rendre l'électronique accessible.*
>
> Nathan Seidle

(développement de nouveaux produits) et la gestion quotidienne d'une entreprise en pleine croissance ?

Seidle : D'abord, SparkFun c'était juste moi qui m'occupais du site web, de l'achat des produits et de la photographie, de l'expédition des commandes, du téléphone, de la conception et de la fabrication des produits, etc. Quand j'ai réalisé la rapidité du développement, j'ai commencé à embaucher parce qu'il me fallait de l'aide. En 2006, nous avons emménagé dans un espace commercial de 200 m² et en 2008 SparkFun comptait 50 employés.
La croissance rapide de SparkFun, m'a conduit à passer de la conception de produits à la gestion d'une entreprise. Avant de passer les rênes à Glenn, je m'efforçais de réserver quatre à cinq heures hebdomadaires à la conception technique au milieu de mes responsabilités de PDG.

Abate : Avant de devenir *Fondateur et ingénieur*, **vous avez signé quelques articles sur le blog de** *Nate l'ingénieur*. **Quelles ont été les difficultés et les satisfactions de la transition ?**

Seidle : Pendant les 13 années à la direction de SparkFun, je me suis éloigné de la conception de nouveaux produits et de l'écriture, les deux moteurs de SparkFun. C'est pourquoi, en 2015, nous avons annoncé l'embauche d'un nouveau PDG. Ce fut le début d'un lent processus de six mois. La recherche de notre PDG Glenn a été riche en défis, mais la décision a été la bonne. Grâce à cette transition et au retour de *„Nate l'ingénieur„*, j'ai pu lancer SparkX, notre version de *Skunkworks*. La conception et la création de nouveaux produits pour rendre l'électronique accessible sont ma passion. Le fait d'avoir SparkX et Glenn à la tête de SparkFun m'a permis de revenir à mes racines.
Quant aux difficultés de la recherche d'un PDG, voici ce que je peux en dire :

- L'embauche d'un PDG est comme une transplantation cardiaque, incroyablement risquée et stressante pour tous.
- À un moment donné, un conseiller m'a dit que l'embauche d'un PDG était extrêmement risquée et insensée. Cela a été dévastateur, car nous étions alors à la fin des entretiens, et je travaillais dur pour transférer SparkFun à quelqu'un qui pourrait prendre la relève avec succès. Alors que le temps était compté, il a été pénible d'entendre que j'avais tort. J'ai décidé de ne pas suivre ce conseil. Il faut demander conseil à des personnes intelligentes, les écouter, mais choisir son propre chemin.

Le téléphone à cadran rotatif de Nathan a été très apprécié dans la communauté SparkFun. Il en est question dans la rubrique „Rétronique" de cette édition.

La partie la plus gratifiante du processus a été de faire de Glenn le prochain PDG de SparkFun. Il bénéficie de plus de 20 ans d'expérience dans notre domaine et il a dirigé des organisations à travers les défis qui accompagnent l'énorme croissance organique et les tensions entre cultures issues des acquisitions stratégiques. Sa manière de travailler dans les entreprises ressemble à ma manière de concevoir de belles cartes. À quoi bon se rebeller contre la façon dont on est câblé !

Abate : Beaucoup de vos produits doivent avoir une histoire intéressante. Pourquoi avez-vous créé Artemis ?

Seidle : En octobre 2018, nous avons rencontré les gens de TensorFlow pour faire une carte basse consommation avec la nouvelle Apollo3 d'Ambiq. Dès le survol de la fiche technique, il était évident que cette carte était aussi stimulante qu'excitante ! Un Mo de flash, près de 400 Ko de RAM et une consommation exceptionnellement faible nous ont fait rêver à des possibilités qui dépasseraient celles de la vénérable Uno. Cependant, la présence d'un BGA de 81 billes avec un pas de 0,5 mm changeait nos règles du jeu habituelles en matière de PCB. Le projet a été baptisé Edge, parce qu'il pousserait la micro-informatique à la pointe du progrès. Nous nous sommes lancés dans la découverte de ce nouveau circuit intégré.
Nous avons pu livrer *The Edge* dans les délais et (presque) dans le budget prévus pour la conférence *TensorFlow* de Google. Ce qui m'a peiné, c'est que nous étions tenus à des spécifications „très serrées" pour seulement quelques millimètres carrés autour de l'Apollo3 ; pour le reste de la carte, banal, on pouvait se contenter de pistes et d'écarts ordinaires de 8 mil. Hélas il fallait payer pour la totalité du PCB. Or, *Edge* est assez grand, donc le coût du PCB a rivalisé avec celui du circuit intégré principal. En outre, le long délai de fabrication de ce type de PCB avancé impliquait que *Edge* souffrirait aussi de longs délais de livraison en raison du temps de fabrication des cartes. Toute modification éventuelle du PCB coûterait des centaines de dollars à tester et prendrait de nombreux mois.
Et si nous pouvions créer un module CMS pour l'Apollo3 ? À partir de cette question, les idées et les hypothèses de travail se sont succédé. SparkFun a décidé de continuer d'innover avec l'Apollo3. C'est de ce bricolage que sont nées la puce Artemis et les cartes de liaison (BoB) qui ont suivi.

Abate : Le module Artemis a été certifié par la FCC en août 2019. Comment s'est déroulée cette certification ?

Seidle : Longtemps SparkFun n'a jamais cherché à obtenir de certification complète par la FCC. Le processus était mystérieux, son coup rédhibitoire. Nous avions certes écrit sur les exigences de la FCC pour les produits et les projets de loisir, mais ce n'est que pour Edge et la création d'Artemis qu'a été prise la décision de nous aventurer en terrain inconnu.
Le processus de certification a été un processus d'apprentissage pour SparkFun : de la recherche d'une installation de test à l'obtention de devis pour les formulaires et les composants requis par l'installation de test. Une fois trouvée une installation de test, le processus de certification lui-même a pris environ deux mois et demi. L'entreprise nous a beaucoup aidés à obtenir des informations en retour et nous a guidés dans les modifications à apporter aux formulaires. Je ne peux pas dire avec certitude si SparkFun passera à nouveau par le processus de certification de la FCC, mais si c'est le cas, nous savons maintenant où nous mettrons les pieds.

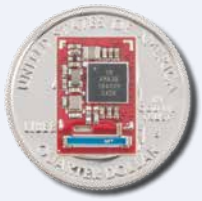
Abate : Pouvez-vous évoquer un ou deux des projets en cours ?

Seidle : Je continue à soutenir et à développer MicroMod et ALC. J'ai également travaillé sur le RTK Surveyor. Vous découvrirez bientôt parmi nos nouveaux produits ceux sur lesquels je travaille en ce moment, mais que je ne peux évoquer ici.

Abate : Avez-vous un circuit ou un projet préféré ?

Seidle : Vous me demandez quel est mon enfant préféré… je n'ai pas de préféré, mais j'aime rire. Le *Rotary-Cell-Phone* (téléphone cellulaire à cadran) a époustouflé et fait rire beaucoup de monde. Je prépare un cadeau pour un ami récemment promu capitaine dans la marine. Son grade lui impose désormais, tradition navale oblige, de sonner une cloche chaque fois qu'il monte sur un navire ou qu'il le quitte. Alors, pour rigoler, j'ai assemblé pour son épouse une clochette avec Wi-Fi, qu'elle peut faire sonner depuis son téléphone et le taquiner chaque fois qu'il entre dans une pièce de leur maison.

(210018 – VF Richard Kerr)



Module Artemis, taille réelle, sans blindage.

Quelques anecdotes

Premier amour - matériel ou logiciel ?
Souder. Le code, c'est intéressant, mais ce qui a changé la donne pour moi, c'est de pouvoir faire fondre du métal, de le plier électriquement et physiquement selon mes désirs et mes besoins.

L'outil de travail indispensable ?
Patafix, la punaise collante. C'est la 3e main qui tient bien, pour trois fois rien, et si commode pour bricoler quand vous en avez besoin.

Ingénieur préféré ?
Inspecteur Gadget. Nous voulons tous des armes „go-go gadget".

Que lisez-vous ?
Ready Player Two par Ernest Cline. Avant cela, Network Effect de Martha Wells me faisait rire aux larmes mais faisait flipper mes deux enfants. „Pourquoi tu ris des robots, papa ?"

Quel sujet lié à l'électronique vous passionne le plus en ce moment ?
Triangulation d'événements basée sur la vitesse du son en utilisant le GNSS comme référence temporelle.

De combien de sommeil avez-vous besoin ?
8,5 h dans l'idéal, mais avec les gamins, je dors plutôt 7 h.

Nathan présente les produits SparkFun sur sa chaîne YouTube en pleine expansion.

Démarrer avec MicroMod

Check out the poster on page 104

Nathan Seidle (États-Unis)

MicroMod est une interface compacte reliant un microcontrôleur à divers périphériques. Le système MicroMod est alors assimilé à un „cerveau“ qui se branche sur une „carte porteuse“.

Une carte processeur MicroMod mesure environ 22 × 22 mm et peut être insérée dans n'importe quelle carte porteuse (*carrier board*) MicroMod. Une petite vis maintient la carte processeur en place. Alors que la norme M.2 originale [1] était conçue pour remplacer des **périphériques** (l'utilisateur pouvait remplacer un disque dur statique par un plus grand), la norme MicroMod est conçue pour remplacer les **contrôleurs** (D'un processeur puissant, on peut passer à un contrôleur à faible consommation pour économiser la batterie).

Lectures suggérées

Si vous ne connaissez pas l'écosystème MicroMod, lisez l'aperçu sous [2]. Si les concepts suivants ne vous sont pas familiers, nous vous recommandons de consulter ces tutoriels avant de continuer.

- *Interface Périphérique Série (SPI).* Le SPI est couramment utilisé pour connecter des microcontrôleurs à des périphériques tels que des capteurs, des registres à décalage et des cartes SD [3].
- *Modulation de la largeur d'impulsion*. Une introduction au concept MLI [4].
- *Niveaux logiques.* Différence entre les appareils 3,3 V et 5 V et les niveaux logiques [5].
- *I2C*, l'un des principaux protocoles de communication embarqués utilisés aujourd'hui [6].

Comment cela fonctionne-t-il ?

La norme MicroMod s'appuie sur le connecteur M.2 et sa spécification [7] pour augmenter la disponibilité des pièces et réduire le coût du connecteur. Tous les „cerveaux“ MicroMod ont le même brochage. Par ex., les broches I2C du MicroMod ESP32 sont dans la même position que celles du MicroMod Artemis.

Une variante des cartes porteuses MicroMod permet l'accès à différentes technologies. Comme le connecteur MicroMod est standardisé, le contrôleur peut être facilement et rapidement remplacé en ce qui concerne la puissance de traitement, la consommation électrique et la connectivité sans fil. Par exemple, un utilisateur peut commencer avec le MicroMod Artemis et une carte porteuse RFID, puis décider qu'il a besoin du WiFi pour son projet. Le passage au MicroMod ESP32 permet à l'utilisateur d'ajouter instantanément la WiFi sans changer le matériel sous-jacent.

L'interface MicroMod est spécifiée dans les documents.

- Interface SparkFun MicroMod v1.0 - Brochage [8]
- Interface SparkFun MicroMod v1.0 - Descriptions des broches [9].

Aperçu du matériel

Q. Quel est le connecteur et la clé utilisés par le MicroMod ?

R. Le MicroMod utilise le connecteur standard **M.2 fig. 1)**. C'est celui que l'on trouve sur les cartes mères et les ordinateurs portables modernes. Nous recommandons le modèle de hauteur 4,2 mm.

TE fabrique le 2199230-4, largement disponible et pour un coût raisonnable [10] (0,56 $ l'unité pour 1000 pièces). Vous pouvez également commander le kit MicroMod DIY qui comprend un lot de 5 connecteurs, vis et plots soudables par refusion [11]

Il y a plusieurs positions pour le détrompeur sur le connecteur M.2 qui empêche l'insertion d'un appareil incompatible. La norme MicroMod utilise la **position „E“** mais s'écarte de la norme M.2 en décalant la vis de fixation de 4 mm sur le côté. La position „E“ étant assez courante, il est possible d'insérer un module WiFi compatible M.2 mais à cause du décalage de la vis de fixation, un appareil incompatible ne pourrait pas être monté sur une carte porteuse MicroMod.

Qu'est-ce qu'une carte processeur ?

R. Chaque carte processeur mesure environ 22 × 22 mm (**fig. 2**) et embarque un microcontrôleur ou un microprocesseur. Les broches du processeur sont routées vers le bord de la carte selon la spécification du brochage MicroMod.

L'USB D+/- Devrait suffire pour programmer une carte processeur. Il faut donc l'ajouter aux processeurs qui n'ont pas de support USB intégré. Par ex., le CH340E a été ajouté à la carte Artemis pour fournir la fonction de programmation série.

Chaque carte processeur doit avoir une LED d'état non routée vers le bord de la carte.

Note : La spécification MicroMod décale la position de la vis du centre de la carte de 4 mm vers la droite pour éviter la confusion entre un nombre croissant d'appareils qui utilisent le connecteur M.2 (tels que les cartes WiFi, les SSD, les modems cellulaires, etc.) et les modules MicroMod. Bien qu'un utilisateur **puisse** insérer une carte WiFi dans une carte porteuse d'acquisition de données SparkFun, ce décalage montrerait d'évidence cette incompatibilité.

La spécification MicroMod pourrait intégrer des tailles plus importantes à l'avenir, et les utilisateurs sont invités à créer leurs propres cartes de processeur, mais notez que le trou de verrouillage sur la plupart des cartes porteuses sera situé de manière à s'adapter au détrompeur MicroMod 2222.

Qu'est-ce que le brochage MicroMod ?

R. L'interface MicroMod est spécifiée dans les documents suivants :

- Interface SparkFun MicroMod v1.0 – Brochage [8]
- Interface SparkFun MicroMod v1.0 - Descriptions des broches [9].

Lors de l'utilisation du facteur de forme MicroMod, la connexion de toutes les broches n'est pas garantie. Veuillez consulter la documentation spécifique à votre carte processeur pour plus d'informations. Les données de référence se trouvent en ligne.

- Tableau général de brochage MicroMod [12].
- Descriptions générales des broches MicroMod [13].

La spécification attribue à chaque broche du connecteur M.2 une fonction déterminée. La spécification MicroMod est assortie de règles supplémentaires pour assurer la compatibilité inter-plates-formes. On peut avoir un maximum de 49× GPIO. En général, MicroMod met l'accent sur les types et les emplacements des interfaces. Par exemple, si une carte porteuse utilise la MLI, elle devrait comporter les broches 32 (alias PWM0) et 47 (alias PWM1), attribuées d'ordinaire à la MLI.

Interfaces supportées :

- USB pour la programmation et le débogage en mode série
- 2× Analogique dédié
- 2× MLI dédiée
- 2× E/S numériques dédiées
- 12× GPIO
- 2× I2C
- 2× SPI
- 2× UART
- SDIO
- USB-HOST
- CAN
- SWD
- PDM / PCM / I2S
- CAN différentiel

Les 12 GPIO peuvent sembler peu, mais une fois que toutes les autres interfaces ont été connectées (UART, SPI, I2C, PWM, ADC), alors elles devraient couvrir la plupart des applications restantes.

Figure 1 : Vue avant (en haut) et vue arrière (en bas) du connecteur M.2.

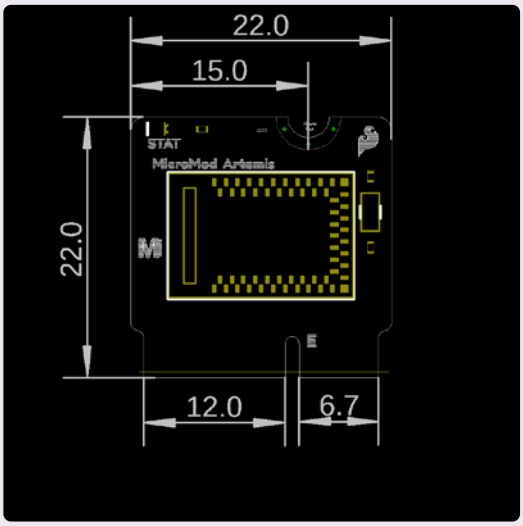


Figure 2 : Chaque carte processeur est à la norme M.2 de „2222" ou 22 × 22 mm de taille hors tout.

Branchement du matériel

Pour démarrer avec MicroMod, il faut une carte processeur et une carte porteuse. Ici, nous utilisons la carte processeur Artemis MicroMod avec la carte porteuse d'apprentissage machine. Alignez l'encoche supérieure de la carte processeur Artemis MicroMod sur le plot à vis de la carte porteuse d'apprentissage et insérez la carte en biais dans le connecteur M.2.

Remarque : il est impossible d'insérer le processeur à l'envers, car le détrompeur empêche l'insertion dans le connecteur M.2. En outre, pour éviter d'insérer un processeur compatible avec le détrompeur, la vis de fixation est décalée, interdisant la mise en place d'une carte incompatible. Cette carte resterait dressée à un angle (d'environ 25°) comme le montre la **figure 3**.

Une fois la carte enfoncée, appuyez doucement dessus et serrez la vis à tête Phillips (**fig. 4**). Nous recommandons le mini tournevis réversible SparkFun [14] ou le tournevis de poche [15] mais n'importe quel tournevis pour tête Phillips #00, #0 ou #1 fera l'affaire. Une fois la carte fixée, votre système MicroMod assemblé devrait ressembler à l'image de la **figure 5** !

Remarque : si vous n'avez jamais connecté un appareil CH340 à votre ordinateur, vous devrez peut-être installer des pilotes pour le convertisseur USB-série. Pour cela, consultez au besoin la page *Comment installer les pilotes CH340* [16].

Concevoir avec MicroMod

Puis-je fabriquer ma propre carte processeur MicroMod ?

R. Oui. Le matériel informatique de SparkFun est ouvert, non breveté. Tout ce que nous vous demandons, c'est de ne pas altérer les spécifications, de suivre les règles et de ne pas créer de confusion en proposant des interfaces similaires concurrentes ou partiellement compatibles. Nous recommandons de partir d'une de nos cartes processeur à conception ouverte. Actuellement, tous les fichiers sont au format EAGLE PCB. Si vous utilisez un autre logiciel de conception de circuits imprimés et que vous souhaitez ajouter votre conception à la liste de références, veuillez nous le faire savoir [17]. Les fichiers suivants sont sur github.

- Carte de processeur MicroMod ESP32 [18].
- Carte de processeur MicroMod SAMD51 [19]
- Carte de processeur MicroMod Artemis [20].

Comme indiqué ci-dessus, l'article *„Designing with MicroMod"* [21] explique comment créer de bonnes cartes processeur ou porteuse.

Puis-je fabriquer ma propre carte porteuse MicroMod ?

R. Oui ! C'est là que cela devient vraiment passionnant. Nous disposons de diverses ressources, dont une emprise de connecteur et son symbole pour Eagle PCB (**fig. 6**). Il y déjà pas mal de cartes porteuses ouvertes disponibles pouvant servir de référence et de point de départ pour la vôtre. Nous avons hâte de voir ce que vous en ferez.

Actuellement, tous les fichiers de ces cartes sont au format EAGLE PCB. Si vous utilisez un autre logiciel de conception de circuits imprimés et que vous souhaitez ajouter votre conception à la liste de références, veuillez nous le faire savoir [19]. Les fichiers suivants sont sur github.

- Carte porteuse MicroMod toutes broches (All The Pins, ATP) [22].
- Carte porteuse MicroMod d'acquisition de données [23].
- Carte porteuse MicroMod d'apprentissage machines [24].
- Carte porteuse MicroMod d'entrée et d'affichage [25].

De plus, nous avons écrit *Designing with MicroMod* [21] qui explique en détail comment créer de bonnes cartes processeur et porteuse.

Figure 3 : Carte processeur MicroMod insérée à un angle d'environ 25 degrés.

Figure 5 : Bon pour le service !

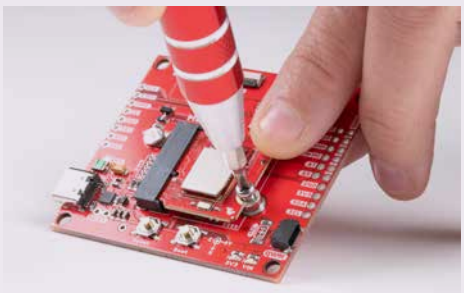


Figure 4 : Maintenez la carte processeur MicroMod bien en place et serrez la vis Phillips avec précaution.

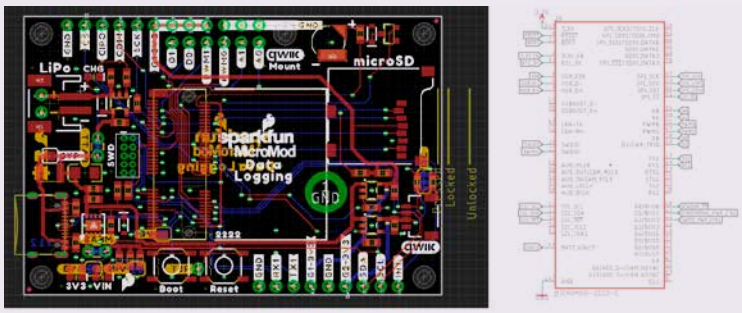


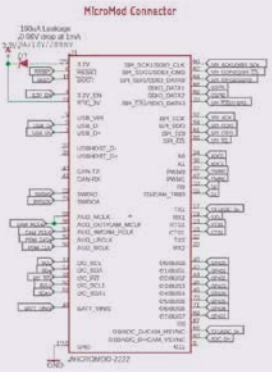
Figure 6 : Emprise Eagle PCB prête et symbole d'une carte porteuse.

Tutoriel

Concevoir avec MicroMod

Ce tutoriel vous guidera à travers les spécifications du processeur et de la carte porteuse MicroMod ainsi que les bases de l'intégration du dimensionnement MicroMod dans vos propres études de circuits imprimés.

https://learn.sparkfun.com/tutorials/designing-with-micromod



Guide de branchement de la carte porteuse d'acquisition de données MicroMod

Démarrez avec un système d'acquisition de données personnalisable avec la carte MicroMod.

https://learn.sparkfun.com/tutorials/micromod-samd51-processor-board-hookup-guide

Guide de branchement de la carte porteuse d'acquisition de données MicroMod

Démarrez avec un système d'acquisition de données personnalisable avec la carte MicroMod.

https://learn.sparkfun.com/tutorials/micromod-data-logging-carrier-board-hookup-guide

Guide de branchement de la carte porteuse d'apprentissage machine MicroMod

Lancez-vous avec ce tutoriel sur carte porteuse d'apprentissage machine :

https://learn.sparkfun.com/tutorials/micromod-machine-learning-carrier-board-hookup-guide

Accessoires

Si vous cherchez les accessoires présentés dans cet article, vous les trouverez chez SparkFun et Elektor

www.elektormagazine.com/esfe-en-micromod6

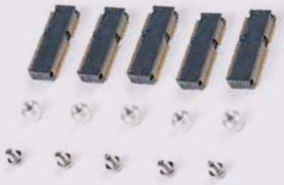
Carte processeur Micromod Artemis

Carte d'apprentissage machine

Carte d'acquisition de données

Carte d'entrée et d'affichage

Kit MicroMod DIY (par 5)



Lorsque vous concevez votre propre carte porteuse, gardez ces règles à l'esprit :

- Toutes les cartes porteuses doivent fournir une alimentation régulée de 3,3 V 1 A.
- Toutes les cartes porteuses doivent avoir une connexion USB D+/- pour la programmation.
- Toutes les cartes de processeur n'offrent pas l'accès à toutes les broches
- Les broches A0/1, PWM0/1 et D0/1 **doivent** être supportées par **chaque** carte processeur afin que vous puissiez être sûr qu'elles sont disponibles.
- Les ports UART1, SPI et I²C sont très répandus et se trouvent sur presque toutes les cartes processeurs, mais la présence de ports supplémentaires varie d'une carte à l'autre. C'est le cas d'un deuxième port I²C, alors, si votre carte porteuse vérifiez sa présence sur la carte processeur.

Pour vous aider à commencer votre propre carte porteuse, nous avons conçu le kit MicroMod DIY [26] (**fig. 7)** qui comprend un jeu de 5 connecteurs, vis et plots afin que vous disposiez de toutes les pièces „spéciales" dont vous pourriez avoir besoin pour la réaliser.
Le connecteur M.2 a un pas de 0,5 mm et des chevilles d'alignement. Le dessiner au pochoir et le souder par refusion à la maison sont possibles, mais nous recommandons un pochoir en inox (et pas en Mylar) et un four à refusion de bonne qualité pour éviter les ponts de soudure.

Figure 7 : Kit SparkFun MicroMod DIY (lot de 5).

Q. Parlez-moi de l'évacuation de la chaleur !

R. L'un des avantages de la norme M.2 est la possibilité de placer des composants sous le module, ce qui nous permet de pourvoir nos microcontrôleurs de radiateurs ! Pour cette raison, nous recommandons le connecteur de 4,2 mm de hauteur. TE fabrique le **2199230-4** [27], largement disponible et pour un coût raisonnable (0,56 $ l'unité pour 1000 pièces).

Q. Et si j'ai besoin de beaucoup de GPIO ?

R. Il existe des applications qui nécessitent plus de 12 GPIO. La spécification MicroMod est souple. Si vous souhaitez concevoir un MicroMod qui n'a que quelques périphériques connectés (par ex., juste UART et I²C) et utiliser le reste comme GPIO (45 broches GPIO disponibles ici), c'est parfait. Votre carte porteuse utiliserait les broches UART et I²C standard et les GPIO hors standard. Cela empêcherait d'autres MicroMods d'être totalement compatibles (peut-être qu'un ou deux des MicroMods Artemis ne pourraient pas piloter les relais de votre carte porteuse) mais c'est autorisé. À charge pour vous de gérer les compromis.
Nous avons rédigé un guide pour la création d'une carte de processeur MicroMod dont voici les principes directeurs :

- Connectez les fonctions correspondantes du microcontrôleur aux broches I2C, SPI, UART, USB, USB_HOST, CAN, SDIO et JTAG prévues sur connecteur MicroMod.
- Ensuite, A0/A1 sur le connecteur MicroMod doivent être attribuées aux broches du microcontrôleur qui sont exclusivement CAN (pas de capacité MLI).
- PWM0/PWM1 doivent être attribuées aux broches qui sont exclusivement MLI (pas de capacité CAN).
- D0/D1 doivent être attribuées aux broches qui sont exclusivement GPIO (pas de capacité CAN ou MLI).
- Les broches restantes doivent être attribuées aux Gx, les broches capables de CAN+MLI étant prioritaires (0, 1, 2, etc.).
- L'objectif est de garantir les fonctions MLI, CAN et E/S numériques sur ces broches spécifiques, alors que sur les Gx les fonction CAN/MLI ne sont pas garantie.
- Si le microcontrôleur manque d'une broche de fonction spécifique et qu'il reste du GPIO, elle peut être remplacée par du GPIO. Par exemple, CTS/RTS peut devenir GPIO si le microcontrôleur n'a pas de contrôle de flux.

Le tutoriel „ *Designing with MicroMod"* [21] vous guidera à travers les spécifications du processeur et de la carte porteuse MicroMod ainsi que les bases de l'intégration du dimensionnement MicroMod dans vos propres études de circuits imprimés.

Pour approfondir

Quelques ressources importantes au sujet de MicroMod :

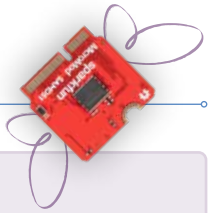
- Interface SparkFun MicroMod v1.0 - Brochage [8]
- Interface SparkFun MicroMod v1.0 - Description des broches [9].
- Les bibliothèques SparkFun Eagle Libraries contiennent des exemples d'emprises pour le connecteur M.2 et la hauteur des SMD [28].
- Fiche technique du connecteur MicroMod M.2[29].
- Fiche technique du plot soudable par refusion M2.5 [30].
- Page d'information sur MicroMod [31]
- Forums MicroMod [32]

Maintenant que vous êtes familiarisé avec les bases du MicroMod, vous trouverez dans les **encadrés** de cet article des **tutoriels** relatifs à MicroMod.

200684-02

VF Helmut Müller

Espace en ligne lié à cet article d'Elektor :
www.elektormagazine.com/esfe-en-micromod



Processeur SparkFun MicroMod / Cartes porteuses «Crystal Ball»

MicroMod

Voici une liste des cartes de processeurs MicroMod Sparkfun et des cartes porteuses correspondantes disponibles au moment de l'impression de ce magazine. Pour une vue d'ensemble à jour, voir sur **www.sparkfun.com/micromod.**

Cartes porteuses

ATP: Accès à toutes les broches de votre carte processeur.

Saisie et affichage : Un excellent moyen d'ajouter l'affichage des données et des entrées.

Acquisition de données : Une plateforme d'enregistrement de données personnalisable et de faible consommation.

L'apprentissage machine : Explorez-le sans ordinateur central ni connexion à l'internet.

Nouveau 1: En préparation

Nouveau 2: En préparation

Cartes processeurs

Artemis: choisissez le Cortex-M4F ARM à haut rendement avec BLE 5.0, jusqu'à 96 MHz et une consommation de seulement 6 µA/MHz (moins de 5 mW).

ESP32: Branchez la carte processeur puissante d'Espressif avec le double cœur Tensilica LX6, avec Bluetooth et WiFi intégrés.

SAMD51: Choisissez le SAMD51 pour une plate-forme de développement économique, puissante et facile à utiliser, avec le confort d'un chargeur d'amorçage UF2.

Nouveau 1: En préparation

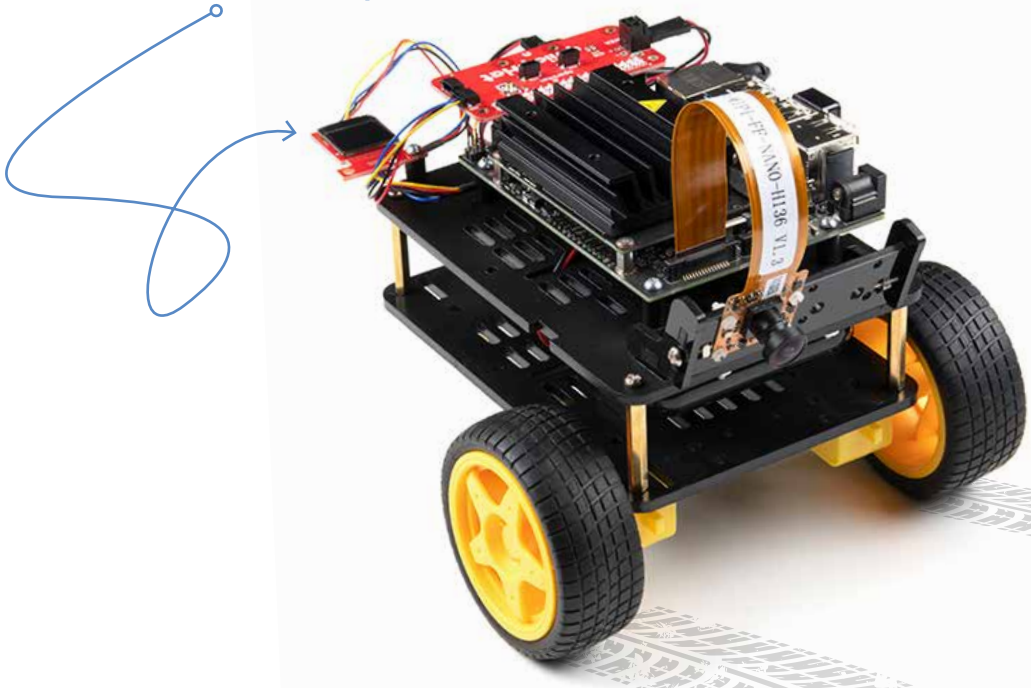
Nouveau 2: En préparation

LIENS

[1] Norme originale M.2: https://en.wikipedia.org/wiki/M.2
[2] Aperçu de l'écosystème MicroMod : https://www.sparkfun.com/micromod
[3] Interface périphérique série (SPI) : http://www.elektormagazine.com/esfe-en-micromod1
[4] Modulation de la largeur d'impulsion : https://learn.sparkfun.com/tutorials/pulse-width-modulation
[5] Alimentation en 3,3 V ou 5 V et niveaux logiques : https://learn.sparkfun.com/tutorials/logic-levels
[6] Une introduction à I2C : https://learn.sparkfun.com/tutorials/i2c
[7] Connecteur M.2 et spécification : https://en.wikipedia.org/wiki/M.2
[8] Interface SparkFun MicroMod v1.0 - Brochage : https://bit.ly/3p96pJe
[9] Interface SparkFun MicroMod v1.0 - Descriptions des broches : https://bit.ly/3asRCF1
[10] Connecteur TE 2199230-4 : https://www.findchips.com/search/2199230-4
[11] Kit MicroMod DIY : https://www.sparkfun.com/products/16549
[12] Brochage de MicroMod : https://bit.ly/3rhRvlS
[13] Description des broches de MicroMod : https://bit.ly/34wu4eW
[14] Mini-tournevis réversible SparkFun : https://www.sparkfun.com/products/9146
[15] Jeu de tournevis de poche : https://www.sparkfun.com/products/12891
[16] Installation des pilotes CH340 : https://bit.ly/3p4V50C
[17] Ajoutez votre étude comme étude de référence: https://www.sparkfun.com/static/contact
[18] Carte de processeur MicroMod ESP32 : https://www.elektormagazine.com/esfe-en-micromod2
[19] Carte de processeur MicroMod SAMD51 : https://www.elektormagazine.com/esfe-en-micromod3
[20] Carte de processeur MicroMod Artemis : https://github.com/sparkfun/MicroMod_Artemis_Processor
[21] Concevoir avec MicroMod : https://learn.sparkfun.com/tutorials/designing-with-micromod
[22] MicroMod carte porteuse toutes broches (All The Pins, ATP) : https://github.com/sparkfun/MicroMod_ATP_Carrier_Board
[23] Carte porteuse d'acquisition de données MicroMod : https://github.com/sparkfun/MicroMod_Data_Logging_Carrier
[24] Carte porteuse d'apprentissage machine MicroMod : https://github.com/sparkfun/MicroMod_Machine_Learning_Carrier
[25] Carte porteuse d'entrée et d'affichage MicroMod : https://github.com/sparkfun/MicroMod_Input_and_Display_Carrier
[26] Kit MicroMod DIY : https://www.sparkfun.com/products/16549
[27] TE 2199230-4 : https://www.findchips.com/search/2199230-4
[28] Bibliothèques Eagle, emprise M.2 et hauteur CMS : https://github.com/sparkfun/SparkFun-Eagle-Libraries
[29] Fiche technique du connecteur M.2 MicroMod : https://bit.ly/3nCZM1B
[30] Fiche technique du plot soudable par refusion M2.5 : https://www.elektormagazine.com/esfe-en-micromod4
[31] Page d'information sur MicroMod: https://www.sparkfun.com/micromod
[32] Forums MicroMod : https://forum.

Coup de baguette magique sur le **JetBot** de SparkFun

Ou comment j'ai perfectionné mon JetBot, animé par la **carte NVIDIA Jetson Nano**



Derek Runberg (États-Unis)

Les kits électroniques sont bon marché et permettent d'aller droit au but. Surtout s'il s'agit de robots : le châssis est préassemblé, un exemple de programme et un guide détaillé sont fournis et en peu de temps votre application fonctionne. Un bon kit vous évitera un apprentissage laborieux. Vous concrétiserez rapidement votre idée initiale. L'excellence est à votre portée!

Le kit d'IA *JetBot* de *SparkFun* donne l'exemple. *Nvidia* a lancé le projet JetBot et nous y avons collaboré pour offrir à ceux qui n'ont pas d'accès à l'impression 3D la possibilité d'acquérir une version prête à monter. Ce kit JetBot comprend tout ce qu'il faut pour construire un robot expérimental, châssis et systèmes mécaniques inclus. Il comprend également les composants nécessaires pour débuter immédiatement en vision par caméra (*CV*) et apprentissage machine (*ML*) grâce à la carte *Jetson Nano*™ de NVIDIA® – rien de superflu, tout y est. Le kit d'IA JetBot de SparkFun supprime le délai d'impression 3D et la corvée du prototypage. Le gain est considérable.

On me questionne souvent sur l'extensibilité du JetBot : „est-il possible de commencer avec lui et le faire évoluer vers un *système robotique* ? La réponse est „oui !". Le JetBot est conçu pour l'expérimentation avec, hormis la caméra, des capteurs rudimentaires. Il est cependant adaptable à différents châssis, moteurs et caméras… Au lieu de me tenir à cette promesse de Gascon, je me suis retroussé les manches pour étudier de près comment procéder pour étendre les capacités de mon JetBot. Cet article décrit ma démarche et les pièges à éviter pour obtenir quelque chose de plus amusant.

Cinématique en temps réel (RTK) et à l'estime (Dead Reckoning)

Les GPS à cinématique en temps réel (RTK) reçoivent les signaux habituels des systèmes mondiaux de navigation par satellite (GNSS). Un récepteur RTK reçoit en plus un flux de messages de correction en temps réel *RTCM* (*Real Time Correction Messages*) qui lui permet d'affiner sa position avec une précision de 1 cm en temps réel. La cadence de correction varie d'un récepteur à l'autre, mais la plupart d'entre eux travaillent à 1 Hz, certains atteignent 20 Hz.

La navigation dans une ville dense, un court tunnel ou un parking peut affaiblir le signal, voire l'occulter. En l'absence de signal GNSS, la navigation à l'estime consiste à déterminer la position instantanée à partir des dernières données connues de position, de cap et de vitesse. L'estimation exploite les données fournies par la centrale inertielle 3D (IMU) et les données de déplacement du véhicule (par ex. capteurs de rotation des roues et odomètres) pour actualiser sa position.

Cinématique temps réel :
https://fr.wikipedia.org/wiki/Cinématique_temps_réel

Planification

Pour avancer avec mon JetBot, j'ai d'abord défini ce dont je disposais pour le modifier. Avec la Jetson Nano NVIDIA, il y a un impératif : pour intégrer les cartes de notre écosystème *Qwiic*, il faut que *Python* les prenne en charge, je n'ai guère envie de me coltiner des primitives I²C en Python pour l'instant.
Grâce au dépôt *GitHub* de SparkFun pour la bibliothèque *Qwiic_Py*, j'ai pu rapidement trouver les articles disponibles. À partir des différents répertoires de pilotes, j'ai compté 20 cartes intégrables. Le JetBot utilise déjà deux de ces 20 cartes : le pilote de moteur Qwiic et l'afficheur *OLED* Qwiic.
J'ai scruté la liste à la recherche des cartes les plus utiles pour le transformer en robot d'avant-garde. Après mûre réflexion, j'ai choisi trois cartes susceptibles de libérer un potentiel maximal (**fig. 1**).

1. Carte d'GPS-RTK-SMA ZED-F9P

À mon avis, un robot haut de gamme ne peut se passer d'un *GPS*, sinon comment faire en extérieur loin du bureau. Le module *ZED-F9P* de *u-blox* est récent et performant. Il exploite tous les avantages de cette technologie, notamment la cinématique en temps réel (RTK) pour obtenir une précision centimétrique si j'utilise une station de référence. Le GPS sera utilisé à la fois comme système de navigation et comme moyen de télésurveillance. En combinant la vision par ordinateur et le GPS, il serait possible de cartographier différents objets : rochers, plantes ou même personnes.

2. Carte Auto Phat de SparkFun

L'*Auto Phat* est une carte d'extension (partielle, d'où le p de *Phat*) *Raspberry Pi* compatible avec la carte NVIDIA Jetson. Cette carte et le pilote de moteur Qwiic livré avec le JetBot de série exploitent le même pilote de moteur, mais l'Auto Phat apporte d'autres fonctions et remplace avantageusement une série de cartes Qwiic. L'Auto Phat comprend le contrôleur de moteur, des entrées encodeurs de moteur, une *centrale inertielle (IMU)* et des sorties de servocommande. En remplaçant le pilote de moteur simple par l'Auto Phat, non seulement j'améliore la précision de la commande des moteurs, mais je peux piloter des servos et j'ai une IMU en prime.

3. VL53L1X Télémètre à temps de vol

Le JetBot est livré avec un capteur : une caméra. La vision par ordinateur est puissante. Ce que le JetBot peut en faire avec un peu d'apprentissage est étonnant. Avec quelques capteurs supplémentaires, le robot peut identifier des objets et mesurer à quelle distance ils se trouvent. Ou mieux encore, détecter des objets d'intérêt en dehors du champ de vision de la caméra.

Il existe de nombreuses possibilités pour la télédétection. J'ai opté pour un capteur à temps de vol (*ToF*) peu onéreux, à faible consommation avec une portée convenable pour la taille du JetBot. J'ai élu le *VL53L1X* pour sa portée de quatre mètres et son champ de vision relativement étroit. J'espérais pouvoir l'aligner avec le centre de vision du robot et obtenir une distance approximative de l'objet en vue de la caméra - plus facile à dire qu'à faire avec un objectif à très grand angle.

Intégration du matériel

J'ai commencé par l'Auto Phat, car elle nécessitait d'intervenir sur les systèmes existants et la construction de mon JetBot de série. Elle remplace le Phat Qwiic simple et le pilote de moteur Qwiic, il fallait donc les retirer du JetBot. De plus, comme l'Auto Phat a des entrées d'encodeurs, j'ai remplacé les motoréducteurs d'origine par des modèles avec encodeurs (**fig. 2**).
Ce projet était relativement simple. Le plus difficile a été d'acheminer efficacement les câbles du moteur vers la Phat. Cependant, leurs fils sont assez longs, il a donc été possible de les faire passer à travers le châssis jusqu'aux bornes à vis de la Phat, et je les ai branchés selon le guide de la Phat.
Une fois la Phat remplacée, il ne me restait plus qu'à rebrancher le câble Qwiic de l'afficheur OLED ainsi que le câble USB de la batterie (dans le port USB de la Phat), et tout était intégré !

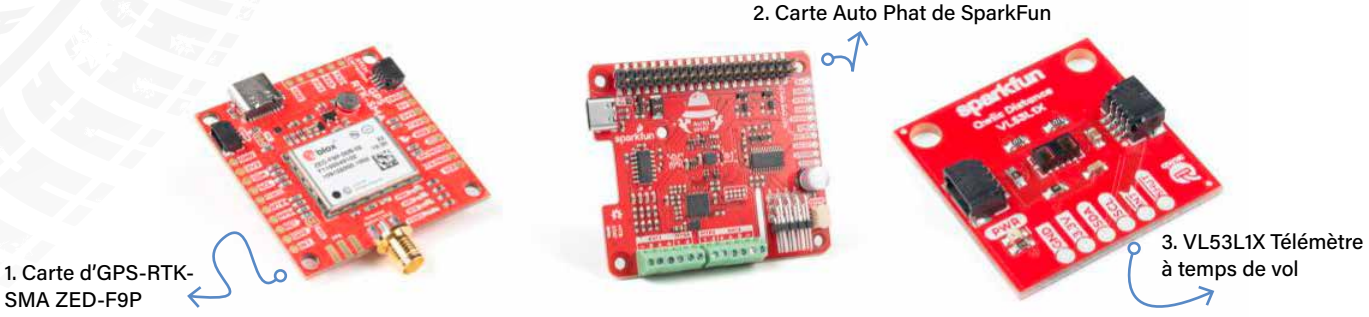


Figure 1. Les trois cartes SparkFun retenues pour perfectionner le JetBot.

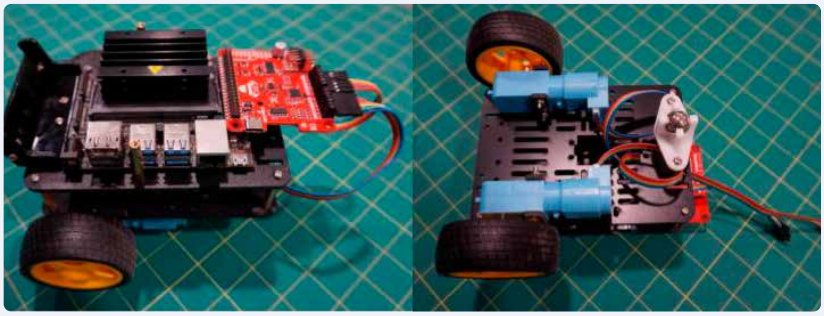


Figure 2. Sur le JetBot modifié, l'Auto Phat remplace le Qwiic Phat simple et le pilote de moteur Qwiic.

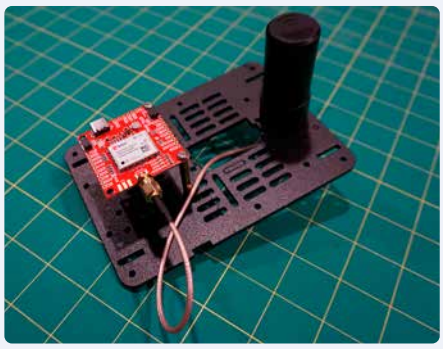


Figure 3. La carte GPS montée sur un châssis séparé, installée au-dessus de la Jetson Nano et de la monture de la caméra.

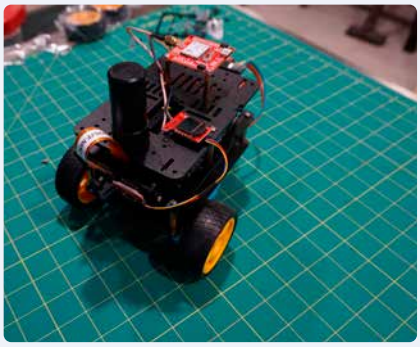


Figure 4. Vue arrière du JetBot modifié, avec plaque de montage supplémentaire pour fixer la carte GPS (sur des entretoises) et la carte OLED.

Figure 5. Télémètre ToF compatible Qwiic et module de caméra montés sur un support de fixation au JetBot.

Figure 6. Aspect final du JetBot perfectionné.

Ce fut ensuite le tour du GPS. La carte elle-même est un peu plus grande que notre carte Qwiic standard de 2,5 × 2,5 cm, il n'était donc pas possible de la monter sur l'Auto Phat. Où monter une antenne sur le JetBot déjà très encombré ? Un bout de châssis trouvé dans ma réserve de pièces a fait l'affaire, je l'ai monté au-dessus de la Jetson Nano et du support de caméra (**fig. 3**). Cela m'a ouvert un espace pour monter ensuite le GPS et mon capteur télémétrique (**fig. 4**). Le GPS s'est bien installé dans les fentes, et pour monter l'extension SMA j'ai foré un trou de taille convenable dans le châssis. Comme elle communique par liaison UART, la carte GPS fonctionne à merveille sur un ordinateur monocarte. Je pouvais la brancher à un des UART du connecteur, mais j'ai préféré utiliser le port USB, déjà configuré en UART d'origine. Inutile de chercher midi à 14 h. Le dernier ajout prévu au JetBot était le capteur de distance à temps de vol. Par bonheur, ce capteur a l'encombrement standard Qwiic : 2,5 × 2,5 cm. Je l'ai monté dans le même plan, à côté de la caméra en utilisant un support simple de mon cru et un carré de *bande Velcro* (**fig. 5**). Le Velcro offre une certaine souplesse de montage du capteur sans perçage. Il ne me restait plus qu'à trouver la longueur convenable de câble Qwiic pour relier le capteur à la carte Qwiic la plus proche de la chaîne. Avec ma configuration, c'était la carte d'affichage OLED sur le châssis ajouté. La **fig. 6** montre le résultat.

Intégration du logiciel

Il y a plusieurs façons de gérer l'intégration du logiciel à ce robot, mais la plus simple est d'utiliser l'image personnalisée pour le SparkFun JetBot. Celle-ci comprend le pilote de moteur Qwiic de SparkFun, ainsi que des exemples choisis par NVIDIA pour le JetBot et qui s'appuient sur *Jupyter Notebooks* pour accéder au système de fichiers et exécuter des scripts Python à distance.

À partir de cela, vous pouvez alors installer le paquet Python *Qwiic_py* sur votre JetBot et commencer à tester le programme d'exemple pour chaque carte. En ligne de commande sur le terminal et avec `pip` c'est rapide. Ouvrez une fenêtre de terminal sur votre NVIDIA Jetson Nano depuis le bureau ou Jupyter Notebooks, et tapez :

```
sudo pip install sparkfun-qwiic
```

Cette commande télécharge et installe les pilotes pour toutes les cartes Qwiic actuellement prises en charge par l'écosystème. Pour finir, il faut installer *PySerial*, la méthode de base de communication par *USB* entre la Jetson Nano et la carte GPS F9P. Pour installer le paquet PySerial, tapez :

```
sudo pip install py_serial
```

Les deux ensembles logiciels installés, nous allons exécuter un programme d'exemple pour chaque capteur ajouté. Le dépôt Github Qwiic_py fournit des exemples de programmes Python

pour chacune de nos cartes Qwiic prises en charge [1]. Téléchargez ou copiez-collez les exemples dans un fichier Python et exécutez-les depuis la ligne de commande en tapant :

```
python3 exemple.py
```

Si vous exécutez un script en rapport avec le GPS connecté via USB à votre Jetson Nano, vous devrez utiliser `sudo` dans votre commande, car cela lui confère la permission de lire et écrire des messages UART via USB :

```
sudo python3 exampleGPS.py
```

Note : si vous travaillez à partir de Jupyter Notebooks et que vous voulez utiliser le GPS ou exécuter un script nécessitant une connexion UART, vous devrez modifier les permissions de votre port UART. Vous pouvez le faire à l'aide de la commande suivante :

```
sudo chmod 660 /dev/ttyAMC0
```

Si vous êtes curieux, j'ai déposé sur GitHub quelques exemples de programmes Python créés en utilisant le JetBot modifié via Jupyter Notebook. Vous pouvez les trouver et les télécharger sur votre propre JetBot [2]. Chaque Jupyter Notebook fournit une explication des extraits de code et permet de les exécuter directement, sans ligne de commande. Si vous ne connaissez pas encore Jupyter Notebooks, consultez le tutoriel rapide [3].

Conclusion

Nous n'avons fait qu'effleurer le sujet de la mise à niveau de votre JetBot. Il y a une infinité de capteurs et d'actionneurs faciles à intégrer et à utiliser avec les ordinateurs monocartes. Les capacités d'apprentissage automatique et d'intelligence artificielle de la Jetson Nano de NVIDIA en font un outil de premier plan pour conférer des performances hors norme aux plateformes robotiques les plus simples.

Moyennant quelques accessoires, j'ai mis à niveau le JetBot lui-même, mais pourquoi s'arrêter là ! Si germe en vous l'idée de commander avec la Jetson Nano une plateforme robotique de votre choix, osez, le monde bientôt vous appartiendra !
L'an passé, j'ai porté JetBot sur la plateforme robotique *RVR* de *Sphero* avec quelques composants du catalogue SparkFun. Un peu d'usinage, ponçage et perçage et le RVR devient plus intelligent et tire parti du *Machine Learning* de la Jetson Nano. Vous pouvez trouver mon tutoriel étape par étape ici [4].

La Jetson Nano de NVIDIA et la plateforme JetBot sont parmi les kits et produits robotiques les plus excitants que j'ai utilisés depuis longtemps. Ils marient admirablement technologie de pointe, accessibilité et souplesse. Si vous voulez les bidouiller, ne vous gênez pas. Je ne me lasse pas de perfectionner ce robot, tout en développant mes compétences en apprentissage automatique et en intelligence artificielle.

(200689 – VF : Yves Georges)

SparkFun & NVIDIA

L'apprentissage automatique (*Machine Learning* ou ML) est nouveau pour nous tous ! La coopération entre SparkFun et Nvidia a débouché sur des innovations concrètes accessibles à nos clients. Nous proposons depuis plusieurs années des robots doués d'auto-apprentissage sous forme de kits autour de la Jetson Nano et de tutoriels en ligne gratuits. NVIDIA repousse constamment les limites du ML tandis que SparkFun s'active à simplifier ces techniques.

LIENS

[1] Exemple de programme Python pour les cartes Qwiic : https://bit.ly/2KNPf5k
[2] Exemples de programmes Python dans Jupyter Notebook : https://bit.ly/3a9RbO7
[3] Tutoriel de Jupyter Notebook : https://bit.ly/36eanJA
[4] Tutoriel sur l'univers du RVR de Sphero : https://bit.ly/3c7Non3

Accessoires

Vous trouverez chez Elektor et Sparkfun les accessoires mentionnés dans cet article.

- Kit JetBot AI Kit v2.1 de SparkFun basé sur une Jetson Nano
  www.elektormagazine.com/esfe-en-jetbot1
- Carte GPS-RTK-SMA de SparkFun - ZED-F9P (Qwiic)
  www.elektormagazine.com/esfe-en-jetbot2
- SparkFun Auto pHAT pour Raspberry Pi
  www.elektormagazine.com/esfe-en-jetbot3
- Carte télémètre SparkFun - 4 m, VL53L1X (Qwiic)
  www.elektormagazine.com/esfe-en-jetbot4

Programmation d'un FPGA

Figure 1 : Carte de développement FPGA d'Alchitry Au.

Resources matérielles utiles

Si vous recherchez les produits utilisés dans cet article, vous les trouverez chez Elektor et chez SparkFun :

Alchitry Au FPGA Development Board

www.elektormagazine.fr/esfe-en-fpga1

Justin Rajewski (Alchitry)

Dans ce tutoriel, nous aborderons les notions de base des étapes de création de circuits FPGA et les blocs élémentaires à utiliser. De prime abord, un FPGA (*Field-Programmable Gate Array* = réseau d'opérateurs logiques programmables in situ) c'est assez intimidant pour un débutant, mais armé de logiciels accessibles, de matériel polyvalent et de quelques exemples judicieux, vous vous en sortirez sans peine.

Avant de nous lancer, rappelons qu'un FPGA ne se programme pas. [1] C'est un abus de langage qui s'explique par la similitude de la démarche : on écrit un texte qui, une fois transformé en binaire, est chargé dans le FPGA. Vous n'écrivez pas un programme, vous créez un circuit. Un langage de programmation ne peut pas servir à créer des circuits ; il s'agit en réalité d'un langage de description de circuits (*hardware description langage* ou HDL). [2]

Il serait fastidieux de dessiner le schéma d'une architecture complexe, et nous nous contentons de décrire le comportement attendu du circuit, ce sont ensuite les outils qui se chargent de trouver une solution concrète. Lors de la conception d'un projet FPGA, il faut garder à l'esprit que nous décrivons le matériel et que tout ce que nous écrivons sera transposé en un circuit physique. Il est possible de décrire des circuits impossibles à réaliser ou de décrire quelque chose qui semble simple, mais dont la réalisation nécessite une énorme quantité de ressources. C'est pourquoi il est essentiel de savoir comment pourra être réalisé le circuit que nous essayons de décrire.
Pour suivre ce tutoriel, il suffit d'une carte FPGA Alchitry Au (**fig. 1**) et d'un câble USB A-C réversible.

Structure d'un projet

En général, un HDL s'articule autour de l'idée de module. Un module est un bloc de circuit doté d'un certain nombre d'entrées et de sorties et contenant une logique pour les relier entre elles. À l'instar d'un programme que l'on décompose en fonctions, un module peut contenir des sous-modules. Il peut aussi être autonome.
Bien qu'il soit possible de regrouper un projet entier dans un seul module, il est préférable d'utiliser des modules plus petits pour réaliser chaque élément du projet. En décomposant notre projet en modules,

nous simplifions chaque partie sur laquelle nous travaillons à un moment donné. Certains modules seront conçus pour effectuer des tâches communes et seront utilisés à maintes reprises.
Lorsque nous débutons la conception, il est souvent utile de dessiner un schéma fonctionnel montrant les différents modules et leur interconnexion. Cela permet de définir l'étendue de notre projet et de le décomposer logiquement.

Lucid

Tout au long de ce tutoriel, nous utiliserons Lucid [3]. Lucid est un HDL conçu spécifiquement pour les FPGA. Il évite beaucoup de pièges (et croyez-moi, ils sont nombreux) dans lesquels il est facile de tomber avec d'autres HDL comme Verilog et VHDL.
Lucid est un outil extraordinaire pour débuter avec les FPGA. Souvent des personnes craignant des limitations avec Lucid ou voulant simplement passer à Verilog ou VHDL pour une autre raison me contactent. Si vous voulez bien débuter, faites-le avec Lucid. Il vous apprendra les bases de la conception sur matériel FPGA avant que vous n'exploriez d'autres HDL plus lourds. Si vous souhaitez le faire par la suite, ce n'est pas très difficile. Lucid est basé en gros sur Verilog et Alchitry Labs convertira Lucid en Verilog pour vous si vous voulez utiliser ailleurs vos modules hypersophistiqués.
Plongeons dans les entrailles d'un module.

Anatomie d'un module

Lorsque nous créons un projet, nous obtenons un module de niveau supérieur (= top level). C'est le module dont les entrées et sorties sont des entrées et sorties réelles sur les broches du FPGA. Pour tout projet Alchitry, il s'agit soit de *cu_top.luc*, soit de *au_top.luc* selon que nous utilisons la carte Cu ou Au. Les modules de niveau supérieur initiaux sont pratiquement identiques pour les deux cartes (**listage 1**).
La section de tête du module contient la déclaration des ports. C'est là que les entrées et sorties du module sont déclarées. Dans ce cas, comme il s'agit du module *top level*, il s'agit de signaux présents sur la carte elle-même (**listage 2**).
Vous avez peut-être remarqué que l'entrée *led est* suivie d'un numéro entre crochets. Il ne s'agit donc pas d'une, mais de huit entrées distinctes rassemblées en un tableau. Nous reviendrons sur la syntaxe des tableaux.
Un module peut comporter une liste de paramètres de personnalisation. Elle est omise ici, car inutile sur un module *top level* puisque les paramètres sont passés par le module parent qui l'instancie. Le terme d'*instanciation* est utilisé pour désigner le moment où un module (ou une autre ressource) est ajouté(e) à un projet. Il signifie qu'une *instance* de ce module (ou de cette autre ressource) est créée.
Dans un programme, l'appel à une fonction lance un code exécutable existant en un seul exemplaire, indépendamment du nombre de fois que la fonction est appelée. Au contraire, dans un projet FPGA, à chaque instanciation d'un module, l'ensemble du circuit qui le compose est dupliqué. Si, en tant que concepteurs, nous souhaitons réutiliser les mêmes ressources pour plusieurs tâches, il nous appartient de trouver comment jongler avec cela. À propos de l'instanciation, notons que nous instancions généralement **tout** ce dont le module a besoin **juste après** la déclaration de port.
La première ligne consiste à déclarer un signal en utilisant le mot-clé *sig* :

```
  sig rst;                    // reset signal
```

Les signaux ne sont pas des valeurs stockées en mémoire. Il faut les considérer comme des fils. Un fil peut avoir une valeur, ce n'est en fait qu'une connexion d'un point à un autre.

Listage 1.

```
module au_top (
   input clk,      // 100MHz clock
   input rst_n,    // reset button (active low)
   output led [8], // 8 user controllable LEDs
   input usb_rx,   // USB->Serial input
   output usb_tx   // USB->Serial output
  ) {

  sig rst;          // reset signal

 .clk(clk) {
   // The reset conditioner is used to
   // synchronize the reset signal to
   // the FPGA clock. This ensures the
   // entire FPGA comes out of reset at
   // the same time.
   reset_conditioner reset_cond;
  }

  always {
   reset_cond.in = ~rst_n; // input raw inverted
                           // reset signal
   rst = reset_cond.out;   // conditioned reset
    led = 8h00;            // turn LEDs off
    usb_tx = usb_rx;       // echo the serial data
  }
}
```

Listage 2.

```
    input clk,          // 100MHz clock
    input rst_n,        // reset button (active low)
    output led [8],     // 8 user controllable LEDs
    input usb_rx,       // USB->Serial input
    output usb_tx       // USB->Serial output
```

Dans ce projet, nous utilisons le signal *rst* comme paramètre de sortie du module *reset_conditioner.* Les deux lignes ci-dessous réalisent l'instanciation de ce module :

```
.clk(clk) {
    // The reset conditioner is used to synchronize
// the reset signal to the FPGA clock. This
// ensures the entire FPGA comes out of reset
// at the same time.
    reset_conditioner reset_cond;

  }
```

Pour instancier un élément, il suffit d'utiliser le nom de la ressource suivi du nom de cette instance particulière. Ainsi, la ligne `reset_conditioner reset_cond;` crée une instance du module *reset_conditioner* nommée *reset_cond*. Le bloc dans lequel cette instanciation est enveloppée s'appelle bloc de connexion. Il permet de connecter une entrée ou un paramètre (ayant un nom donné) de plusieurs modules au même signal.

Dans notre cas, nous connectons l'entrée *clk* au signal *clk* (qui est une entrée de notre module). La syntaxe est *.port(signal)* où *port est le* nom de l'entrée sur le module en cours d'instanciation et *signal* est le signal à y connecter.
Le module *reset_conditioner* a une entrée nommée *clk,* cette entrée est donc directement reliée au signal *clk de* notre module. Nous pouvons également la relier directement au module sur la ligne d'instanciation comme ceci :
reset_conditioner reset_cond(.clk(clk));
Nous ne le faisons pas ici parce que l'entrée *clk* fait partie de presque tous les modules et qu'il est pratique d'avoir un bloc comme celui-ci pour pouvoir simplement ajouter les autres instanciations qui doivent être connectées à *clk*.
Presque tous les modules disposent d'une entrée *clk* et souvent d'une entrée *rst* pour la réinitialisation. Nous verrons plus tard à quoi servent exactement l'horloge et l'initialisation. Nous pouvons ajouter plusieurs connexions au même bloc de connexion :
.clk(clk), .rst(rst) { ... }
Nous pouvons également imbriquer les blocs et comme tous les blocs qui utilisent *clk* n'utilisent pas *rst - en* général, nous verrons quelque chose de ce genre au début d'un module :

```
.clk(clk) {
    // clk only instantiations
    .rst(rst) {
        // rst and clk instantiations
    }

}
```

Les blocs *always*

C'est l'essence même du module : les blocs *always* sont les endroits où nous décrivons toute la logique qui s'y exprime. Ils renferment ce qu'on appelle de la logique combinatoire. Tout circuit numérique dont la sortie est exclusivement fonction de ses entrées est de la logique combinatoire. Il n'a ni état ni mémoire internes. Un additionneur est un bon exemple de logique combinatoire. La sortie est exclusivement déterminée par les deux nombres présents en entrée. Ni les chiffres entrés auparavant, ni le nombre de changements antérieurs n'ont d'importance. La sortie est toujours fonction des entrées en cours.
À l'intérieur du bloc "*always*", nous écrivons des instructions. Il y en a quatre types principaux :

- l'affectation ;
- l'instruction *if* (instruction conditionnelle) ;
- l'instruction *case* (instruction conditionnelle) ;
- l'instruction *for* (boucle).

Affectation

L'affectation est de loin la plus courante. À gauche, nous trouvons un signal, suivi d'un signe égal et à droite une expression.

```
signal = expression;
```

La puissance de cette instruction vient de l'expression. Pour celle-ci, nous disposons d'une quantité d'opérateurs différents pour manipuler les bits. Il s'agit notamment de certains opérateurs mathématiques comme +, –, et *. Il faut noter que pour la division, / ne peut pas être utilisé si l'expression est dynamique. En effet, la division est trop compliquée pour que les outils déterminent une valeur par défaut raisonnable à notre place. La division est toujours possible, elle demande juste un peu plus d'effort et doit être planifiée.

Instruction *if*

L'instruction *if* obéit à la syntaxe habituelle :

```
if (expr) { … } else { … }
```

Si l'expression qui suit le "*if*" est vraie (non nulle), alors la 1ère série de lignes est validée. Si elle est fausse (zéro), les lignes du bloc "else" sont validées. La partie "else" est facultative. Remarque : j'ai écrit "validée" et non "exécutée". On tombe facilement dans ce piège lorsque nous avons des instructions *if* et des boucles *for* à introduire dans la structure du programme. Les *if* sont le plus souvent réalisés par un multiplexeur matériel. Il suffit de sélectionner l'une des deux entrées selon la valeur d'une expression.
Lorsque nous attribuons une valeur à un signal dans un bloc *always* , il faut qu'en toute circonstance une valeur lui soit attribuée. En général, cela signifie que si nous attribuons une valeur à quelque chose dans une instruction *if*, la partie *else* de l'instruction doit avoir son pendant. La seule exception à cette règle est l'entrée *d* d'un type *dff* ou *fsm*. Nous y reviendrons.
Une autre façon de s'assurer que nous attribuons toujours une valeur est de commencer le bloc *always* par quelques valeurs par défaut raisonnables. Examinons le pseudo-code suivant :

```
led = 0;
if (button_pressed)
    led = 1;
```

Lorsque le bouton n'est pas enfoncé, *led* a la valeur 0. Cependant, que se passe-t-il si le bouton est enfoncé ? *led* prend-il la valeur 0 puis est-il mis à jour avec une valeur de 1 ? Eh bien non. Lorsque le bouton est pressé, *led* a toujours la valeur 1. Les affectations en queue d'un bloc *always* ont la priorité sur les affectations de tête. Cela revient à dire que le bloc *always* est *toujours* évalué et cela, *instantanément*.
Maintenant, imaginons que nous n'ayons pas cette valeur par défaut avant le *if*. Quelle valeur aurait *led* bouton non enfoncé ? Il est tentant de penser qu'il conserverait sa valeur précédente, mais souvenons-nous que les signaux ne peuvent pas stocker de valeurs. Ils sont juste comme des fils reliant deux choses entre elles.
Une telle valeur par défaut de 0, pourrait être réalisée avec un circuit de type multiplexeur. Dans ce cas

Premier FPGA : s'amuser avec la MLI

Lorsque vous allumez une carte Alchitry Au [5] ou Alchitry Cu [6] pour la première fois, la configuration FPGA par défaut crée sur les LED un effet de vague amusant. Dans le tutoriel, nous passons en revue les différentes étapes permettant de parvenir à un tel résultat. C'est un très bon guide pour aborder un circuit et se perfectionner avec, compte tenu du fait qu'*in fine* nous travaillons sur du matériel et non du logiciel.

Pour en savoir plus, consultez le tutoriel complet :
https://learn.sparkfun.com/tutorials/first-fpga-project---getting-fancy-with-pwm

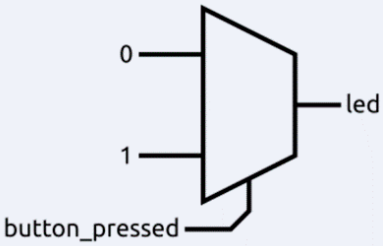


Figure 2 : Bouton de type FPGA pour le contrôle de l'allumage et de l'extinction d'une LED.

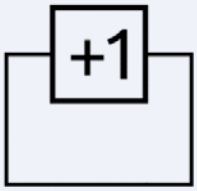


Figure 3 : Fonction d'addition.

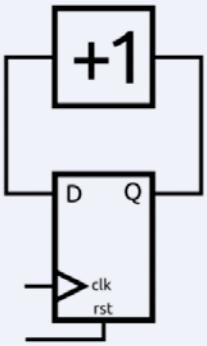


Figure 6 : DFF et compteur combinés.

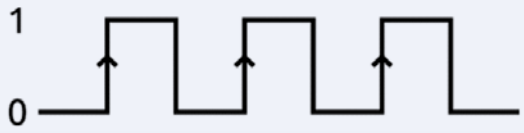


Figure 4 : Signal d'horloge dont les fronts montants sont marqués par des flèches.

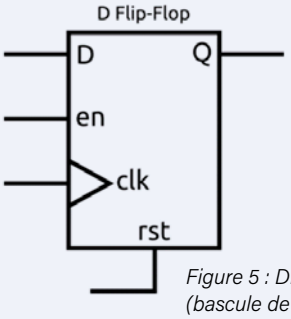


Figure 5 : DFF (bascule de type D).

trivial, cela pourrait être simplifié en se contentant de connecter les signaux *led* et *bouton pressé* ensemble (**fig. 2**).

Instruction *case*

L'instruction *case* obéit à la syntaxe suivante :

```
case(expr) {
    value: statement;
    value: statement;
    default: statement;
}
```

Elle fonctionne exactement comme l'instruction *if*. Elle ne sert qu'à réaliser plusieurs branchements à l'aide d'une seule expression. Chaque partie value de l'instruction doit être une constante. La branche optionnelle défaut est un fourre-tout.

Contrairement à l'instruction case d'un programme informatique, *case* n'offre ici aucun gain de vitesse par rapport à une série d'instructions *if*. Elle améliore la clarté et l'ergonomie du code, c'est tout.

Boucle *for*

Avec Lucid, la boucle *for* adopte la même syntaxe que le C ou le Java, mais il y a quelques restrictions :

```
for (init; eval; increment) { … }
```

La boucle *for* est généralement utilisée avec un type *var* qui sert à stocker des valeurs utilisées dans la description, mais n'apparaissant pas directement dans le circuit. La principale limitation matérielle de la boucle *for* est qu'elle doit avoir un nombre d'itérations défini par une constante. Les outils doivent en effet pouvoir dérouler la boucle. Une boucle *for* consiste à cloner la partie de code correspondante autant de fois que nécessaire. C'est beaucoup plus lisible. En l'absence d'une bonne raison, mieux vaut les éviter. Avec des boucles *for,* on se retrouve facilement avec un grand circuit très lent.

Nombres

Dans Lucid, il y a diverses façons de définir une constante numérique. Le plus simple est de taper le nombre par ex. 14. Lorsque nous voyons un nombre isolé, il est en décimal (en base $_{10}$) et le nombre de bits utilisés pour le représenter est le minimum requis dans un format non signé, à moins qu'il ne soit négatif.

Pour mieux contrôler le nombre de bits utilisés, nous pouvons ajouter un préfixe *xd* où *x* est le nombre de bits à utiliser. Par exemple, *8d14* est la valeur décimale 14 représentée sur 8 bits. En remplaçant le *d* par *h* ou *b* nous exprimerons respectivement le nombre en hexadécimal (base $_{16}$) ou en binaire (base $_{2}$). Avec ces deux formats, nous pouvons spécifier le nombre de bits à utiliser avant le h ou le b. Si nous omettons le nombre de bits, les nombres hexadécimaux utilisent par défaut 4 bits par chiffre. Par exemple, *h08* utilise 8 bits puisqu'il y a deux chiffres alors que la valeur 8 pourrait être représentée avec seulement 4 bits. Pour le binaire, le nombre de bits est simplement le nombre de chiffres lorsqu'il n'est pas explicitement spécifié. Ainsi, *b101001 a une* largeur de 6 bits. Si un nombre décimal est écrit avec un *d,* mais que le nombre de bits est omis, il se comporte comme si le *d* était également omis.

Arrays (tableaux)

Nous rencontrerons de nombreux signaux multibits comme l'entrée *led* de notre module de niveau supérieur. Les bits d'un tableau peuvent être indicés individuellement en utilisant la syntaxe *signal[bit]* où *bit* est une expression. S'il s'agit d'une valeur dynamique, nous devons nous assurer que la valeur sera toujours dans les limites du tableau.

L'accès à des sous-ensembles de bits se fait en utilisant la syntaxe de tableau : *[max:min]*. Ici, la plage de bits de *min* à *max,* (tous deux inclus) est sélectionnée. Avec cette syntaxe, les deux valeurs doivent être des constantes.

Pour sélectionner dynamiquement un sous-ensemble de bits, nous pouvons utiliser la syntaxe *[start+:width]*. Ici, *start* est le bit inférieur à sélectionner et *width* est le nombre total de bits à sélectionner à partir du bit de départ (*start*) inclus. Dans cette syntaxe, seule la *largeur* doit être une constante. Nous pouvons également utiliser la variante : *[start-:width]*. Dans cette syntaxe, *start* est le bit supérieur de la sélection et non le bit inférieur.

Dans Lucid, les tableaux peuvent être multidimensionnels. Il suffit d'ajouter les dimensions voulues dans la déclaration :

```
sig my_array[dim1][dim2][dim3];
```

Les dimensions d'un tableau doivent être déclarées avec des valeurs constantes. Nous pouvons ensuite indicer le tableau avec des sélecteurs comme ci-dessus. Notons que nous pouvons utiliser les sélections de sous-tableaux seulement comme dernier sélecteur.

Logique séquentielle et DFF

Ici cela devient vraiment intéressant. La logique combinatoire est très importante, mais un système sans état ni mémoire est assez limité. Alors comment créer une mémoire ? En gros, nous avons juste besoin d'une sorte de boucle de rétroaction. Pour créer un compteur, il suffit d'ajouter 1 au résultat de la dernière addition. Le problème est de savoir comment contrôler cette boucle. À première vue, cela peut sembler trivial, mais y regarder de plus près revient à ouvrir la boîte de Pandore. En effet, nous verrons que rien ne fonctionne comme prévu. Nous pourrions créer ce compteur avec un additionneur dont l'une des valeurs d'entrée serait fixée à 1. Pour simplifier, logeons-le dans un seul bloc (**fig. 3**). Si nous connectons son entrée à la sortie (notre boucle), nous pensons créer un compteur incrémentiel.

Mais des questions se posent d'emblée. À quelle valeur ce compteur commence-t-il et à quelle vitesse compte-t-il ? La valeur initiale dépend de la façon dont l'alimentation du circuit a été appliquée et de la disposition du circuit. Elle dépendrait sans doute de la température et d'autres facteurs environnementaux. Il serait désastreux que notre circuit se comporte différemment selon qu'il pleuve ou qu'il vente.

Le pire, c'est que ce circuit ne fonctionnerait même pas. En effet, un circuit d'addition, comme la plupart des logiques combinatoires à sorties multibits, produit des résultats intermédiaires erronés. Dans le cas d'un additionneur, le bit le moins significatif est calculé en premier et chaque bit suivant utilise le résultat du bit qui le précède. Comme on n'attend pas que le résultat soit valide, les valeurs intermédiaires erronées sont renvoyées dans l'additionneur qui propage alors d'autres valeurs erronées et notre bel objet ne produit que des résultats faux. Comment régler ce problème ? Il nous suffirait de pouvoir contrôler la chronologie de la boucle de rétroaction. C'est justement ce que font les bascules de type D ou DFF (D *flip-flop*).

Avant de voir le DFF en détail, parlons des horloges. Une horloge est un signal qui passe de 0 à 1 puis de 1 à 0, indéfiniment, à une fréquence donnée (**fig. 4**). L'horloge des cartes Alchitry bascule 100 millions de fois par seconde, soit 100 MHz. Grâce à ce signal périodique, nos circuits vont percevoir le temps. La transition de 0 à 1, à savoir le front montant, constitue généralement la partie active du signal. Dans l'illustration, ces fronts sont matérialisés par des flèches.

Revenons au DFF (**fig. 5**). Les DFF sont un type de mémoire. Ils ont une entrée, D, et une sortie, Q. Lorsque leur entrée d'horloge passe de 0 à 1, la valeur de D est mémorisée et renvoyée sur Q jusqu'au prochain front montant de l'horloge. Peu importe que D change *entre* deux fronts montants consécutifs de l'horloge, Q ne change pas. Dans la figure 5, le DFF est dessiné avec les signaux optionnels d'activation (en = *enable*) et de réinitialisation (rst = *reset*). Le signal *en* peut être utilisé pour empêcher le DFF de copier une nouvelle valeur sur un front montant. Le signal *rst* est utilisé pour forcer la valeur de Q à une valeur connue. Les DFF des FPGA sont prévus pour être réinitialisés à 0 ou 1. Nous utilisons le DFF de notre compteur pour contrôler la boucle (**fig. 6**). La valeur initiale de Q, p.ex. 0, sera fixée par *rst*. Cela signifie que nous savons que Q vaut 0. Alors D passe à 1. Sur le front montant suivant de l'horloge, Q prend la valeur de D. Par conséquent Q passe à 1 et D passe à 2. Sur chaque front montant, Q augmente de 1. C'est ce que nous voulions ! La fréquence de l'horloge détermine la vitesse d'incrémentation de notre compteur, non sans quelques limitations. L'horloge doit être suffisamment lente pour que la valeur de D ait le temps de s'actualiser après un changement de Q. Nous excluons que notre DFF sauvegarde une des valeurs intermédiaires invalides que l'additionneur produit. Le temps nécessaire pour que la valeur de sortie de l'additionneur soit valide s'appelle le délai de *propagation.* C'est le temps qui s'écoule entre un changement des entrées et la stabilisation de la sortie. Plus nous ajoutons de logique, plus ce délai est long. Le délai dépend aussi de la technologie de fabrication du circuit. Les outils disposent de modèles pour chaque FPGA et si nous indiquons la fréquence d'horloge que nous utilisons, ils tenteront de réaliser notre projet de sorte que les exigences en matière de délai soient respectées. Dans cet exemple, nous avons un ensemble de DFF inséré pour boucler un bloc de logique combinatoire. Il est plus courant que la sortie d'un ensemble de DFF soit envoyée vers un autre ensemble de DFF via un bloc de logique combinatoire, créant ainsi un pipeline. Dans tous les cas, c'est le plus long délai de propagation du circuit qui fixe la fréquence maximale de l'horloge. En décomposant notre circuit en blocs de logique combinatoire ayant un délai de propagation voisin, nous pouvons optimiser la fréquence d'horloge. La synchronisation peut être laborieuse, mais en général, nous pouvons nous en sortir en utilisant la même horloge pour tout le circuit. Pour peu que nous n'ayons pas de circuit trop long à parcourir, les outils résoudront le problème pour nous. À 100 MHz, nous pouvons en pratique faire beaucoup de choses entre les DFF. Les problèmes ne surviennent que si nous essayons d'enchaîner trop de choses ou d'inclure trop de multiplications.

Le respect de la chronologie devient plus ardu à l'approche de la limite des ressources du FPGA. Comme les outils doivent intégrer de plus en plus de choses dans un FPGA de taille donnée, les possibilités de configuration se réduisent, ce qui peut les empêcher d'atteindre la vitesse nécessaire.

Listage 3.

```
module blinker (
   input clk,  // clock
   input rst,  // reset
   output out
 ) {

  always {
    out = 0;
  }
}
```

Travaux pratiques : faire clignoter une LED

Appliquons maintenant ce qui précède dans un projet de démonstration qui fera clignoter une LED. Créons un autre projet dans Alchitry Labs [4] et choisissons *Base Project* sur le menu déroulant *From Example*, (**fig. 7**). Cliquons sur l'icône *nouveau fichier* de la barre d'outils (icône

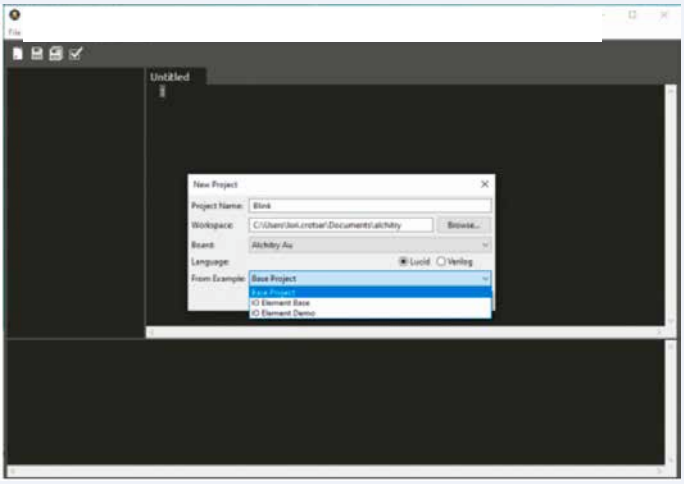


Figure 7 : Sélection du projet de base dans le logiciel Alchitry Labs.

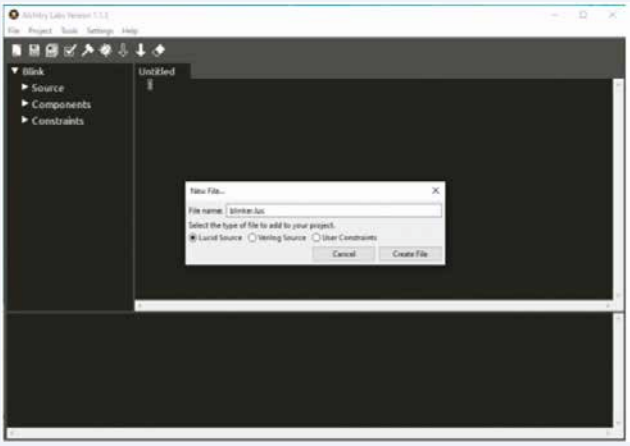


Figure 8 : Création d'un nouveau fichier source Lucid.

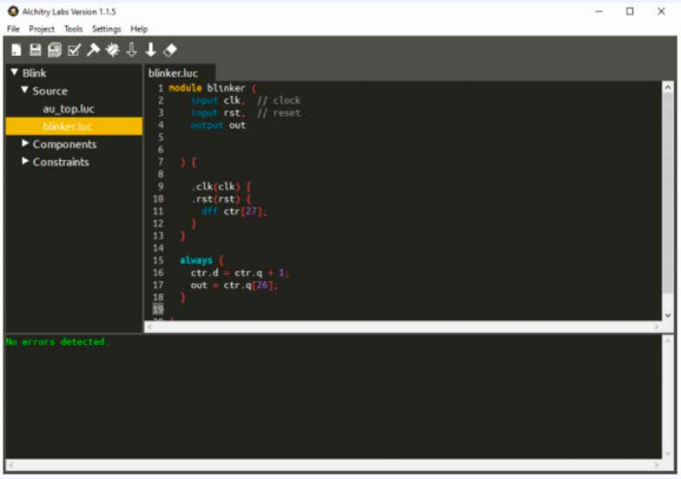


Figure 9 : Le "module" créé à partir du programme dans le listage 3.

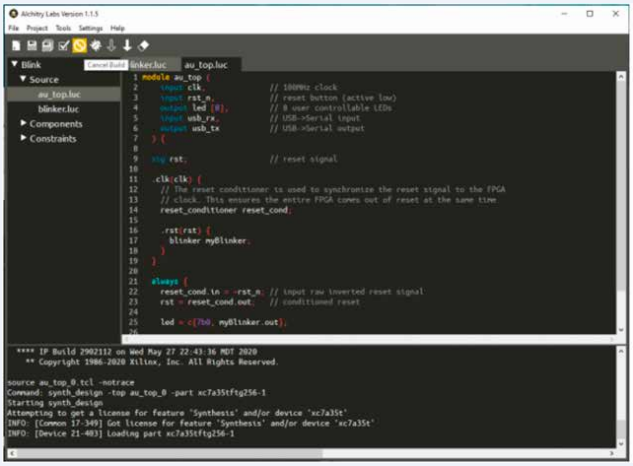


Figure 10 : Téléchargement du projet sur le tableau d'Alchitry.

la plus à gauche) et créons un nouveau fichier *Lucid Source* nommé *blinker.luc* (**fig. 8**). Cela va créer un module de base qui ressemble au **listage 3** pour le code et à la **figure 9** pour l'écran.

Le modèle de module par défaut ajoute les entrées d'horloge et de réinitialisation et une sortie factice pour le moment. Pour faire clignoter une LED, il faut créer un compteur qui sera utilisé pour basculer la LED. Si la LED bascule à chaque cycle d'horloge, le clignotement est trop rapide pour être visible.

Pour faciliter la tâche, nous définissons les blocs de connexion pour l'horloge (*clk*) et la réinitialisation (*rst*) puis déclarons le DFF à l'intérieur de ceux-ci :

```
.clk(clk) {
  .rst(rst) {
    dff ctr[27];
  }
}
```

Cela crée un ensemble de 27 DFF nommés *ctr*. Nous insérons ensuite le DFF dans le bloc "*always*".

```
always {
  ctr.d = ctr.q + 1;
  out = ctr.q[26];
}
```

Pour accéder aux signaux d'un module ou d'un DFF, nous utilisons la notation pointée. La première ligne du bloc *always* relie l'entrée D des DFF à la sortie Q plus 1. La valeur de *ctr.q* s'incrémentera ainsi une fois par cycle d'horloge. Notons que le signal *d* est en écriture seule et que *q* est en lecture seule. La 2e ligne prend le bit le plus significatif, le n° 26, et le connecte à la sortie. Comme *ctr* a une largeur de *27* bits, il a des indices de 0 à 26. Comme *ctr* s'incrémente une fois par cycle d'horloge, un nombre de 27 bits peut contenir 227 = 134 217 728 valeurs différentes. De plus, notre fréquence d'horloge étant de 100 MHz, *ctr* déborde une fois toutes les 1,34 s. Durant la première moitié de ce cycle, le bit le plus significatif sera 0 puis 1 durant la seconde moitié. En connectant ce bit à la sortie, nous basculerons la sortie toutes les 0,67 s.

Passons maintenant au module *top level* et instancions notre nouveau module. Notons que j'utilise une carte Au. Le module aurait la même apparence pour la Cu, mais le nom serait *cu_top* et non *au_top* (**listage 4**). Ici, j'ai ajouté un bloc de connexion pour le signal de réinitialisation et créé une instance du module clignotant appelée *myBlinker*. Ensuite, dans le bloc *always*, je l'ai connecté à la sortie *led* en utilisant la syntaxe de *concaténation.* Les éléments à l'intérieur de *c{...}* sont collés ensemble pour former un seul tableau. Dans notre cas, nous concaténons donc sept 0 avec le bit de *myBlinker.out*. Cela va éteindre les 7 LED de poids fort et connecter notre signal de clignotant à la LED de poids faible. Nous pouvons maintenant réaliser le projet en cliquant sur l'icône du marteau, puis le charger dans notre carte en

Listage 4.

```
module au_top (
    input clk,          // 100MHz clock
    input rst_n,        // reset button (active
low)
    output led [8],     // 8 user controllable
LEDs
    input usb_rx,       // USB->Serial input
    output usb_tx       // USB->Serial output
  ) {

  sig rst;              // reset signal

  .clk(clk) {
    // The reset conditioner is used to
    // synchronize the reset signal to the FPGA
    // clock. This ensures the entire FPGA comes
    // out of reset at the same time.
    reset_conditioner reset_cond;

    .rst(rst) {
      blinker myBlinker;
    }
  }

  always {
    reset_cond.in = ~rst_n; // input raw inverted
reset signal
    rst = reset_cond.out;   // conditioned reset

    led = c;

    usb_tx = usb_rx;        // echo the serial data
  }
}
```

Listing 5.

```
module blinker #(
    MAX_VALUE = 50000000 : MAX_VALUE > 0
  )(
    input clk,  // clock
    input rst,  // reset
    output out
  ) {

  .clk(clk) {
    .rst(rst) {
      dff ctr[$clog2(MAX_VALUE)];
      dff led;
    }
  }

  always {
    ctr.d = ctr.q + 1;
    if (ctr.q == MAX_VALUE-1) {
      ctr.d = 0;
      led.d = ~led.q;
    }

    out = led.q;
  }
}
```

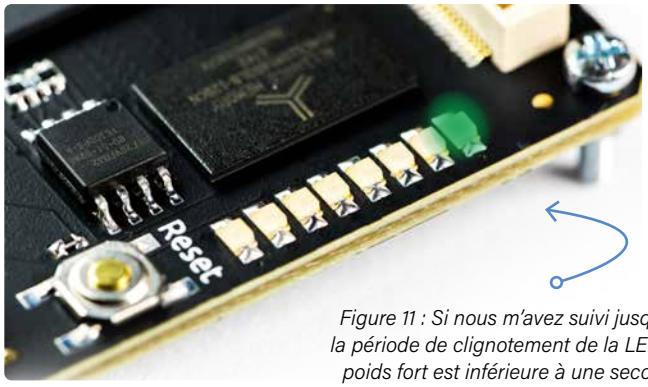


Figure 11 : Si nous m'avez suivi jusqu'ici, la période de clignotement de la LED de poids fort est inférieure à une seconde.

appuyant sur l'icône de la flèche pleine vers le bas (**fig. 10**). La LED du haut devrait maintenant clignoter un peu plus lentement qu'une fois par seconde (**fig. 11**).

Amélioration des modules

Nous pouvons améliorer notre clignotant. Tout d'abord, la LED clignote avec une période inhabituelle. Ramenons-la à 1 s exactement en comptant jusqu'à 50 000 000 avant d'allumer la LED. Pour ce faire, nous avons besoin d'un autre DFF pour mémoriser l'état de la LED.

```
  .clk(clk) {
    .rst(rst) {
      dff ctr[28];
      dff led;
    }
  }
```

Dans le bloc "*always*", nous vérifions que *ctr.q* vaut 49 999 999 (de 0 à 49 999 999, il y a 50 000 000 incréments) et si c'est le cas, nous le remettons à 0 et basculons la LED.

```
  always {
    ctr.d = ctr.q + 1;
    if (ctr.q == 49999999) {
      ctr.d = 0;
      led.d = ~led.q;
    }

    out = led.q;
  }
```

Ici, j'ai utilisé l'opérateur d'inversion de bits, le tilde : ~. Il permet d'inverser chaque bit d'un signal. Comme *led.q* n'a qu'un seul bit, l'opération n'inverse que ce bit.

Si nous avons bonne mémoire, nous savons qu'en toutes circonstances un signal doit avoir une valeur, sauf pour les DFF. Comme les DFF peuvent mémoriser leur valeur, si l'entrée *d* ne reçoit pas d'affectation pendant un cycle d'horloge, la valeur du DFF ne changera pas.

Notre module ne stocke plus que les valeurs de 0 à 49 999 999 dans *ctr*, mais il s'agit toujours d'un tableau de 28 bits. C'est du gaspillage, car il ne faut que 26 bits pour stocker nos valeurs. Nous pourrions juste réduire à 26 la taille du tableau, mais si nous voulons changer la valeur maximale, il faudrait recalculer cette valeur. Des fonctions Lucid peuvent se charger de ce calcul pour nous. Par ex. *$clog2(50000000)* qui calculera la valeur plafond du log base 2 de la valeur donnée. Cela équivaut à "combien de bits faut-il pour stocker un tel nombre". Dans notre cas, il trouvera 26.

```
dff ctr[$clog2(50000000)];
```

En ce qui concerne le changement de la valeur maximale, nous modifions notre module pour qu'il l'accepte comme paramètre afin

Sur la vague FPGA

L'internet offre de nombreuses ressources sur les FPGA. Voici quelques liens choisis pour stimuler votre intérêt.

- Alchitry, "Using FPGAs", www.sparkfun.com/fpga.
- J. Rajewski, "How Does an FPGA Work?", SparkFun, https://learn.sparkfun.com/tutorials/how-does-an-fpga-work
- J. Rajewski, "First FPGA Project: Getting Fancy with PWM", SparkFun, www.elektormagazine.fr/esfe-en-fpga2
- J. Rajewski, "External IO and Metastability", SparkFun, https://learn.sparkfun.com/tutorials/external-io-and-metastability
- S. Cass, "Painless FPGA Programming", IEEE Spectrum, November 2020, https://spectrum.ieee.org/geek-life/hands-on/painless-fpga-programming

de pouvoir la spécifier lors de l'instanciation du module. Il suffit d'ajouter une *liste de paramètres à* notre module. Cette liste précède la liste des *ports*.

```
module blinker #(
    MAX_VALUE = 50000000 : MAX_VALUE
> 0
  )(
    input clk,  // clock
    input rst,  // reset
    output out
  ) {
```

La syntaxe spécifiée pour la liste des ports est : #(param, param, param). Chaque paramètre peut n'être qu'un simple nom (en capitales) ou plus complexe, comme dans notre exemple où nous avons fixé une valeur par défaut de 50 000 000. L'usage de valeurs par défaut est conseillé, sauf quand on veut forcer la spécification d'une valeur à l'instanciation. Après l'affectation par défaut, nous pouvons spécifier une condition à laquelle le paramètre doit répondre. Elle doit utiliser le paramètre lui-même et peut aussi utiliser tous les paramètres déclarés avant lui. L'évaluation de cette condition doit donner "*true*" (vrai). Si ce n'est pas le cas, une erreur est signalée lors de l'instanciation. Cela permet d'écrire le module avec quelques hypothèses sur la valeur du paramètre et de savoir qu'elles seront respectées. Nous pouvons alors remplacer les occurrences de 50 000 000 dans notre module par `MAX_VALUE` (**listage 5**).

Si nous compilons et chargeons le projet maintenant, la LED doit clignoter à un rythme d'une fois par seconde. Cependant, si nous revenons au module de niveau supérieur, nous pouvons modifier l'instanciation comme ceci :

```
blinker myBlinker(#MAX_VALUE(25000000));
```

Maintenant, si nous compilons et chargeons le projet, la LED clignotera deux fois par seconde.

En résumé

Vous voilà au bout de vos peines. Les circuits FPGA sont constitués de blocs de logique combinatoire qui effectuent tous les traitements, et de DFF qui mémorisent les valeurs et contrôlent le flux de données. Les circuits eux-mêmes sont décomposés en modules. Les modules peuvent être utilisés par d'autres modules et peuvent avoir des paramètres pour personnaliser chaque instanciation de ceux-ci. Chaque fois qu'un module est instancié, le circuit qui lui est destiné est dupliqué dans le FPGA.

Lors de la conception de matériel, il est important de réfléchir à la manière dont ce projet pourrait être mis en œuvre pour créer des circuits efficaces. Dans le cas de notre clignotant, sans nous préoccuper de la vitesse de clignotement, la première version du module nécessiterait beaucoup moins de ressources pour sa mise en œuvre. En effet, il n'a pas besoin d'un comparateur pour vérifier la valeur du compteur. Il continue simplement à ajouter 1 et utilise le débordement naturel de l'addition binaire pour initialiser le compteur.

200646-03

Alchitry

Alchitry et Justin Rajewski

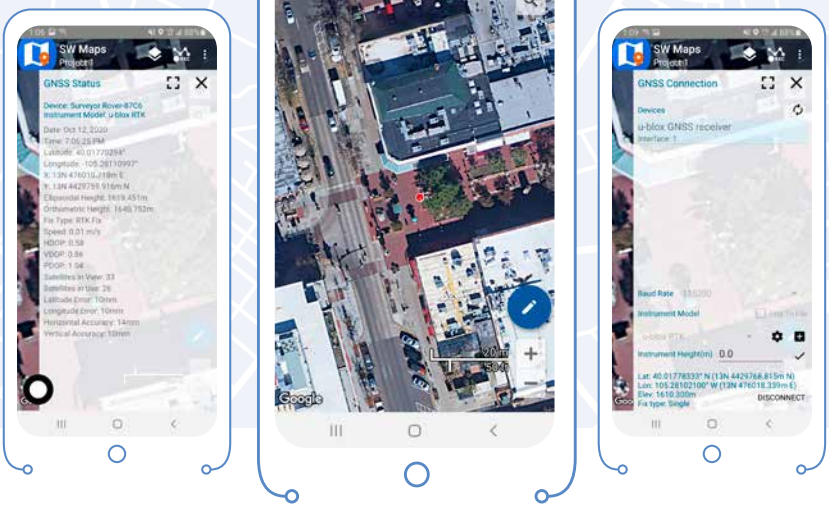
Soutenu sur KickStarter, Alchitry est l'une des sources les plus complètes et les mieux informées sur les FPGA. Séduite par les cartes Alchitry Au et Alchitry Cu, l'équipe de SparkFun a lancé une collaboration avec Justin Rajewski, fondateur et ingénieur à tout faire d'Alchitry. Aujourd'hui, Justin se concentre sur le développement de nouveaux produits, la construction de projets et la génération de contenu. Il a tiré parti de l'expérience de SparkFun en matière de production et de logistique pour fabriquer ses cartes. Justin apporte à SparkFun son expertise inégalée en matière de FPGA.

WEBLINKS

[1] SparkFun, "Using FPGAs" : https://www.sparkfun.com/fpga
[2] Wikipedia, "Hardware Description Language" : https://en.wikipedia.org/wiki/Hardware_description_language
[3] Lucid: https://alchitry.com/pages/lucid-fpga-tutorials
[4] Alchitry Labs: https://alchitry.com/blogs/tutorials/getting-started-with-the-au
[5] carte Alchitry Au: https://www.elektormagazine.fr/esfe-en-fpga1
[6] carte Alchitry Cu: https://www.sparkfun.com/products/16526

J'ai construit ma **station de référence GNSS**

Global Navigation Satellite System



Nathan Seidle (États-Unis)

La cinématique en temps réel du GNSS (RTK) est étonnante, mais la principale source de difficultés est l'accès aux données de correction. Cet article est consacré à la mise en place de votre propre antenne fixe sur votre toit ou sur une autre structure fixe et à la configuration d'un mini-ordinateur pour justement distribuer ces données sur l'internet et y accéder par WiFi ou à partir d'un téléphone cellulaire ou d'un modem.

Dans de précédents articles publiés sur le site de SparkFun, j'ai expliqué comment obtenir des données de correction RTCM accessibles au public, mais celles-ci peuvent être incomplètes [1]. J'ai également expliqué comment mettre en place votre propre base temporaire pour envoyer des données de correction RTCM via une liaison de télémesure radio [2], mais des difficultés surviennent si vous êtes à un kilomètre ou plus de votre base. Avant d'aller plus loin, assurez-vous d'être à l'aise avec U-Center [3], et consultez le tutoriel *What Is GPS RTK* ? de Sparkfun [4]. Puis considérez cet article comme la suite de *Setting up a Rover Base RTK System* [5].

Nous allons beaucoup parler de **NTRIP**. J'ai trouvé frustrants et déroutants les descriptions et graphiques de NTRIP, celui-ci n'est qu'un moyen sophistiqué pour fournir à un récepteur par Internet des données de correction par rapport à un point donné. Voyez-le comme un flux musical pour votre récepteur. Pour que votre récepteur puisse jouer, il lui faut une source de musique constante. Pour la musique, il y a le choix (YouTube, Spotify, Pandora), de même, il existe plein de sources pour RTCM (Trimble, Leica, Telit, *etc.*). Tous ces services RTCM facturent des montants différents et fonctionnent de manière suffisamment différente pour prêter à confusion. Moi, ce que je veux, c'est ma musique !

Ce tutoriel vous montrera comment créer vos propres données de correction GNSS (*Global Navigation Satellite System*) et les distribuer sur Internet, le tout gratuitement (ou au prix d'un mini-PC dédié si nécessaire) ! Vous serez votre propre service de diffusion musicale ! Votre récepteur pourra écouter ces données de correction en utilisant une connexion de téléphone portable. Oui, nous parlerons des clients et serveurs NTRIP et des points de référence, mais ne vous inquiétez pas, il s'agit simplement de faire passer des octets d'un ordinateur à l'autre via l'internet.

Installation de la base statique et des lasers !

Dans le tutoriel [2], nous avons décrit comment créer une station de base temporaire avec la méthode de sondage de 1 à 10 minutes. La méthode de base temporaire est souple, mais moins précise et le temps nécessaire peut varier considérablement. Le ZED-F9P offre un moyen beaucoup plus rapide de fournir des corrections de base : si vous connaissez l'emplacement de votre antenne, vous pouvez définir les coordonnées du récepteur et celui-ci commencera immédiatement à fournir des corrections RTCM. La question, c'est : « quelle est la position de l'antenne ? ». C'est comme si vous aviez besoin d'un fer à souder pour assembler votre kit de fer à souder. Par où commencer ?

Q. Pourquoi ne pas faire juste un relevé de mon antenne fixe pour connaître sa position ?

R. Un relevé est facile à faire et parfait pour établir l'emplacement d'une base sur le terrain, mais il n'est pas recommandé pour obtenir la position fixe d'une station de base statique, car il est moins précis. Le PPP ou *Precise Point Positioning* est beaucoup plus précis et recommandé pour obtenir la position de votre antenne. C'est un procédé similaire, mais qui implique de faire rebondir des lasers sur les satellites ! Un problème majeur est le décalage, de souvent 1 m ou plus, des orbites prédites. Les stations au sol font rebondir des lasers sur les satellites individuels lorsqu'ils passent au-dessus d'elles et utilisent ces nouvelles données pour calculer les orbites réelles des satellites. L'utilisation de ces nouvelles données d'éphémérides, lorsqu'elles sont disponibles, combinées aux données brutes du récepteur, permet de calculer de meilleures positions. C'est la base du PPP, et le processus fonctionne ainsi :

- Installer une antenne à un endroit fixe ;
- Recueillir 24 h de données GNSS brutes de cette antenne ;
- Transmettre les données brutes à un centre de traitement pour le PPP ;
- Obtenir une position très précise de l'antenne que nous utilisons pour établir un « mode fixe » sur un récepteur.

Il y a de très bons articles sur le PPP. Nous allons effleurer le sujet, mais pour plus d'informations, consultez :

- L'excellent « *HOWTO PPP* » de Gary Miller [6] ;
- Le « *PPP* » d'Emlid [7] ;
- Le « *GNSS PPP* » de Suelynn Choy [8].

Fixez votre antenne

Il ne faut plus que votre antenne bouge une fois sa position déterminée. Il est envisageable d'investir dans une antenne haut de gamme, mais l'antenne classique u-blox L1/L2 [9] donne satisfaction. Montez l'antenne sur un plan de masse approprié sur une surface fixe qui a une vue très dégagée du ciel. Rien à proximité.

Nous avons fixé l'antenne u-blox sur la couvertine métallique au sommet du bâtiment de SparkFun (**fig. 1**). Les aimants de l'antenne u-blox ne sont pas complètement permanents, mais ont été testés pour résister au flux d'air sur une voiture, ce qui devrait être suffisant pour les vents de plus de 160 km/h qui soufflent dans le Colorado ou la vallée du Rhône. L'antenne u-blox ANN-MB-00 est équipée d'un câble de 5 m, mais comme celui-ci n'était pas assez long pour aller du toit de SparkFun au récepteur, nous avons fixé une rallonge SMA de 10 m. S'il est vrai que la plupart des antennes L1/L2 ont un amplificateur intégré, chaque mètre de rallonge et chaque connecteur dégradera tout de même un peu le signal GNSS. Limitez l'utilisation de convertisseurs de connecteurs et utilisez une rallonge aussi courte que possible. Si vous voulez utiliser une antenne de qualité supérieure sans base magnétique, nous avons trouvé un excellent moyen de créer un point fixe stable sans avoir à percer votre toit !

La plupart des antennes de topographie ont un filetage de ⅝" (16 mm) 11-TPI (filets par pouce) sur le bas de l'antenne. Heureusement, ⅝" 11-TPI est également le filetage que l'on trouve sur les plots d'ancrage dans les magasins de bricolage aux États-Unis. Les plots d'ancrage, conçus pour assujettir murs et fondations, conviennent aussi pour fixer une antenne. Il existe également des ancres en béton avec de l'époxy, renseignez-vous.

Comment monter une antenne sur mon toit ? Heureusement, il me restait deux parpaings d'une station météo à base d'Electric Imp, hors service depuis longtemps (**fig. 2**).

La première étape consiste à percer le trou de ⅝" dans le parpaing (**fig. 3**). La mèche pour maçonnerie m'a coûté 20 €, on en trouve pour moins de 10 €. Le ruban bleu, c'est pour la profondeur : environ 6,5 cm

Figure 1. L'antenne U-blox fixée au parapet de SparkFun.

Figure 2. Station météorologique sur trépied lesté.

Figure 3. Oui, c'est un parpaing. Et ça marche !

Figure 4. Errghh, un parpaing foutu.

Figure 5. Ancrage fixé par un boulon - et ce parpaing-là est intact !

Figure 6. Fixation de la base de l'antenne sur le filetage de l'ancrage.

pour un parpaing de 9 cm. Une fois le trou percé, basculez le bloc pour en faire sortir toute la poussière. Ensuite, mettez l'ancrage en place. Ne serrez pas trop, vous risquez de fendre le bloc (**fig. 4**). Heureusement, j'en avais un second !

Une fois l'ancrage engagé d'environ 5 cm dans le trou, serrez le boulon (**fig. 5**). L'ancrage remontera alors en comprimant le collier en place. **Note :** J'ai serré le boulon à la main puis donné un demi-tour avec une clé. Si vous serrez trop fort, vous risquez de pousser le collier trop loin et de fendre le parpaing. On arrime juste une antenne de 400 g.

Comme le montre la **fig. 6,** j'ai utilisé un deuxième boulon, fixé contre la base de l'antenne pour la maintenir en place et empêcher toute rotation dans un sens ou dans l'autre. Les lecteurs avertis remarqueront mon adaptateur TNC à SMA sur la photo. Ce n'est pas le bon genre. À l'origine, j'ai utilisé une extension SMA pour connecter mon GPS-RTK-SMA à mon antenne u-blox L1/L2 sur mon toit. Le GPS-RTK-SMA s'attend à une connexion SMA normale, donc l'extrémité de l'extension ne se connectera pas à cet adaptateur. Alors, avant de descendre de l'échelle, testez et connectez tout ! Heureusement, j'ai un jeu d'adaptateurs et j'ai trouvé le bon convertisseur TNC vers SMA pour répondre au besoin.

J'ai fait une boucle avec la rallonge SMA autour de la base. Si on tire sur le câble SMA, la tension sera transférée au boulon, pas à la connexion TNC de l'antenne.

Attention à la foudre : Mon antenne est plus basse que le parapet (**fig. 7**), donc les coups de foudre sont peu probables. Si votre antenne est le point le plus élevé aux alentours, alors pensez à la protection contre la foudre.

Collecter des données GNSS brutes

Une fois l'antenne placée là **où elle ne bougera pas ni ne sera déplacée**, il faut établir sa position. Ouvrez le programme U-Center et voyez s'il se verrouille avec plus de 25 satellites en vue avec votre ZED-F9P. En supposant que vous ayez une bonne réception, il faut maintenant régler le récepteur pour qu'il émette les données brutes des satellites (**fig. 8**). Une fois le message *RXM-RAWX* activé sur USB, vérifiez la réception dans la visionneuse de paquets (**fig. 9**). Les messages RAWX sont binaires, vous ne pourrez donc pas les voir dans la visionneuse de texte.

Appuyez sur le bouton *Record* (**fig. 10**). Cela enregistrera toutes les données (NMEA, UBX et RAWX) du récepteur dans un fichier *.ubx. Laissez ce dernier fonctionner pendant 24 h. Ne vous inquiétez pas si vous allez trop loin, mais sachez qu'un fichier de 24 h occupera environ 300 Mo ; ne le laissez donc pas fonctionner pendant un mois.

Une capture de 6 h est correcte ; une de 24 h un peu meilleure (notez l'échelle logarithmique pour l'erreur de position dans le graphique de la **fig. 11**). La plupart des services d'analyse des PPP acceptent plus de 24 h de données, mais ils peuvent les tronquer à 24 h. Si vous saisissez 30 h de données RAWX, c'est bon, nous vous montrerons comment découper un fichier trop long.

Le fichier UBX de 300 Mo devra être converti en RINEX (*Receiver Independent Exchange Format*). Le fameux *RTKLIB* est là pour vous aider [10]. Nous conseillons la version modifiée de RTKLIB de rtklibexplorer [11] mais vous pouvez obtenir la version originale de RTKLIB [10]. Ouvrez *RTKCONV*. Sélectionnez votre fichier .UBX et cliquez sur *Convert*. La conversion de notre fichier de 300 Mo a pris environ 30 s. Vous devriez voir un fichier *.obs une fois la conversion terminée.

Si votre fichier de données dure 25 h ou un peu plus, c'est parfait. Pour réduire un fichier RINEX trop volumineux (ou qui dure 40 h, réduisez la fenêtre temporelle (**fig. 12**). Convertissez tout le fichier, puis cliquez sur l'icône du bloc-notes pour ouvrir le fichier OBS. Vous verrez les heures de début et fin du GPS de cette capture (**fig. 13**). En utilisant ces temps, vous pouvez limiter la fenêtre de temps à ce dont vous avez besoin et reconvertir le fichier.

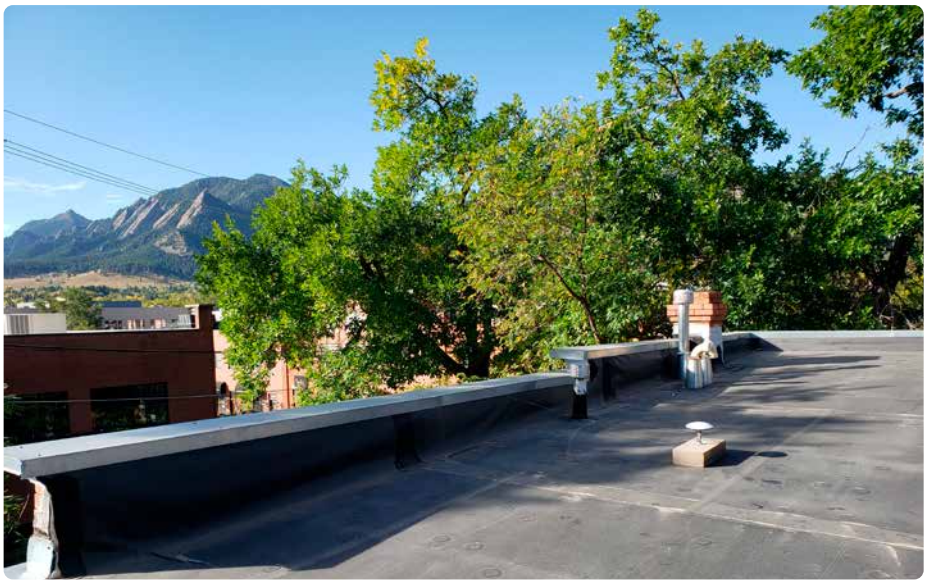


Figure 7. Antenne L1/L2 semi-fixe sur toit plat. C'est du boulot de hisser ces 20 kg de béton sur le toit, mais la vue est assez spectaculaire !

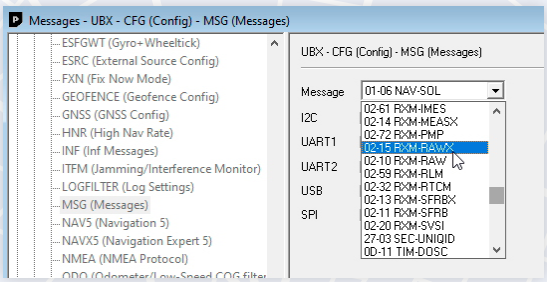


Figure 8. Sélectionnez le format de données brutes pour les messages (MSG).

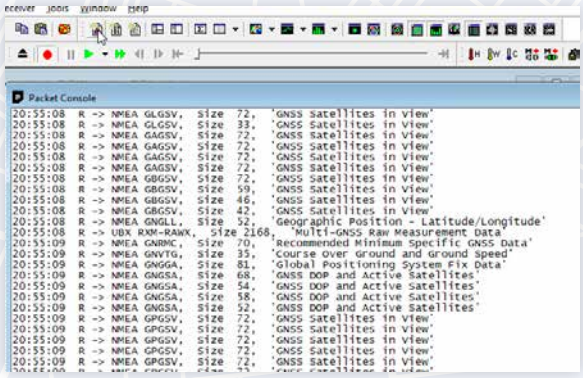


Figure 9. Visualisation d'un paquet RAWX dans la visionneuse de paquets.

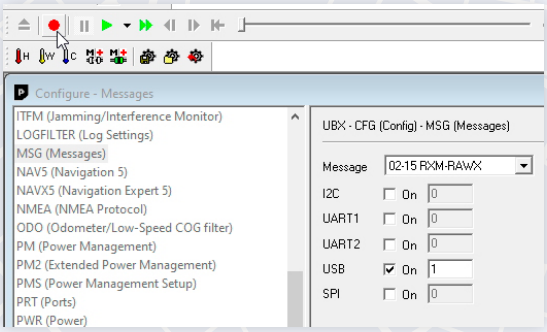


Figure 10. Appui sur le bouton d'enregistrement.

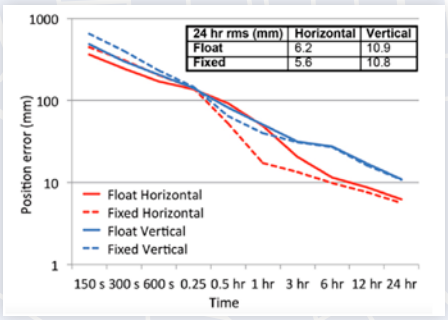


Figure 11. Graphique tiré de la présentation de Suelynn Choy sur le positionnement de points de précision par GNSS en 2018 [8].

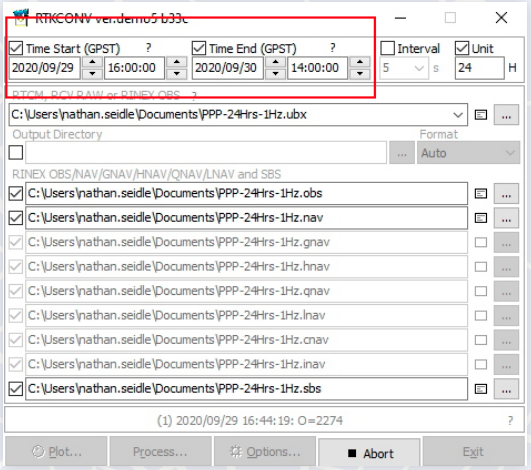


Figure 12. Le raccourcissement de la fenêtre temporelle est une méthode pour que la taille des données reste gérable.

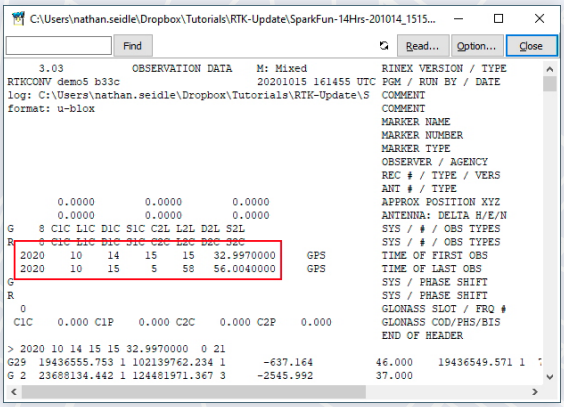


Figure 13. Contenu d'un fichier OBS réduit à 14 h de données.

Figure 14. Prêt à envoyer des données au CSRS pour une analyse approfondie des chiffres.

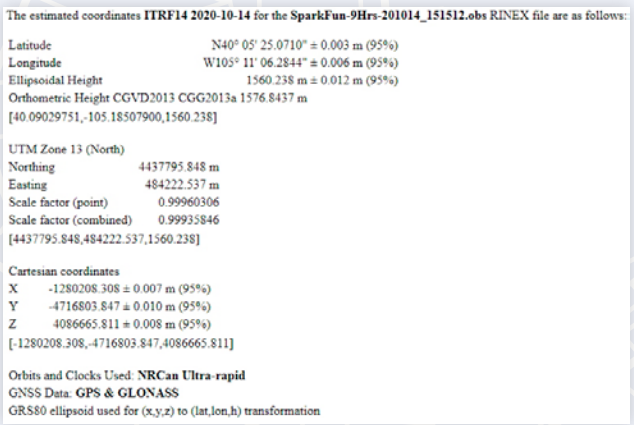


Figure 15. La position de l'antenne de SparkFun avec une précision de ±7 mm !

Q. Pourquoi n'augmentons-nous pas la cadence ? Plus c'est mieux !

R. Le ZED-F9P peut aller jusqu'à 30 Hz. Pourquoi ne pas obtenir des données RAWX à plus de 1 Hz ? Parce que la nature ne se déplace pas aussi vite. La plupart des services d'analyse PPP ignoreront tout ce qui est supérieur à 1 Hz. OPUS va même jusqu'à « décimer tous les taux d'enregistrement à 30 s ». Et vos fichiers OBS seront monstrueux. Si 24 h représentent 300 Mo à 1 Hz, il s'ensuit que 24 h à 30 Hz feront environ 9 Go. Donc non, gardez 1 Hz.

Nous devons maintenant faire passer les données brutes des satellites GNSS au format RINEX (**.obs*) par un centre de post-traitement pour essayer d'obtenir la position réelle de l'antenne. Il existe plusieurs services, mais nous avons utilisé avec succès le service canadien CSRS-PPP [12]. Le service géodésique national américain fournit un service appelé OPUS [13], mais nous avons été frustrés par ses limitations en taille et format de fichier. Vous verrez peut-être les choses autrement. Zippez votre fichier .obs, puis créez un compte auprès du CSRS. Sélectionnez ITRF, puis téléchargez votre fichier (**fig. 14**). Tournez-vous les pouces pendant quelques heures, vous finirez par recevoir un courriel avec un rapport PDF de la position de votre antenne (**fig. 15**).

Si tout va bien, vous devriez avoir une position très précise pour votre antenne. Pour les récepteurs u-blox, nous sommes surtout intéressés par les coordonnées ECEF. L'ECEF [14] est **fascinant**. Plutôt que *lat*. et *long*. l'ECEF est le nombre de mètres à partir du référentiel internationalement reconnu du centre de masse de la Terre. En gros, vos coordonnées ECEF indiquent votre distance par **rapport au centre de la Terre**. Super.

Maintenant que vous connaissez la position de votre antenne, indiquons à la ZED-F9P où se trouve son antenne à quelques millimètres près.

Revenez au message TMODE3 et entrez les coordonnées ECEF du rapport (**fig. 16**). En supposant que ce récepteur soit relié à une antenne fixe, nous conseillons de sauvegarder ces paramètres en BBR/Flash afin qu'à chaque mise sous tension de ce récepteur, il passe immédiatement en mode TIME et commence à émettre des données RTCM.

Presque immédiatement après la saisie de l'ECEF, votre module devrait commencer à produire des messages RTCM. Utilisez la visionneuse de paquets pour confirmer (**fig. 17**). Si vous ne les voyez pas, consultez le tutoriel SparkFun qui décrit comment installer une station de base [2]. Il est fort probable que vous n'ayez pas activé le message RTCM requis. Encore une fois, veillez à sauvegarder votre réglage sur BBR/Flash afin qu'à chaque mise sous tension, ce récepteur commence à distribuer des données de correction sans intervention de l'utilisateur.

Configuration du mini-ordinateur

Vous avez installé votre antenne. Vous avez la ZED-F9P qui produit du RTCM. Beau travail ! Maintenant, comment transmettre ces données au monde entier et aux récepteurs où qu'ils soient ?

Vous aurez besoin d'un ordinateur dédié connecté à l'internet pour vous connecter au ZED-F9P, recevoir les données série, puis transmettre ces données sur l'internet. Nous vous conseillons d'utiliser un PC pour la distribution. Oui, Windows est pénible et pas aussi stable que Linux, mais comme U-Center n'existe que sur Windows, c'est la seule option. Il est pratique d'avoir un bureau déporté dans la machine dédiée, de bidouiller quelques paramètres du récepteur avec U-Center, et de continuer à distribuer. Je ne sais pas comment obtenir cette même souplesse sous Linux. En outre, je suis loin d'être un administrateur système. Voyez ce qui suit comme une méthode de configuration de votre ordinateur, mais ce ne sont que mes propres notes pour recréer le système en cas de besoin.

- Procurez-vous une machine désaffectée avec Windows. Comptez ~100 € pour un mini-PC. N'importe quel vieux PC fera l'affaire. Pas besoin d'une grosse puissance de calcul ! Si possible, achetez un mini-PC avec son kit de fixation, il sera plus facile à fixer à un mur.
- Dans le BIOS du mini-PC, activez la fonction *Auto-Power On* après une coupure d'alimentation. Activez le bureau à distance. De nombreux tutoriels montrent comment faire. Le but est d'amener le mini-PC au stade où vous pourrez configurer et contrôler le PC confortablement depuis votre bureau, sans monter sur le toit.

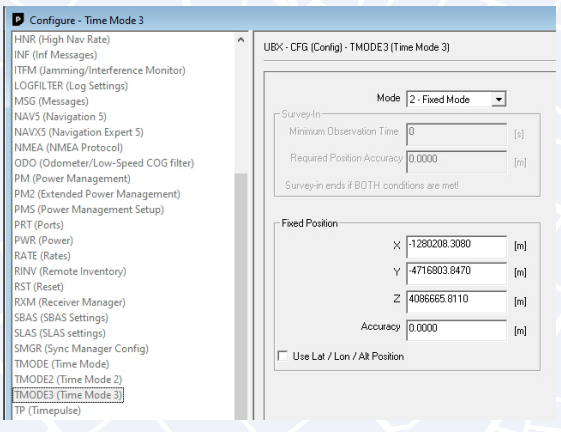


Figure 16. Copie des données reçues du CEF dans la fenêtre Time Mode 3.

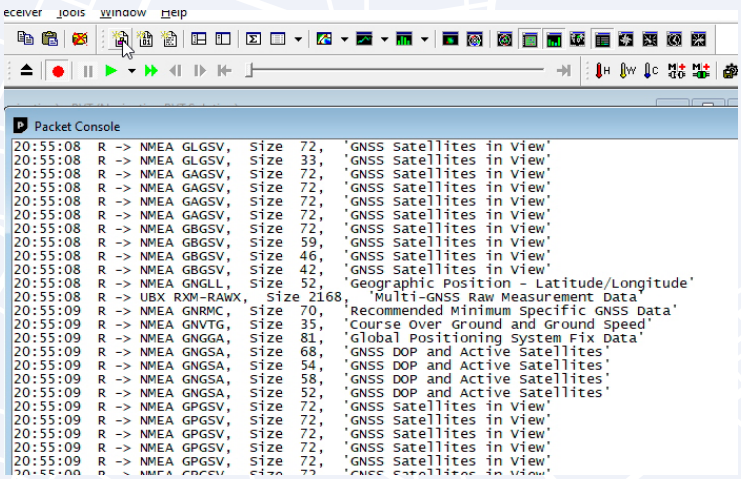


Figure 17. La console de paquets montrant les messages RTCM transmis par le module.

Produisez vos propres données de correction GNSS et distribuez-les sur l'internet. C'est gratuit !

Figure 18. Boîtier « Orbit » résistant aux intempéries avec disjoncteur différentiel (GFCI) intégré.

- Pensez à rendre statique l'adresse IP du mini-PC, soit à partir du mini-PC, soit avec son adresse MAC depuis votre routeur. Cela en facilitera l'accès via le protocole RDP (*Remote Desktop Protocol*).
- Si vous prévoyez d'accéder au mini-PC depuis un réseau externe, vous devrez activer la redirection de port sur le port 3389 (pour RDP) vers son adresse IP statique.
- Si vous prévoyez d'utiliser U-Center comme distributeur, il faudra activer la redirection de port sur le port 2101 (pour NTRIP) vers l'adresse IP statique du mini-PC (j'y reviendrai).
- Désactivez l'économie d'énergie de Windows. Le mini-PC ne doit jamais s'éteindre. Le coût énergétique n'est pas nul : j'ai mesuré environ 4,4 W lorsque l'appareil était connecté au récepteur GNSS et à la distribution. Cela représente moins de 7 € par an à 0,18 € par kWh (moyenne française en 2020).

À ce stade, vérifiez que vous pouvez accéder au PC depuis un bureau à distance. Si vous y parvenez, installez le mini-PC sur le terrain. Nous devrions pouvoir configurer tout le reste à distance.

- Si l'ordinateur doit être installé à l'extérieur, pensez à un disjoncteur différentiel intégré. Le boîtier Orbit [15] est très bien (**fig. 18**).
- Désactivez toutes les notifications de Windows (utilisez le sous-menu *Notifications et actions*).
- Pour des raisons de sécurité, désactivez Bluetooth et si vous utilisez un réseau Ethernet câblé, pensez à désactiver le WiFi.
- Installez U-Center.
- Installez une copie locale de la version de RTKLIB de rtkexplorer.
- Installez un terminal série de votre choix.
- Connectez le ZED-F9P sur l'USB C.
- Branchez l'antenne (fixe ou semi-fixe).
- Envisagez de connecter une radio 915 MHz [16] par une liaison micro USB (sans la souder sur la BoB ZED-F9P). Voir ci-dessous.

La mise en place est propre à chaque site. Chez moi, le mini-PC est à l'intérieur, le câble d'antenne passe par la fenêtre entrouverte. Chez SparkFun, le mini-PC est dans un boîtier extérieur Orbit avec alimentation électrique (très pratique !) (**fig. 19**). Les bandes auto agrippantes facilitent l'installation de l'électronique dans un boîtier ; elles maintiennent l'appareil en place tout en permettant à l'utilisateur d'y accéder si besoin. Une fois tout installé, connectez-vous à distance et confirmez que vous pouvez configurer le récepteur GNSS en utilisant U-Center.

Si vous ne l'avez pas encore fait, ou si votre antenne a été complètement déplacée, pensez à refaire le relevé PPP de votre antenne comme décrit dans la section précédente.

Configuration du distributeur

Le mini-PC Windows est maintenant configuré, parlons de la distribution. Comme mentionné, NTRIP est la norme industrielle pour le transfert des corrections RTCM sur l'internet. Pour nos besoins, nous devons « distribuer » depuis la station de base et utiliser un « client » au niveau du récepteur pour avoir accès au distributeur. Plusieurs options existent :

- utiliser les distributeur et client de U-Center ;
- utiliser le distributeur de STRSVR ;
- utiliser STRSVR comme serveur et RTK2GO comme distributeur.

Il existe plusieurs façons de transmettre des données du ZED-F9P à Internet. Commençons par la plus simple.

U-center comme distributeur

U-center dispose d'un distributeur *NTRIP* très facile à utiliser (**fig. 20**). En termes de facilité d'utilisation, c'est de loin la plus simple (la deuxième après l'utilisation de radios qui ne nécessitent aucune configuration). Il suffit d'entrer un nom d'utilisateur, un mot de passe et des informations sur le point de référence, puis de cliquer sur OK. U-center configurera *automatiquement* le récepteur pour diffuser les trames RTCM et commencera à transmettre les données de correction sur le port 2101 à toute personne qui saisit l'adresse IP de ce PC avec les bons identifiants (généralement en utilisant un client NTRIP).

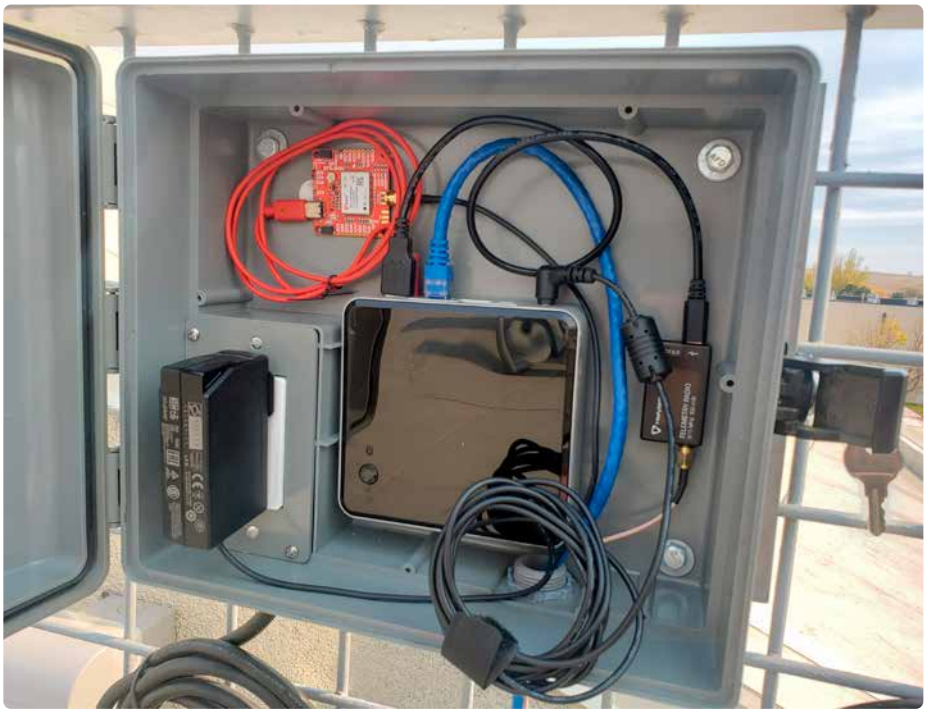


Figure 19. NUC (mini PC) connecté à un ZED-F9P et à une radio 915 MHz par USB.

Pour :
- Incroyablement simple à configurer.

Contre :
- Vous devrez créer une faille dans votre routeur pour le port 2101 (qui n'est pas le plus sûr).
- Il n'y a actuellement aucun moyen de démarrer automatiquement U-Center en mode « distributeur ». Cela signifie qu'à chaque fois que le mini-PC s'éteint ou redémarre, vous devrez vous connecter à la machine, ouvrir U-Center et redémarrer le distributeur.

Nous recommandons d'utiliser U-Center pour se familiariser avec NTRIP. C'est une expérience très satisfaisante et très instructive pour obtenir des données de correction, mais ce n'est pas notre choix à long terme.

STRSVR comme distributeur

Rapidement, et juste parce que j'ai perdu une journée à essayer de faire fonctionner ce système : Même s'il l'affiche (**fig. 21**), RTKLIB ne prend pas en charge la distribution NTRIP.

Pour:
- STRSVR peut être démarré automatiquement.

Contre :
- Le distributeur ne fait rien.

STRSVR et RTK2GO

La solution gagnante est d'utiliser STRSVR comme serveur NTRIP, puis RTK2GO comme distributeur NTRIP. C'est déroutant, mais cela suppose de télécharger des données sur un serveur sur l'internet.

Beaucoup d'applications Windows prétendent être des distributeurs NTRIP. En général, nous les avons trouvées *horribles*. La solution la plus simple est d'utiliser RTK2GO. RTK2GO semble être un projet favori du SNIP. Nous conseillons de créer un point de référence et un mot de passe via RTK2GO.com. Oui, cela ressemble à du spam, mais nous pensons que c'est la meilleure solution.

Note : Nous avons rapidement été bannis de RTK2GO car, après avoir terminé notre inscription, nous avons lancé STRSVR et l'avons pointé sur rtk2go.com avec le mot de passe temporaire. Très vite (1 à 2 mn), notre compte et notre mot de passe ont été activés. Ce qui signifie que notre diffusion sur RTK2GO avec le mot de passe temporaire est devenue invalide. Après environ 60 s de connexions non valides par STRSVR (parce que le mot de passe est passé de temporaire à permanent), notre IP a été interdite pendant quelques minutes, puis des heures. Notre conseil est d'attendre quelques minutes. **Ne** connectez **pas** STRSVR à RTK2GO en utilisant le mot de passe temporaire. Attendez simplement le courriel de confirmation de RTK2GO, répondez par « oui, je ne suis pas un robot » et attendez le courriel de RTK2GO indiquant que votre point de référence et votre mot de passe sont valides. À ce stade, lancez STRSVR avec vos identifiants.

Avec votre point de référence et votre mot de passe, pointez STRSVR sur RTK2GO (**fig. 22**). Comme nous envoyons des données vers un serveur NTRIP, nous n'avons pas besoin d'ouvrir le port 2101 sur notre réseau local.

Figure 25. L'antenne SparkFun 915-MHz.

Appuyez ensuite sur « *Start* ». Après quelques secondes, les voyants devraient devenir verts, indiquant que les données du ZED-F9P sont correctement transmises à RTK2GO (**fig. 23**). Bien. Donnez-vous environ 60 s, puis ouvrez un navigateur et allez sur rtk2go.com:2101. Vous devriez voir la liste des points de référence actuels. Votre point de référence devrait s'y trouver. Sinon, vérifiez que vous avez entré le bon mot de passe, que la base est correctement configurée avec les messages RTCM activés, et en mode TIME.

Remarque : lorsque vous fermez le STRSVR, ces paramètres sont enregistrés. Si vous lancez *strsvr.exe -auto*, il démarrera STRSVR et lancera automatiquement la distribution (**fig. 24**).

Félicitations ! La ligne d'arrivée est en vue. Vous pouvez récupérer une panoplie de géomètre, aller sur le terrain, et utiliser SW Maps pour vous connecter à vos données de correction en utilisant le client NTRIP intégré. Vous pouvez également utiliser U-Center comme client.

Astuce subtile : ajouter une radio

Bien sûr, nous avons une antenne de 915 MHz sur le toit de SparkFun (**fig. 25**). Alors pourquoi ne pas la brancher ? Contrairement à la petite antenne GNSS de la section précédente, cette antenne dispose d'une protection *indispensable contre la* **foudre**.

STRSVR est très puissant. Vous pouvez transmettre vos données RTCM vers plusieurs destinataires, pas seulement à un serveur NTRIP. Nous avons connecté une radio de 100 mW à 915 MHz [16] à l'USB. Elle est énumérée comme un port COM. Nous pouvons ensuite ajouter ce port COM (n'oubliez pas de

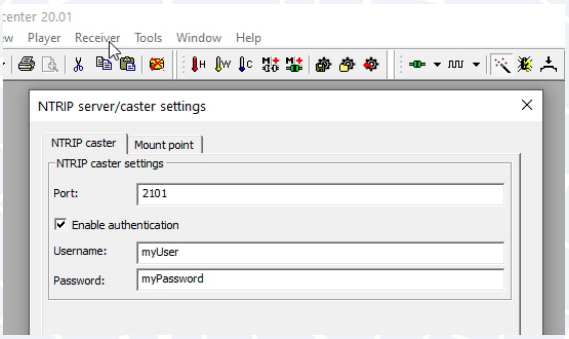


Figure 20. Dans U-Center, sélectionnez Receiver -> NTRIP Server/Caster.

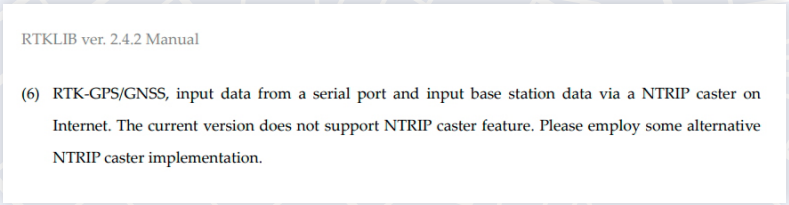


Figure 21. Il y a écrit « caster » ! Ne vous laissez pas berner.

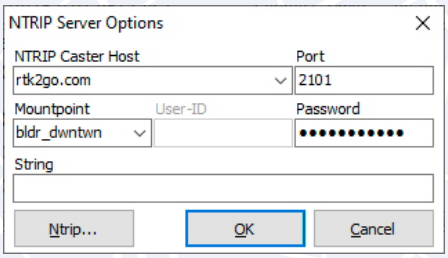


Figure 22. Sélection de RTK2GO comme service de distribution.

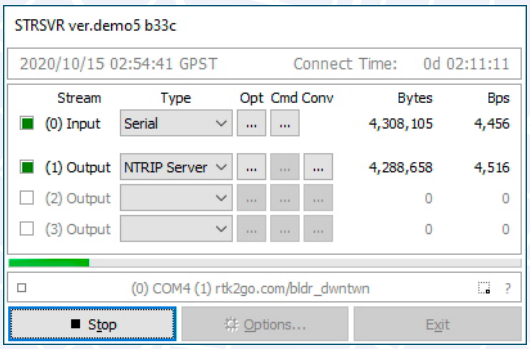


Figure 23. STRSVR envoyant des données à RTK2GO.

```
STR;BBDS_Leica4G;Bridgetown;RTCM 3.2;1006(15),1013(90),1033(15),1230(15),4092(1);;;SNIP;BRB;13.17;-59.52;1;0;sNTRIP;none;N;N;3120;none
STR;BBDS_RTCM_MSM5;Saint John;RTCM 3.2;1006(15),1008(15),1013(90),1033(15),1075(1),1085(1),1095(1),1125(1),1230(15);0;GAL;SNIP;BRB
STR;BBDS_RTCM_MSM7;Bridgetown;RTCM 3.2;1006(15),1008(15),1013(90),1033(15),1077(1),1087(1),1097(1),1127(1),1230(15);;GPS+GLO+GAL
STR;bldr_dwntwn;Boulder, CO;RTCM 3.2;1005(1),1074(1),1084(1),1094(1),1124(1),1230(10);0;GPS+GLO+GAL+BDS;SNIP;USA;40.02;-105.28;1;0;
STR;bldr_SparkFun1;Boulder, CO;RTCM 3.2;1005(1),1074(1),1084(1),1094(1),1124(1),1230(5);;GPS+GLO+GAL+BDS;SNIP;USA;40.09;-105.19;1;0
STR;Blotnica;Przemet;RTCM 3.2;1005(1),1074(1),1084(1),1094(1),1124(1),1230(1);;GPS+GLO+GAL+BDS;SNIP;POL;51.99;16.30;1;0;sNTRIP;none;N;N;3120;none
STR;BPACA;Blacka Palanka;RTCM 3.2;1005(30),1074(1),1084(1),1094(1);;GPS+GLO+GAL;SNIP;SRB;45.25;19.41;1;0;sNTRIP;none;N;N;3120;none
STR;BPGAJIC;Blacka Palanka;RTCM 3.2;1005(30),1074(1),1084(1),1094(1);;GPS+GLO+GAL;SNIP;SRB;45.26;19.40;1;0;sNTRIP;none;N;N;3120;none
```

Figure 24. Deux points de référence en plein travail !

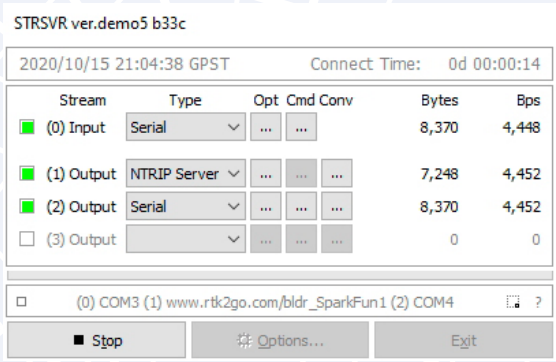


Figure 26. STRSVR mis en place pour desservir deux ports COM.

Figure 27. Modification des paramètres de la radio par commandes AT.

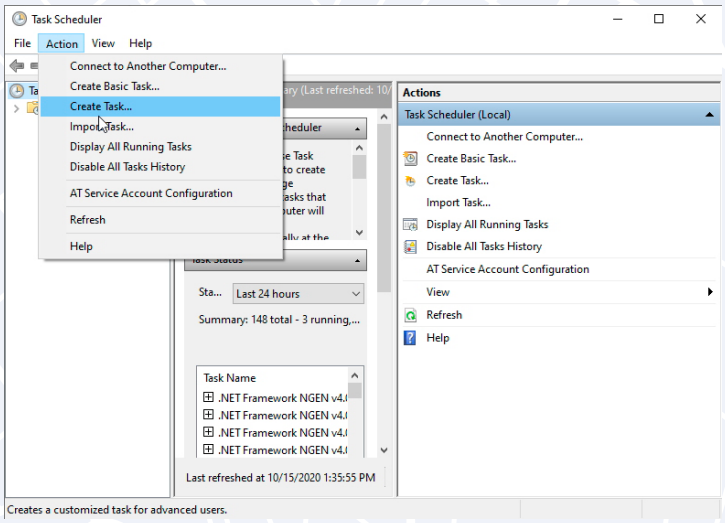


Figure 28. Création d'une nouvelle tâche dans le planificateur de tâches de Windows.

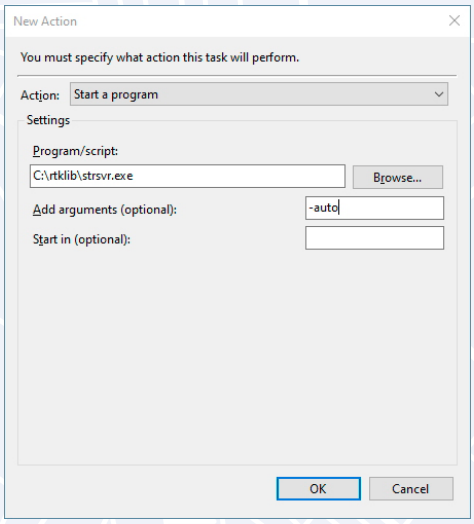


Figure 29. STRSVR doit démarrer -auto(matiquement) !

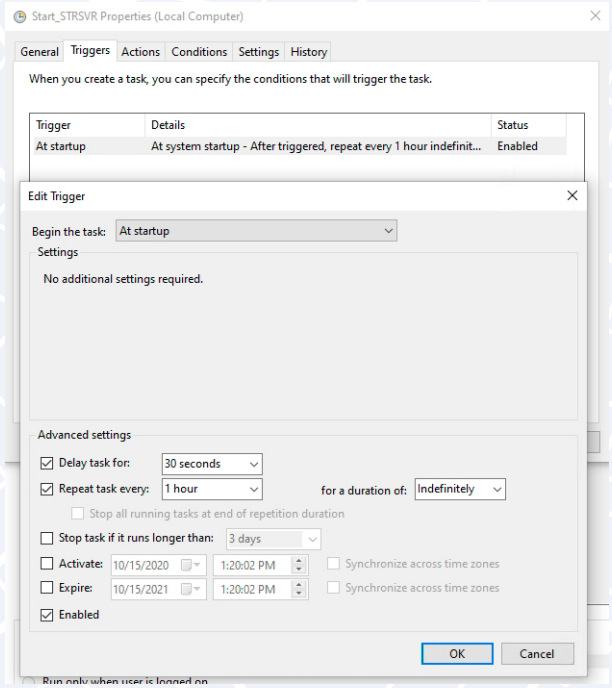


Figure 30. Paramètres « Report maximal de la tâche » et « Répéter la tâche toutes les : » pour le démarrage automatique de STRSVR.

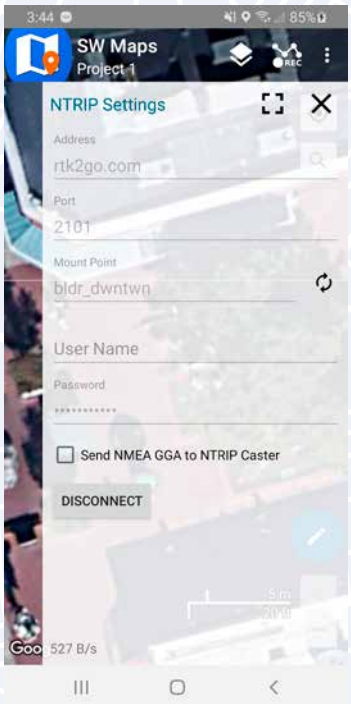


Figure 31. Utilisation de SW Maps pour obtenir des données de correction via NTRIP.

régler le débit en bauds à 57600 bps pour qu'il corresponde à celui de la radio) à STRSVR afin que les données RTCM soient transmises à la *fois au* serveur NTRIP et à la radio (**fig. 26**). Cela nous permet d'utiliser une connexion radio pour la RTK locale et de passer au cellulaire si nous nous trouvons hors de portée radio.

Q. Pourquoi ne pas souder la radio à l'UART2 ?

R. En connectant la radio à l'USB, nous pouvons configurer la radio [17] via une fenêtre de terminal (**fig. 27**). Si elle était directement connectée à l'UART2 sur le ZED-F9P, les données de correction seraient transmises, mais ainsi nous pouvons modifier les paramètres AIR_SPEED de la radio [18] et d'autres pour étendre la portée.

Lectures complémentaires

En savoir plus sur RTK ? Consultez ces tutoriels complémentaires :

1. Démarrage avec U-Center pour u-blox. Apprenez les trucs et astuces pour utiliser l'outil logiciel u-blox pour configurer votre récepteur GPS.
   https://learn.sparkfun.com/tutorials/getting-started-with-u-center-for-u-blox

2. Guide de connexion GPS-RTK2. Obtenez une précision jusqu'au diamètre d'une pièce de dix cents avec le nouveau ZED-F9P de u-blox.
   https://learn.sparkfun.com/tutorials/gps-rtk2-hookup-guide

Planifier une tâche

Il faut que STRSVR démarre automatiquement. STRSVR mémorisera ses paramètres et démarrera automatiquement avec les derniers paramètres utilisés en exécutant *strsvr.exe -auto*. Maintenant, créons une tâche dans Windows pour nous assurer qu'elle démarre à chaque mise sous tension. Ouvrez l'application *Planificateur de tâches* de Windows et créez une nouvelle tâche (**fig. 28**). Ensuite, demandez à la tâche d'exécuter STRSVR avec '-auto' (**fig. 29**). Les principales actions que la commande optionnelle *-auto* doit exécuter sont de démarrer la tâche au démarrage sans avoir besoin de se connecter, et de la retarder de 30 à 60 s. Nous avons constaté que cette tâche (STRSVR) ne démarrait pas au démarrage de Windows, peut-être parce que le ZED-F9P et/ou les ports COM radio n'étaient pas encore énumérés. Un retard de 30 s ou plus corrige le problème

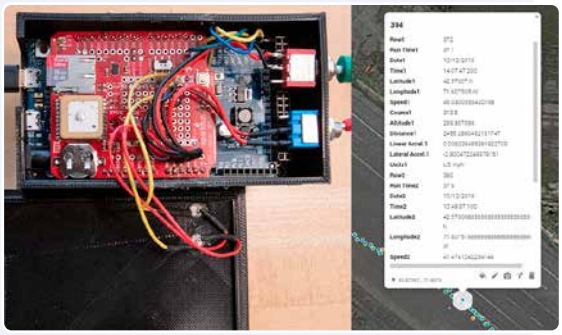
Exemples d'utilisation communautaire

#1. Le GPS RTK donne vie aux rêves des courses d'orientation.

Avid orienteer Don Bayly had been using a single-frequency GPS-Glonass receiver that was Depuis plusieurs années, Don Bayly, orienteur passionné, utilisait un récepteur GPS-Glonass à fréquence unique, avec une précision généralement inférieure à 5 m. Cependant, des ajustements étaient souvent nécessaires en raison des variations des positions observées à différents moments. Lorsqu'il a commencé à faire des recherches sur des récepteurs GPS plus précis, il a découvert que cela coûtait des dizaines de milliers de dollars pour un instrument RTK de précision centimétrique, et que même un instrument SIG d'un mètre de précision coûtait environ 2000 $.

C'est alors qu'il a trouvé la carte SparkFun GPS-RTK2 - ZED-F9P (Qwiic). Pour 200 $, avec une précision de 10 mm, en trois dimensions, et des fonctions de récepteur et de station de base, Bayly était intrigué. Au départ, il pensait que l'absence de stations de base de l'UNAVCO au Canada rendrait la carte impossible à utiliser. Cependant, au fil de ses recherches, il a découvert le service NTRIP du BKG et ses stations de base dans le monde entier, aussi à Calgary où il vit. C'est alors qu'il a décidé d'essayer le SparkFun GPS-RTK.

#2. Utiliser le GPS u-blox pour une plus grande précision



Anker Berg-Sonne a remarqué que le temps passé à écrire des données dans la micro-SD était critique. Pour que l'Arduino Mega 2560 puisse suivre le flux de données à 10 Hz du GPS, il a dû effectuer un décodage minimal, réduire au maximum les messages envoyés par le GPS et envoyer les données à la micro-SD avec le moins de traitement possible.

Le suivi des données d'autocross représente un défi unique, car le circuit des cônes change à chaque course. Berg-Sonne a donc fait fonctionner l'enregistreur en deux modes. Dans l'un, il a marqué les coordonnées GPS du circuit des cônes en appuyant sur un bouton de l'enregistreur à chaque position de cône. Dans l'autre, les données étaient enregistrées chaque fois que la vitesse de la voiture dépassait 15 km/h.

« Utiliser la vitesse des voitures pour déclencher l'enregistrement s'est avéré efficace », constate Anker. « La plupart des enregistreurs de données commerciaux exigent que vous activiez et désactiviez manuellement l'enregistrement, ou que vous marquiez très soigneusement le début et la fin du parcours pour activer et désactiver l'enregistrement. L'autocross est très intense et il est très facile d'oublier de démarrer l'enregistreur au début d'un parcours ! Avec mon enregistreur, vous n'avez pas à vous en inquiéter ».

Accessoires

Vous recherchez les principaux accessoire mentionnés dans cet article ?
Voici ce que proposent SparkFun et Elektor !

GNSS Multi-Band Magnetic Mount Antenna - 5m (SMA)
www.elektormagazine.fr/esfe-en-diygnss1

SparkFun 915 MHz radio module
www.elektormagazine.fr/esfe-en-diygnss2

Note: si nécessaire, remplacer le module radio SparkFun 915 MHz [16] par un type 868 MHz équivalent.

(**fig. 30**). Et si jamais le programme s'arrête ou se plante pour une raison quelconque, le programme devrait, toutes les heures, redémarrer STRSVR, s'il n'est pas déjà en cours d'exécution.
Une fois votre tâche définie, réinitialisez votre mini-PC et vérifiez qu'il démarre automatiquement STRSVR et commence à émettre.

Déploiement !

C'est tout ! Vous devriez maintenant pouvoir pointer le client NTRIP de votre choix sur RTK2GO.com, port 2101, avec votre point de référence et votre mot de passe et recevoir les données de correction de votre base vers un nombre quelconque de récepteurs sur le terrain dans un rayon de 10 km de votre base. Nous conseillons vivement SW Maps [19] (**fig. 31**) car il est incroyablement facile de récupérer les données de correction de RTK2GO sur le réseau cellulaire et de les renvoyer automatiquement à un ZED-F9P configuré en récepteur. La mise en place d'une station de base de correction dédiée peut être un peu laborieuse, mais avec la bonne configuration, la base devrait fonctionner pendant plusieurs mois ou années sans supervision. J'ai eu beaucoup de plaisir à apprendre sur RTK et la topographie, mais j'ai appris aussi à me contenter de ne pas savoir où exactement **tel** endroit se trouve dans le monde. L'espace est relatif et changeant, tout comme le temps. La plaque tectonique nord-américaine se déplace de 2 cm par an [20]. Il faut savoir profiter du moment présent.

200660-03
(VF: Denis Lafourcade)

u-blox and SparkFun

Les techniques sont complexes, et pour obtenir des produits utiles, les entreprises doivent coopérer. Depuis plusieurs années, SparkFun s'est associé à u-blox pour la conception innovante de cartes de positionnement, de communication et de chronométrage. u-blox a toujours repoussé les limites de l'innovation et SparkFun facilite l'utilisation de sa technologie et accélère le prototypage et la R&D.

LIENS

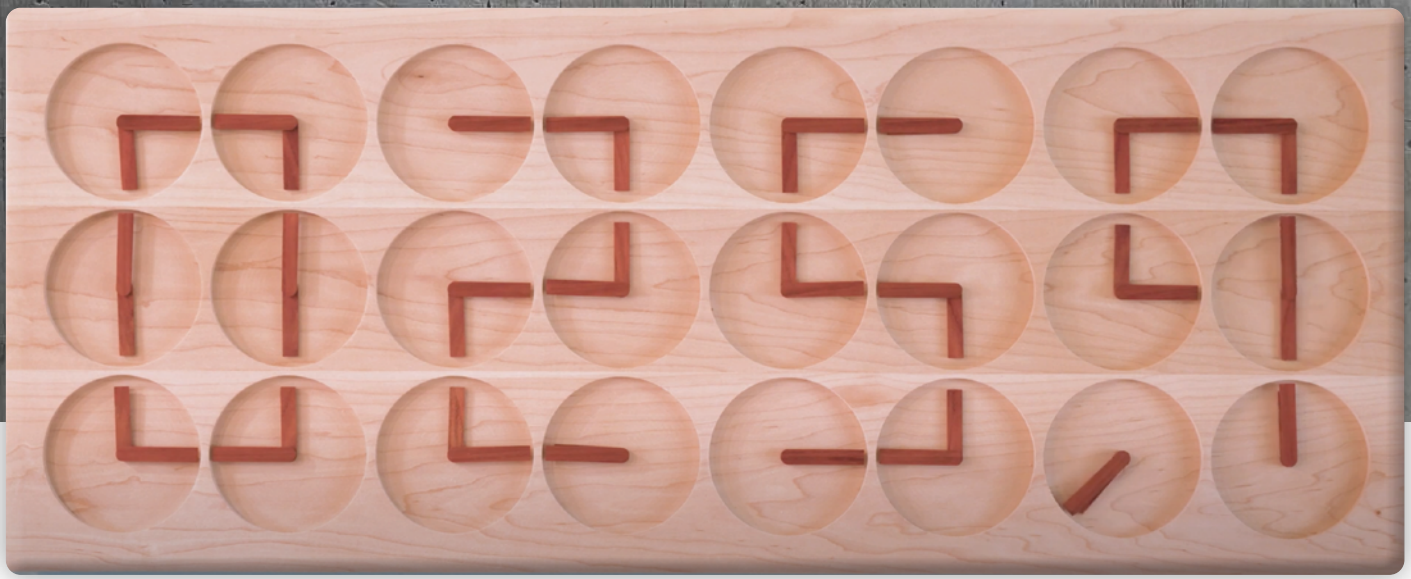
[1] Données de correction du RTCM accessibles au public : https://bit.ly/SF-RTCM-correction
[2] Créez votre propre base temporaire : https://bit.ly/SF-rover-base
[3] Pour commencer avec U-Center : https://bit.ly/SF-U-Center
[4] Tutoriel : Qu'est-ce que le GPS RTK ? : https://bit.ly/SF-GPS-RTK
[5] Mise en place d'un système RTK de base pour les récepteurs : https://bit.ly/SF-rover-base
[6] L'excellent HowTo de Gary Miller sur les PPP : https://gpsd.gitlab.io/gpsd/ppp-howto.html
[7] Le PPP d'Emlid : https://docs.emlid.com/reachrs/common/tutorials/ppp-introduction/
[8] Suelynn Choy au sujet de GNSS PPP : https://bit.ly/GNSS-PPP
[9] Antenne U-Blox : https://www.elektormagazine.fr/esfe-en-diygnss1
[10] RTKLIB: http://www.rtklib.com
[11] rtklibexplorer: http://rtkexplorer.com/downloads/rtklib-code/
[12] Service CSRS-PPP : https://bit.ly/NRCAN-PPP
[13] OPUS: https://www.ngs.noaa.gov/OPUS/
[14] ECEF: https://en.wikipedia.org/wiki/ECEF
[15] Coffret « Orbit » étanche : https://bit.ly/orbit-enclosure
[16] Module radio SparkFun 915 MHz : https://www.elektormagazine.fr/esfe-en-diygnss2
[17] Configuration radio sur une fenêtre de terminal : https://bit.ly/ardupilot-radio
[18] Réglage de la radio AIR_SPEED : https://bit.ly/google-sw-maps
[19] SW Maps: https://bit.ly/google-sw-maps
[20] Mouvement des plaques tectoniques en Amérique du Nord : https://bit.ly/nrcan-tectonic

Espace Elektor en ligne pour l'article :
www.elektormagazine.fr/esfe-en-diygnss

CLOCKCLOCK

horloge récursive

Arduino donne l'heure avec un FPGA



Justin Rajewski (Alchitry) (États-Unis)

Quelle heure est-il ?
L'heure d'attaquer un magnifique projet Alchitry !
Cet article va vous guider dans la construction d'une Clockclock dont les moteurs sont commandés par la carte de développement FPGA d'Alchitry Au.

Qu'est-ce qu'une *ClockClock* ? C'est 'une horloge faite d'horloges ! Une méta-horloge en somme. Pour former chacun des (grands) chiffres de l'heure affichée par *ClockClock*, j'utilise plusieurs horloges à aiguilles ordinaires. L'idée n'est pas de moi. Je suis tombé sur ce concept astucieux il y a déjà des années et j'ai toujours pensé que ce serait un excellent projet de démonstration de FPGA tant il nécessite de signaux de contrôle. Vous trouverez l'horloge originale de *Humans since 1982* ici [1].

Ce projet est une excellente démonstration du FPGA pour différentes raisons. D'abord, la *ClockClock* nécessite 48 moteurs pas à pas pour mouvoir les 2 aiguilles indépendantes de ses 24 horloges. L'utilisation d'une commande standard implique deux signaux (pas + sens) de commande par moteur soit 96 sorties. Pour économiser l'énergie, mieux valait pouvoir désactiver les pilotes quand l'horloge est immobile. Cela a ajouté quatre sorties supplémentaires (une pour chaque «chiffre» de l'horloge). Je voulais également utiliser un Arduino pour créer les animations, plus faciles à obtenir en programmation qu'en matériel. J'ai choisi I²C pour dialoguer avec l'Arduino via le connecteur *Qwiic* de l'Alchitry Au, donc 2 broches de plus. Au total nous atteignons 102 E/S. Or, la carte Alchitry Au a exactement 102 broches d'E/S [2].

Ce projet a le mérite non seulement d'épuiser la foule d'E/S du FPGA, mais aussi d'utiliser le connecteur *Qwiic* du FPGA d'une manière peu conventionnelle : ici, le FPGA agit comme périphérique et non comme contrôleur. L'Arduino le contrôle et lui envoie toutes les commandes. Ce sera un modèle utile pour de nombreux projets. Certaines tâches, très simples sur le plan du logiciel, peuvent nécessiter un matériel très complexe. L'inverse est également vrai. En associant un microcontrôleur et un FPGA, nous obtenons le meilleur des deux mondes. Le connecteur *Qwiic* des deux cartes facilite la tâche.

Matériel et outils nécessaires

Pour suivre cet article, vous aurez besoin du matériel énuméré dans l'encadré **Accessoires**. Vous n'aurez peut-être pas besoin de tout, selon ce que vous avez déjà. Composez votre panier, lisez le guide et rectifiez le panier si nécessaire.
Il y a plusieurs façons d'usiner les pièces pour ce projet. Voici les outils que j'ai utilisés :

- imprimante 3D ;
- fraiseuse CNC Shapeoko XXL ;
- scie à ruban, raboteuse et ponceuse orbitale pour le bois.

Et aussi :

- 48× moto-réducteurs pas-à-pas [3] ;
- 48× modules drivers de moteur pas à pas StepStick avec dissipateur thermique [4] ;
- 1× UBEC BEC ajustable UBEC 2-6S pour drone quadcopter RC [5] ;
- 1× Alimentation à découpage AC-DC sous boîtier [6].

Suggestions de lecture

Si vous connaissez mal le système *Qwiic*, nous vous conseillons de lire la documentation sur le site www.sparkfun.com/qwiic. Nous vous conseillons également de consulter ces tutoriels avant de continuer :

- *Programmation d'un FPGA* [7]. Notions élémentaires pour s'approprier les FPGA.
- *Comment un FPGA fonctionne-t-il ?* [8]. Le b.a.-ba du FPGA (Field Programmable Gate Array = réseau de portes programmable in-situ).
- *Premier projet FPGA - Getting Fancy with PWM* [9]. 1er projet utilisant le FPGA embarqué d'Alchitry pour manipuler la modulation PWM.

Construction physique

Je serai bref car mon sujet est le FPGA, pas le travail du bois. Suivez les liens ci-dessous pour accéder aux fichiers du projet et les télécharger.

- Fichier CAO (Fusion 360) [10].
- Alchitry (FPGA) [11].
- Code Arduino (ZIP) [12]

Pour débuter ce projet il fallait imaginer comment fabriquer l'un des mouvements de l'horloge. J'avais besoin de deux moteurs pas à pas et d'un moyen de les coupler à deux arbres de sortie concentriques. J'ai d'abord trouvé sur Amazon ces moteurs pas à pas miniatures (à peine 8 × 9,2 mm !). J'ai conçu un réducteur double (2 couronnes et 2 pignons) emboîté sur les moteurs (**fig. 1**).

Malheureusement, ces petits moteurs ne parvenaient pas à faire tourner les engrenages. Leur couple est minuscule et, quand j'ai appliqué une tension d'alimentation suffisante pour qu'ils tournent, ils ont chauffé au point de faire fondre les pièces en PLA imprimées en 3D.
J'y ai renoncé, et j'ai commandé un lot de moteurs pas à pas 28BYJ-48 [13], un peu plus gros mais assez petits pour l'horloge. Leur réducteur interne offre un couple fort. Qui dit réducteur interne dit aussi couple résistant interne suffisant pour conserver la position atteinte après coupure du moteur.
Le mouvement que j'ai conçu est à entraînement direct de l'aiguille des minutes et l'aiguille des heures est entraînée par un engrenage, ainsi le moteur est déporté latéralement.
Je raconte la suite de la construction matérielle de notre *ClockClock* en images assorties de brefs commentaires. Profitez du spectacle !

FPGA

Avant de commencer un projet FPGA, quel qu'il soit, il est important de définir ce que nous en attendons. Dans le cas présent, je devais créer un circuit acceptant des commandes par I²C et faire tourner les moteurs en conséquence.
J'ai opté pour les 2 commandes suivantes : un nombre de pas et une valeur équivalant à la période entre pas. Au départ, j'avais pensé rendre le contrôleur plus souple en modulant automatiquement le nombre de pas, mais cela s'est révélé inutile et ne faisait que compliquer la coordination entre les aiguilles.
Je voulais aussi constituer une file d'attente pour une série de commandes. La synchronisation du *Qwiic* n'aurait pas d'importance, car chaque commande serait exécutée l'une après l'autre.
Il fallait enfin que le projet résolve la chronologie de l'envoi des pas pour activer/désactiver les moteurs en conséquence et économiser l'énergie.

L'animateur

Pour commencer, j'ai créé un module contrôlant un seul moteur pas à pas. Ce module accepte une commande permettant de faire un nombre de pas donné avec une temporisation donnée entre chaque pas. Il délivre ensuite les signaux de sens et de pas appropriés pour la commande du moteur pas à pas. Le code du module est donné par le **listage 1**.
La première chose à noter est que le contrôleur pas à pas utilisé exige que l'entrée du sens soit stable pendant au moins 200 ns avant et après le front montant de l'entrée du pas. Il exige également que

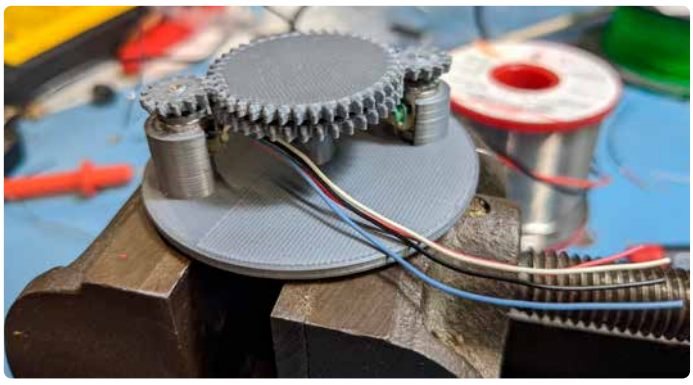


Figure 1 : Moteur pas à pas miniature (Amazon) avec 2 couronnes et 2 pignons montés à la presse.

l'impulsion de pas ait une durée minimale haute ou basse de 1 µs. La largeur d'impulsion des pas est facilement obtenue avec le module `pulse_extender` de la bibliothèque de composants. Ce module reçoit des impulsions isolées et les prolonge jusqu'à la durée spécifiée. J'ai fixé la durée à 2 µs par commodité et sécurité. Dès qu'une nouvelle commande d'animation est reçue, la sortie sens est définie et le module attend que le dépassement du `dirCt`. Ce compteur contient 256 valeurs et avec une horloge à 100 MHz, cela signifie qu'il attend 2,56 µs. C'est nettement plus que les 200 ns nécessaires, mais cela garantit la synchronisation avec les longs fils. Le resserrement de la synchronisation ici n'augmenterait pas non plus les performances.

Le changement d'état du pas incrémente un compteur et fait un pas à chaque dépassement. Ce compteur a un prédiviseur par 8, de sorte que chaque incrément de `delayCycles` dans la commande d'animation ajoute 256 cycles de temporisation. Ce prédiviseur permet de limiter `delayCycles` à 16 bits tout en préservant une gamme de vitesses étendue. Même avec ce prédiviseur, j'ai trouvé que le `delayCycles` le plus bas utilisable sans risque est d'environ 760. Cela correspond à 8 s pour une rotation complète.

Porte d'activation (*enable gate*)

Le module suivant est la porte d'activation. Ce module commande activation et blocage du drapeau *nouvelle animation* des animateurs pendant l'activation des moteurs. Il s'assure aussi qu'après l'animation, les moteurs restent activés assez longtemps pour que le dernier pas soit achevé.

Le module prend en charge les drapeaux de *nouvelle animation* en attente pour 12 des moteurs. Il active ensuite les moteurs et attend 42 ms, afin que l'électronique remette les moteurs sous tension et que les moteurs se stabilisent. Après cette période, il permet au drapeau *animation en attente* de passer aux animateurs.

Tant qu'un animateur du groupe fonctionne, il maintient les moteurs activés. Une fois que le dernier a terminé, il maintient les moteurs activés pendant encore 42 ms avant de les désactiver. Pour le code du module, voir le **listage 2**.

Si le module est inactif, `onCtr` est à 0 et `offCtr` est au maximum. Si une *animation en attente* est détectée (la FIFO n'est pas vide), la sortie *enable* est activée et `onCtr` est incrémenté à chaque cycle. Dès que `onCtr` est plein, les drapeaux `new_animation` sont transmis. Dès que toutes les animations sont réalisées, `offCtr` est incrémenté. Dès sa valeur maximale atteinte, `onCtr` est réinitialisé, ce qui désactive les moteurs.

Une ligne intéressante à regarder est la première du bloc *always* :

```
running = |(animator_busy | ~fifo_empty);
```

Sans pratique des opérateurs de réduction bit à bit, cette ligne paraîtra absconse. Son but est de prendre les 12 signaux `animator_busy` et les 12 signaux `fifo_empty` et de les transformer en un seul bit. Raisonnons d'abord sur un cas simple. Un seul moteur fonctionne si la FIFO n'est pas vide ou si elle est occupée à cet instant. L'expression `animator_busy | ~fifo_empty` résoud ce cas. La barre verticale | ou *pipe* (= tuyau) en anglais) est l'opérateur OU bit à bit. Il réalise simultanément et indépendamment un OU entre les bits de même rang des deux opérandes. Le tilde (~) réalise l'inversion de chaque bit. Il inverse ici chacun des bits de `fifo_empty`.

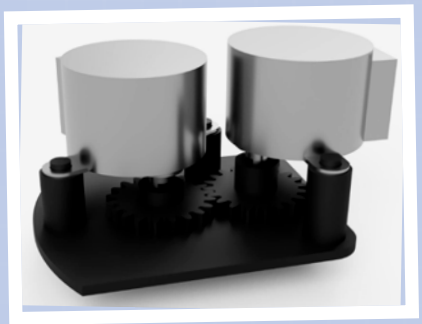
Le résultat de ces opérations est un signal de 12 bits de large qui indique quand chaque animateur est en train de tourner. Cependant, nous devons le condenser en un seul bit. À cet effet, l'opérateur de réduction OR est utilisé. L'opérateur pipe, s'il est placé **seul** devant une valeur, exécute un OU entre **tous** les bits du signal pour les **réduire** à un seul bit. Dans notre cas, cela signifie que si un ou plusieurs des moteurs tournent, running vaut 1.

Plus loin dans le module, j'utilise l'opérateur de réduction AND pour vérifier si tous les bits d'un signal sont à 1 (la *valeur maximale*). Cela fonctionne de la même manière que l'opérateur de réduction OR, mais exécute un ET entre tous les bits. Le résultat est 1 s'ils valent

L'horlogerie de ClockClock

Cette ébauche d'horloge élémentaire montre les deux arbres de sortie. L'arbre central plus long est directement couplé au moteur aligné avec lui. L'arbre extérieur porte une couronne entraînée par un pignon monté sur l'arbre du 2e moteur. Les aiguilles des heures et minutes sont montées par friction sur ces deux arbres.

La friction douce permet de repositionner les aiguilles de l'horloge dans une position neutre avant de la mettre sous tension. C'est important car même si un moteur pas à pas permet une commande de déplacement précise, on ignore toujours la position d'origine.



Les moteurs, simplement collés (cyanoacrylate) sur des entretoises. Toutes les pièces ont été imprimées en PLA noir sur ma Prusa MK3S.

Le gros avantage de l'Au, c'est le bus Qwiic qui permet de communiquer avec un microcontrôleur

tous 1 et 0 dans tout autre cas. L'accent circonflexe (^) est utilisé pour effectuer une réduction XOR (OU exclusif) qui vaudra 1 s'il y a un nombre impair de 1.

Qwiic

Nous allons maintenant nous intéresser au module au_top qui s'occupe de l'interface *Qwiic* et qui coordonne tout. Décortiquons-le **listage 3**.

L'interface *Qwiic* est gérée par le module `i2c_peripheral`. Ce module est un peu compliqué puisqu'il décompose les signaux de démarrage/arrêt et demande qu'on lui indique quand il doit accepter ou envoyer des données.

Dans notre cas, nous pouvons le simplifier en nous contentant de lire les données. Les drapeaux importants deviennent `rx_valid` qui nous indique qu'un nouvel octet a été lu et *stop* qui indique que la transaction I²C a été arrêtée et que nous devons la réinitialiser. La sortie `rx_data` a la valeur de l'octet lu lorsque rx_valid vaut 1.

Si vous voulez réagir à quelque chose, vous devez surveiller les drapeaux start, next, et write. Au cycle d'horloge suivant, vous pouvez mettre `tx_enable` à 1 et fournir des données à envoyer sur `tx_data`. Le module écrira alors cet octet au lieu d'en scruter un.

Le drapeau de départ signale que votre identifiant ID a été détecté sur le bus. En même temps qu'il est activé, write vous dira si le dernier bit de l'octet ID indiquait une lecture (0) ou une écriture (1). Là encore, nous pouvons ignorer tout cela pour ce projet.

Dans le protocole utilisé pour chaque transaction, le 1er octet est l'adresse suivie des données de la commande. Pour les adresses 0 à 47, 4 octets sont attendus. Les 2 premiers codent le nombre de pas et les 2 autres, la valeur de la tempo. L'adresse 8hFF est particulière car elle n'attend qu'un octet après elle et est utilisée pour allumer les LED sur l'*Au*. C'est utile pour tester le bus *Qwiic*.

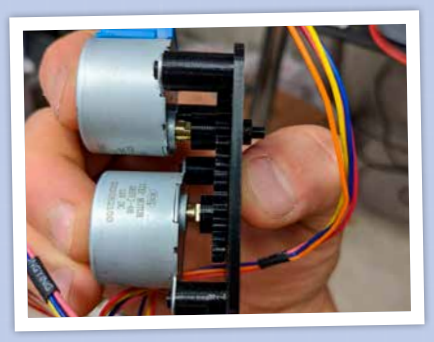
J'ai également fait en sorte que vous n'ayez pas besoin de démarrer/arrêter la transaction I²C pour chaque animation. Chaque paquet de 5 octets est une animation valide, qui tournera en boucle après la réception du dernier octet. Cela permet d'envoyer aux 48 moteurs une nouvelle animation en une seule transaction.

FIFO

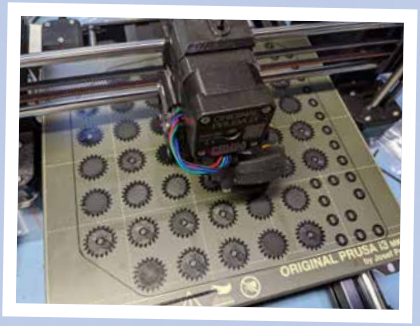
Ce projet comporte 48 FIFO pour conserver des animations supplémentaires pendant que les animateurs sont occupés. Celles-ci sont créées à partir du composant `fifo` de la bibliothèque de composants. Chaque FIFO a une largeur de 32 bits et une profondeur de 128 entrées. Les 32 bits sont répartis en 2 x 16 entre le délai et le nombre de pas. Note : 128 entrées c'est trop pour l'utilisation actuelle, mais cela permettrait d'empiler de nombreuses animations courtes plus rapides et/ou avec gradients de vitesse. L'Au a de toute façon largement assez de RAM intégrée pour le faire.

La FIFO est de type FWFT (*First-Word Fall-Through*) : si *empty* vaut 0, une donnée suit et la valeur de cette donnée est déjà disponible sur `dout`. Mettre `rget` à 1 supprime l'entrée et affiche l'entrée suivante au cycle d'horloge suivant.

Pour fournir des données à la FIFO, il suffit de mettre les données sur `din` et `wput` à 1 ; mieux vaut vérifier que full ne vaut pas 1, sinon les données peuvent être ignorées.



Voici imprimé et assemblé le 1er mouvement que j'ai finalisé.



Dès lors que j'avais un modèle fonctionnel, j'ai pu lancer le lot de 24 dont j'avais besoin.

Toutes les pièces rangées dans une boîte en carton et prêtes à être assemblées.

Attribution des bits

Au début du bloc always, il y a pas mal de manipulations de tableaux et de bits. Dans Lucid, il est facile de créer des tableaux de modules et de regrouper leurs ports en tableaux. Parfois dans notre projet, par ex. sorties de pas, nous pouvons directement remplir ces tableaux car les bits s'alignent parfaitement.

Parfois, nous devons les diviser en sous-sections. Lorsque cela se produit, il est pratique d'utiliser des boucles for. Souvenez-vous que les boucles ne peuvent pas être réalisées par le matériel et que le nombre d'itérations doit être fixé pour que la phase de «compilation» puisse les réaliser. Ces boucles ne sont qu'une convention d'écriture qui raccourcit le texte nécessaire. Par ex., la première boucle passe par 4 itérations, i allant de 0 à 3. À la première itération, la 1ère ligne sera évaluée comme suit :

```
gates.fifo_empty[0] = ani_fifos.empty[0+:12];
```

Le sélecteur de bits `[0+:12]` signifie qu'il faut sélectionner 12 bits à partir du bit 0. Ainsi, les bits 0 à 11 sont sélectionnés. À l'itération suivante, nous aurons :

```
gates.fifo_empty[1] = ani_fifos.empty[12+:12];
```

Ici, la deuxième porte d'activation reçoit les bits 12-23. Les 4 itérations seront :

```
gates.fifo_empty[0] = ani_fifos.empty[0+:12];
gates.fifo_empty[1] = ani_fifos.empty[12+:12];
gates.fifo_empty[2] = ani_fifos.empty[24+:12];
gates.fifo_empty[3] = ani_fifos.empty[36+:12];
```

Cela créera quatre fois le même circuit dans le FPGA. Je suis sûr que vous admettrez que la saisie et la maintenance du code sont lourdes. Il est très courant d'utiliser les sélecteurs start/width pour les boucles au lieu des sélecteurs start/stop. Cela vient de l'impossibilité d'utiliser les sélecteurs de bits start/stop avec des valeurs non constantes.

Le sélecteur start/width utilisé ci-dessus garantit que la largeur de la sélection est toujours de 12 bits. Il est impossible de réaliser un signal qui change de largeur dans le matériel car il est impossible de créer ou supprimer spontanément des connexions. Le `+:` sélectionne le sens croissant du sélecteur. Le `-:` fait l'inverse, c.-à-d. le sens décroissant. Cela sélectionne le bit de départ et ceux qui lui sont inférieurs. Par ex. `[11-:12]` donne le même résultat que `[0+:12]`. Tous deux sélectionnent les bits 0 à 11.

Affectation des broches

Vous vous demandez peut-être comment les signaux `step` et `dir` sont-ils affectés aux broches E/S de l'*Au*. Cette correspondance est définie dans un fichier de contraintes, *clockclock.acf*. L'extension «acf» représente les initiales d'*Alchitry Constraint File*. Ce format très simple permet de spécifier les noms des broches comme étant les broches des cartes Alchitry au lieu du FPGA. Par exemple, A2 correspond à la deuxième broche du connecteur supérieur gauche (rangée A) sur la carte *Au*. Si vous ouvrez ce fichier, vous verrez des lignes qui ressemblent à ceci :

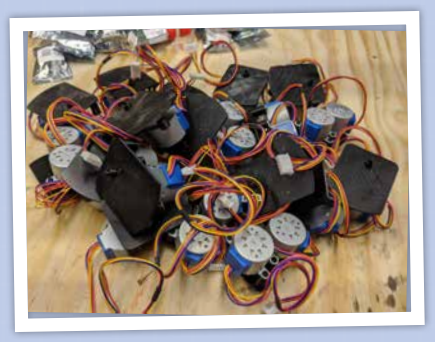
```
pin step[0] A2;
pin dir[0] A3;
```

Chaque port d'E/S doit être associé à une broche physique. Le format est le mot-clé *pin* suivi du nom du signal et de l'emplacement de la broche physique. Les mots-clés `pullup` ou `pulldown` permettent respectivement d'ajouter une résistance interne de rappel au + ou à la masse. Cependant, le `pulldown` est ignoré sur la *Cu* car le FPGA Lattice n'a pas de résistance interne de rappel à la masse.

La plupart des broches d'un FPGA sont totalement interchangeables et le brochage que j'ai utilisé pour l'horloge était arbitraire, à l'exception des signaux *Qwiic* puisqu'ils sont câblés sur le connecteur *Qwiic*. Ce qui était important pour ce projet, c'était qu'ils soient tous droits.

Installation du logiciel et programmation

Note : Cet exemple suppose que vous utilisez un EDI *Arduino* à jour. Si vous débutez avec Arduino, consultez notre tutoriel sur



24 horloges élémentaires assemblées.

L'étape suivante consistait à créer le cadre. Je suis parti de deux planches d'érable. J'ai scié la première en trois minces morceaux...

... contrecollés ensuite pour faire la façade...

Arduino + FPGA = un exemple à suivre pour de nombreux projets

l'installation de l'EDI *Arduino* [14]. Si vous installez votre 1ère bibliothèque *Arduino*, consultez notre guide d'installation [15].
Le microcontrôleur que j'ai utilisé était le *RedBoard Turbo*, mais comme susmentionné, toute carte avec connecteur *Qwiic* peut probablement être utilisée. J'ai d'abord dû installer les bibliothèques pour la carte ainsi que le *RTC RV8803* de *Qwiic* et les boutons. Les boutons ci-dessous sont des liens vers leurs tutoriels d'installation respectifs.
Le code en lui-même n'est pas trop compliqué. La structure relève de mon expérience, je l'adopte pour de nombreux projets. En gros, je construis des couches d'abstraction jusqu'à ce que j'arrive à une couche facile à utiliser pour la gestion des chiffres et la réalisation des animations.
La première couche est, ça va de soi, le FPGA. Le FPGA nous donne une interface pour faire exécuter à un moteur un certain nombre de pas avec un délai fixe entre eux. J'ai utilisé la bibliothèque *Wire* intégrée à *Arduino* pour gérer le bus *I²C*. Cela m'a permis de réaliser une fonction simple qui envoie une seule animation - la voici :

```
void sendAnimation(uint8_t id, int16_t steps, uint16_t
delay_cycles) {
  int32_t p = currentPosition[id] + steps;
  while (p < 0) p += FULL_CIRCLE;
 currentPosition[id] = p % FULL_CIRCLE;
  Wire.write(id);
  Wire.write((uint8_t)(steps >> 8));
  Wire.write((uint8_t)(steps & 0xFF));
  Wire.write((uint8_t)(delay_cycles >> 8));
  Wire.write((uint8_t)(delay_cycles & 0xFF));
}
```

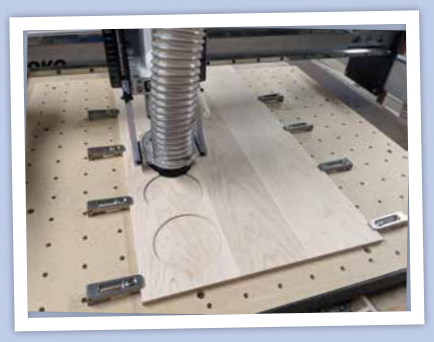
Deux particularités ici, je déclare une constante nommée `FULL_CIRCLE` pour le nombre de pas dans une rotation complète. Dans mon cas, c'est 4096. Cette constante est utilisée pour mettre à jour un tableau global des positions des moteurs. En employant toujours cette fonction pour envoyer les animations, la position des moteurs est connue.
Il n'est pas très naturel de penser un mouvement en termes de pas et de temporisation. Il est plus facile de le concevoir en termes de degrés et de durée. En d'autres termes, au lieu de penser «faire 2048 pas avec 760 cycles de tempo. entre chaque pas», il est bien plus naturel de penser «tourner de 180 ° en 4 s». J'ai donc écrit une fonction qui s'occupe de cette transposition :

```
void animate(uint8_t id, float deg, float duration) {
  float steps = deg * FULL_CIRCLE / 360.0f;
  if (steps > 32767 || steps < -32768) {
    animate(id, deg / 2, duration / 2);
    animate(id, deg / 2, duration / 2);
    return;
 }

  float cycles = constrain(duration * 390625 / abs(steps),
760, 65535);

  sendAnimation(id, (int16_t)steps, (uint16_t)cycles);
}
```

Elle commence par transformer les degrés en pas, à l'aide de la constante `FULL_CIRCLE`. Elle vérifie ensuite s'il y a trop de pas pour une seule commande d'animation et s'appelle récursivement avec la moitié de l'animation.
Les cycles de temporisation sont ensuite calculés. La valeur 390625 est le nombre de cycles en une seconde (100 000 000 / 256 = 390 625). Les cycles doivent être cantonnés à la plage de 760 à 65535. Le minimum de 760 a été trouvé empiriquement en testant la vitesse à laquelle les moteurs peuvent tourner sans sauter de pas. Cela correspond à 8 s par tour.

… que j'ai alors dressée puis j'ai découpé le pourtour avec ma CNC.

L'horloge est trop grande pour mon Shapeoko XXL, j'ai donc dû procéder en 2 fois pour chaque opération. Une fois une face terminée, j'ai légèrement fraisé l'autre face pour un dressage parfait.



La planche était alors prête pour le fraisage des logements des horloges.

La limite supérieure est très élevée et ne devrait jamais être atteinte. Il faudrait 687 s pour faire une rotation complète. Le projet actuel ne peut pas aller plus lentement que cela. Au besoin, il faudrait changer le prédimensionnement du FPGA ou la longueur du cycle de temporisation.
Il me fallait enfin une fonction permettant de dire au moteur de se déplacer jusqu'à une position donnée. La `CurrentPosition` permet à cette fonction de calculer la rotation minimale nécessaire pour l'atteindre. C'est utile pour afficher l'heure réelle, car je peux l'appeler quelles que soient les positions réelles des aiguilles, sans faire référence à leur position en cours.

```
void moveTo(uint8_t id, float pos, float duration) {
  float curDeg = (float)currentPosition[id] * 360.0f /
FULL_CIRCLE;
  float angle = pos - curDeg ;

  if (angle > 180)
    angle = angle - 360;
  if (angle < -180)
    angle = 360 + angle;

 animate(id, angle, duration);
}
```

Cela commence par le calcul de l'angle auquel l'aiguille se trouve à cet instant. Le déplacement angulaire s'obtient en soustrayant de l'angle souhaité l'angle calculé ci-avant. Mais cet angle pourrait ne pas donner le plus court des deux chemins ; les deux instructions if se chargent de le vérifier pour choisir le plus court. Par ex., si la différence entre les angles est de +270 °, il serait préférable de se déplacer de -90 °.

Pour afficher l'heure réelle, il me fallait une carte pour tous les chiffres. Il a suffi de noter les angles de chaque aiguille tels qu'ils forment les chiffres voulus . J'ai saisi le tout dans un tableau 2D (**listage 4**) utilisable pour retrouver n'importe quel chiffre. La RTC me donne l'heure dont il suffit d'afficher les chiffres :

```
void showTime() {
  uint8_t digits[4];
  digits[0] = rtc.getMinutes() % 10;
  digits[1] = rtc.getMinutes() / 10;
 digits[2] = rtc.getHours() % 10;
  digits[3] = rtc.getHours() / 10;
  for (uint8_t d = 0; d < 4; d++) {
    Wire.beginTransmission(0x50);
    for (uint8_t m = 0; m < 12; m++) {
      moveTo(m + 12 * d, digitAngles[digits[d]][m], 4.0f);
  }
    Wire.endTransmission();
  }
}
```

Il convient de mentionner que mon code suppose que les aiguilles commencent à la position 12h00. C'est le repère 0 ° pour toutes les positions.
Dans la fonction `loop()` *Arduino*, j'ai inclus du code qui permet de lire la RTC toutes les 100 ms et de mettre l'heure à jour. Au changement d'heure entière, je lance également une animation. J'en ai écrit trois simples dont l'une choisie au hasard s'exécute avant de fixer la nouvelle heure.
À l'intérieur de `loop()`, je vérifie également l'état des 4 boutons. Mon plan était d'utiliser l'interface FIFO du bouton *Qwiic* pour garder la trace de chaque pression, mais je suis tombé sur un os. J'ai fini par suivre moi-même l'état et la fonction `isPressed()` n'a servi qu'à vérifier leur état instantané.
Lorsqu'un bouton donné est enfoncé, je mets l'heure à jour. Les 4 boutons permettent de régler indépendamment minutes et

Une fois cette plaque de façade finie, elle s'est révélée un peu plus fine que prévu après le dressage. Le fond des logements ne faisait plus qu'1,5 mm d'épaisseur. Cela s'est révélé suffisamment solide.

J'ai ensuite construit un cadre et je l'ai collé…

… pour obtenir un solide coffret en bois.

heures. J'ai ajouté une fonction qui permet, avec un appui simultané sur les 2 boutons *Hour Up* et *Hour Down*, de faire indiquer 12h00 aux 24 horloges. C'est très utile s'il faut éteindre l'horloge ou reprogrammer l'Arduino, cela évite de réajuster chaque aiguille à la main.
Voilà qui résume bien le projet. Partez d'une interface simple et développez-la jusqu'à joindre l'utile à l'agréable.

Conclusion

Ce projet a demandé beaucoup plus de travail que je pensais au départ. La majeure partie du temps a été consacrée à la réalisation matérielle et au câblage. La conception et la mise au point un mouvement fonctionnel et fiable m'ont également pris un temps *certain*. La conception des parties FPGA et *Arduino* n'a souffert d'aucun contretemps notable.
L'utilisation du bus *Qwiic* de l'*Au* pour communiquer avec le microcontrôleur est sans doute une application des plus intéressantes. N'ayant jamais utilisé l'*I²C* sur un *Arduino* auparavant, j'ai été agréablement surpris par la facilité d'installation côté *Arduino*.
Si quelqu'un fabriquait une autre *ClockClock*, il y aurait plusieurs points à améliorer.

- L'horloge est assez bruyante. Pris individuellement, chaque moteur est assez silencieux, mais le fait de les coller ensemble sur un mince morceau de bois amplifie notablement le bruit. J'aurais dû prévoir un moyen d'amortir le bruit lors du montage des moteurs. Une colle à base de caoutchouc pourrait être une solution. De plus, le bois n'a que 1,5 mm d'épaisseur sur la majeure partie de la façade, laquelle fait donc caisse de résonnance. Je pourrais essayer de verser du caoutchouc liquide sur la face arrière de la planche pour essayer d'amortir le bruit.
- L'autre problème majeur tient au jeu du réducteur interne des moteurs pas à pas . En pratique, selon le sens de déplacement, l'aiguille peut ou non se trouver là où elle est censée être. À cause du jeu, l'écart semble faire quelques degrés lors des inversions de sens. Ce n'est pas la fin du monde, mais une fois qu'on s'en est aperçu, on ne voit rien d'autre. Je pourrais sans doute compenser cela par programmation, mais le jeu varie d'un moteur à l'autre et il faudrait un réglage individuel de la compensation. On pourrait aussi y remédier avec des moteurs sans engrenage.

J'espère que ce projet constitue une bonne démonstration des possibilités des FPGA et qu'il allumera (spark) en vous l'envie de prendre du plaisir (fun) à vous lancer dans votre propre *ClockClock* !

(200676-01 VF : Yves Georges)

Espace Elektor en ligne pour les liens internet :
www.elektormagazine.fr/esfe-en-clockclock

La conception des parties FPGA et Arduino de ce projet n'a souffert d'aucun contretemps notable. C'est la réalisation matérielle, la mise au point mécanique et le câblage qui sont chronophages.

Pour les aiguilles, j'ai choisi le padouk, un beau bois rouge qui brunit avec le temps. C'est aussi le bois des tiroirs de ma cuisine où l'horloge sera accrochée et j'ai pensé que ça ferait bien s'ils étaient assortis. J'ai usiné les aiguilles dans une planche de 3 mm fraisée jusqu'à une épaisseur de 2 mm.

Avec les aiguilles terminées…

… J'ai collé toutes les sous-horloges au boîtier de l'horloge. Quatre d'entre elles sont orientées curieusement pour dégager la place de l'alimentation. C'est une alimentation 12 V 6 A suffisante pour tous les moteurs.

Accesssoires

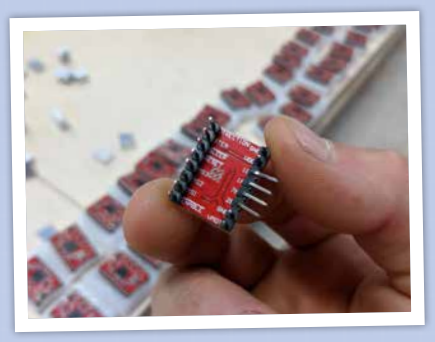
Si vous cherchez les accessoires mentionnés dans cet article, vous en trouverez certains chez SparkFun et Elektor.

- (2) Hook-Up Wire - Assortment (Solid Core, 22 AWG); PRT-11367
- (2) Qwiic Cable - 100 mm; PRT-14427
- (3) Qwiic Cable - 50 mm; PRT-14426
- SparkFun Real Time Clock Module - RV-8803 (Qwiic); BOB-16281
- (4) SparkFun Qwiic Button - Red LED; BOB-15932
- Alchitry Br Prototype Element Board; DEV-16524
- Alchitry Au FPGA Development Board (Xilinx Artix 7); DEV-16527
- SparkFun RedBoard Turbo - SAMD21 Development Board; DEV-14812

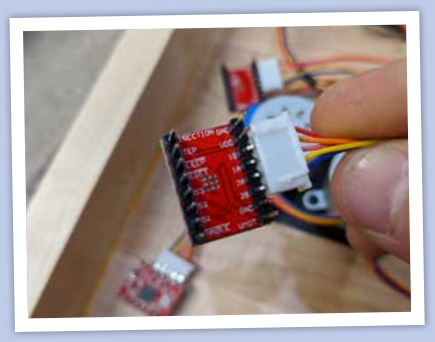
www.elektormagazine.fr/esfe-en-ClockClock1

LIENS

[1] ClockClock by Humans since 1982: https://clockclock.com
[2] Alchitry Au IO pins: https://www.elektormagazine.fr/esfe-en-ClockClock1
[3] Valve Gear Stepper Motor : https://www.amazon.com/gp/product/B01J3KV3B2
[4] StepStick Stepper Motor Diver Module with Heat Sink: https://www.amazon.com/gp/product/B01FFGAKK8
[5] UBEC Adjustable BEC UBEC 2-6S for Quadcopter RC Drone : https://www.amazon.com/gp/product/B07PLSYX9G
[6] Enclosed AC-DC Switching Power Supply : https://www.amazon.com/gp/product/B07Z55FCQQ
[7] Programming an FPGA: https://bit.ly/3nplgif
[8] How Does an FPGA Work? : https://bit.ly/3r42FdG
[9] Getting Fancy with PWM: https://bit.ly/2IUDmcZ
[10] CAD file (Fusion 360) : https://bit.ly/3mpXSjo
[11] ClockClock Alchitry fille (FPGA) : https://bit.ly/2K90HrW
[12] ClockClock Arduino code (zip) : https://bit.ly/3moeXdv
[13] 28BYJ-48 stepper motors: https://www.amazon.com/gp/product/B01J3KV3B2
[14] Arduino IDE installation: https://bit.ly/2Lz6OWM
[15] Installing an Arduino library: https://bit.ly/37owopV
[16] Learning FPGAs: Digital Design for Beginners with Mojo and Lucid HDL: https://amzn.to/38aexlN



J'ai utilisé des pilotes de moteur pas à pas génériques A4988 (www.amazon.com/gp/product/B01FFGAKK8). J'ai coudé les broches de commande des moteurs…

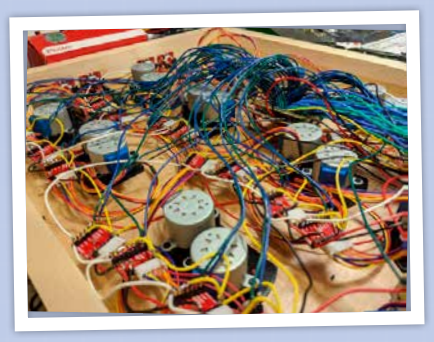


… et pu y brancher directement les connecteurs des moteurs après interversion de deux des fils.

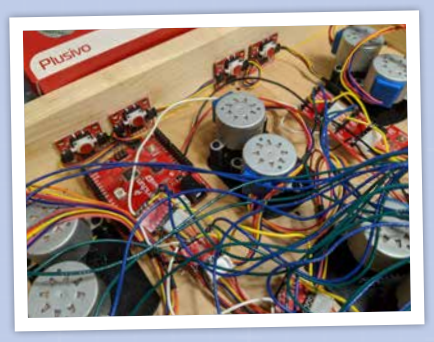
Il ne restait plus qu'à mettre sous tension. Sur cette photo, deux mouvements sont absents car il me manquait encore 4 moteurs.

Listage 1.

```
module animator (
    input clk,  // horloge
    input rst,  // reset
    signed input stepCount[16], // peut être négatif pour indiquer le sens
    input delayCycles[16],      // cycles entre chaque pas
    input newAnimation,         // drapeau pour nouvelle animation
    output busy,                // signale que l'animateur est occupé et n'accepte pas les animations
    output step,                // signal de pas pour le driver de moteur
    output direction            // signal de sens pour le driver de moteur
  ) {

  .clk(clk) {
    // Le driver exige que l'impulsion de „pas“ dure au moins 1 µs, avec 2 µs c'est bien
    pulse_extender stepExt(#MIN_PULSE_TIME(2000));

    dff dirCt[8];      // compteur pour la tempo après inversion de sens. Il faut 200 ns
    dff counter[16+8]; // compteur de tempo entre les pas. Le +8 est le prédiviseur

    .rst(rst) {
      fsm state = ;
      dff dir;         // sens de rotation du moteur mémorisé
      dff delayCt[16]; // valeur de tempo mémorisée
      dff steps[16];   // nombre de pas mémorisé (valeur absolue)
    }
  }

  always {
    busy = state.q != state.IDLE; // occupé quand il n'est pas inactif
    step = stepExt.out;           // la sortie step est l'impulsion étendue

    stepExt.in = 0;               // valeur par défaut du nouveau pas
    direction = dir.q;            // sortie du sens mémorisé

    case (state.q) {
      state.IDLE:
        if (newAnimation && stepCount != 0) {
                                  // si nouvelle animation avec des pas (sauter les animations avec 0 pas)
```



Chaque pilote devait être relié à l'alimentation 12 V pour les moteurs et 3,3 V pour la logique de commande. Sur chaque groupe de six constituant un chiffre, j'ai relié entre elles les broches de validation puis les ai reliées à l'*Alchitry Au*. Il fallait également raccorder à l'*Au* les signaux de sens et de pas de chaque pilote. Quel embrouillamini !



Il me fallait un moyen de régler l'heure, alors j'ai ajouté quatre boutons *Qwiic*. J'en avais un peu assez du câblage, et l'utilisation des connecteurs *Qwiic* a bien facilité les choses. Les deux du haut font *Heure +/-* et les deux du bas *Minute +/-*. Après les boutons, il y a une RTC (horloge en temps réel) *RV-8803* également connectée au bus *Qwiic*. La RTC étant alimentée par une batterie, je n'ai pas besoin de régler l'heure chaque fois que je reprogramme ou débranche la carte. Enfin, j'ai connecté l'*Alchitry Au*. Elle doit se trouver en bout de chaîne puisqu'elle n'a qu'un seul connecteur *Qwiic*. J'ai également retiré le fil d'alimentation du câble *Qwiic* pour que le régulateur 3,3 V de l'*Alchitry Au* n'entre pas en conflit avec celui de la

```
          state.d = state.DIR_WAIT;   // passer à l'état suivant
          dir.d = stepCount[stepCount.WIDTH-1];
                                      // le sens est le signe de l'entrée de pas (0 = positif, 1 = négatif)
          steps.d = stepCount[stepCount.WIDTH-1] ? -stepCount : stepCount;
                                      // mémorise la valeur absolue de stepCount
          delayCt.d = delayCycles;    // mémorise le nombre de cycles de temporisation
        }
      state.DIR_WAIT:
        dirCt.d = dirCt.q + 1;        // attendre la sortie du sens après son inversion
        if (&dirCt.q) {               // si tempo terminée
          state.d = state.STEP;       // passer à l'état rotation pas à pas
        }
      state.STEP:
        counter.d = counter.q + 1;    // incrémenter le compteur de tempo des pas
        if (counter.q[counter.WIDTH-1-:16] == delayCt.q) { // si le compteur a atteint la tempo finale
          counter.d = 0;              // réinit. le compteur
          stepExt.in = 1;             // envoyer une impulsion
          steps.d = steps.q - 1;      // décrémenter le nombre de pas restant
          if (steps.q == 1) {         // s'il n'en reste plus
            state.d = state.IDLE;     // retour à l'état d'inactivité
          }
        }
    }
  }
}
```

Listage 2.

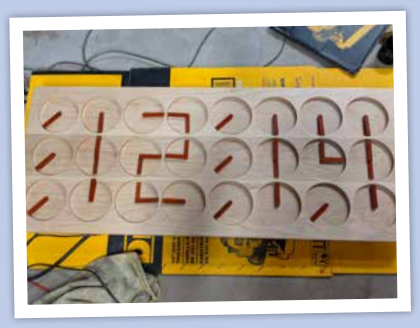
```
module enable_gate (
    input clk,  // horloge
    input rst,  // réinit.reset
    output new_animation[12], // sortie vers les animateurs
    input fifo_empty[12],     // entrée venant des fifos (animations en file d'attente)
    input animator_busy[12],  // entrée venant des animateurs
    output enable             // sortie vers les drivers des moteurs
  ) {
```

RedBoard Turbo. *** Le microcontrôleur est une carte *RedBoard Turbo*. Je l'ai utilisée parce qu'elle avait un connecteur *Qwiic* et que je l'avais sous la main. Tout microcontrôleur avec un connecteur *Qwiic* peut être utilisé. La puissance de calcul nécessaire est minime. *** L'*Alchitry Au* et la *RedBoard Turbo* nécessitent toutes deux du 5 V. J'ai utilisé un petit régulateur abaisseur (12 V en 5 V) commun en radiocommande et débitant jusqu'à 2 A. *** J'ai câblé les drivers pas à pas pour qu'ils utilisent le demi-pas. Au départ, ma configuration utilisait un micro-pas de 1/16e, mais les moteurs gémissaient s'ils n'étaient pas sur un demi-pas ou un pas entier. Une telle résolution était un peu superflue.

Le brochage des 102 fils qui vont dans le FPGA n'a pas vraiment d'importance. Il faut seulement noter comment vous les avez câblés. Le brochage réel est défini par le fichier de contraintes du projet FPGA.

Une fois le câblage fait, j'y ai mis un peu d'ordre…

```
  .clk(clk) {
    .rst(rst) {
      dff onCtr[22];   // compteur pour être sûr que les moteurs sont bien en marche (22 bits ~ 42 ms)
      dff offCtr[22];  // compteur de maintien en marche après la fin des animateurs (22 bits ~ 42 ms)
    }
  }

  sig running; // valeur utilisée pour savoir quand les moteurs doivent être en marche

  always {
    // exécuter si des animations sont en attente ou en cours d'exécution
    running = |(animator_busy | ~fifo_empty);

    // le drapeau enable est activé si l'option onCtr est activée ou n'est pas 0 (réinitialisé après un
dépasst de offCtr)
    enable = running || (onCtr.q != 0);

    // ne transmettre le drapeau new_animation que lorsque onCtr est plein
    new_animation = ~fifo_empty & 12x{&onCtr.q};

    if (running) {
      offCtr.d = 0;              // réinit. le compteur d'arrêt moteur
      if (!&onCtr.q) {           // si non plein
        onCtr.d = onCtr.q + 1;   // incrément onCtr
      }
    } else {                     // à l'arrêt
      if (!&offCtr.q) {          // si offCtr non plein
        offCtr.d = offCtr.q + 1; // incrémente offCtr
      } else {                   // si offCtr plein
        onCtr.d = 0;             // réinit. onCtr
      }
    }
  }
}
```



... et placé les aiguilles sur l'horloge.

Vous savez l'essentiel pour la construction physique. Le câblage est fastidieux, mais pas compliqué.

Listage 3.

```
module au_top (
    input clk,          // horloge 100 MHz
    input rst_n,        // bouton de réinitialisation (actif à 0)
    output led [8],     // 8 LED contrôlables par l'utilisateur
    input usb_rx,       // entrée USB->Série
    output usb_tx,      // sortie USB->Série
    output step[48],    // sortie des pas vers les moteurs
    output dir[48],     // sortie du sens de rotation vers les moteurs
    output enable[4],   // sortie activation vers les moteurs (une par digit)
    inout sda,          // SDA Qwiic
    input scl           // SCL Qwiic
  ) {

  sig rst;              // signal de réinit.

  .clk(clk) {
    // Le conditionneur de réinit. est utilisé pour synchroniser le signal de réinit. vers
    // l'horloge FPGA.

Cela garantit que le FPGA entier sort de la réinit. en même temps.
    reset_conditioner reset_cond;

    dff ani_id[6];            // octet ID mémorisé pour le moteur
    signed dff ani_steps[16]; // nombre de pas mémorisé
    dff ani_delay[16];        // compte de temporisation mémoriséla tempo

    dff byteCt;               // drapeau d'octet pour les nombres 16 bits

    .rst(rst) {
      i2c_peripheral qwiic (.sda(sda), .scl(scl)); // module périphérique i2c pour l'interface qwiic

      dff ledReg[8]; // registre de mémorisation des valeurs des LED (utile pour les tests qwiic)

      fsm state = ;

      animator animators[48]; // il faut 48 animateurs (un par moteur)

      // il faut une fifo par animateur, 32 bits de large (16 bits nbre de pas + 16 bits tempo)
      // la profondeur de 128 est excessive et 16 serait probablement suffisant pour
      // l'usage actuel.
      fifo ani_fifos[48] (#SIZE(32), #DEPTH(128));

      enable_gate gates[4]; // modules de contrôle des signaux d'activation (un par chiffre)
    }
  }

  var i;

  always {
    reset_cond.in = ~rst_n; // entre le signal de réinitialisation brut inversé
    rst = reset_cond.out;   // reset conditionné

    led = ledReg.q;         // envoie ledReg sur les leds

    usb_tx = usb_rx;        // renvoie l'écho des données série reçues

    // le ~ inverse le sens du moteur pour que les pas positifs se fassent dans le sens horaire
    // 4hA = 1010, donc dans 48hAAAAAAAAAAAAAA, 1 bit sur 2 est à 1, de sorte que les
    // moteurs feront tourner les aiguilles des heures et des minutes dans le même sens
    dir = animators.direction ^ ~48hAAAAAAAAAAAA;
    step = animators.step;
```

```
enable = ~gates.enable; // l'activation des contrôleurs est active à 0, donc inversion des bits

qwiic.tx_data = 8bx; // ce projet est „en écriture seule“ et n'envoie jamais de données au microcontrôleur
qwiic.tx_enable = 0; // ne jamais envoyer de données

// groupes combinés de 12 moteurs pour les 4 portes de validation
for (i = 0; i < 4; i++) {
  gates.fifo_empty[i] = ani_fifos.empty[i*12+:12];
  gates.animator_busy[i] = animators.busy[i*12+:12];
  animators.newAnimation[i*12+:12] = gates.new_animation[i];
  // ne supprime une valeur de la fifo que si la porte passe le drapeau new_animation
  // et que l'animateur n'est pas occupé
  ani_fifos.rget[i*12+:12] = gates.new_animation[i] & ~animators.busy[i*12+:12];
}

// pour chaque moteur, diviser la sortie fifo vers les signaux de l'animateur
for (i = 0; i < 48; i++) {
  animators.stepCount[i] = ani_fifos.dout[i][15:0];
  animators.delayCycles[i] = ani_fifos.dout[i][31:16];
}

// pas de nouvelles animations par défaut
ani_fifos.wput = 48b0;

// toujours entrer la tempo. et les pas mémorisés
// Concatène les 2 valeurs de 16 bits en 32 bits, et les range dans un tableau 1x32,
// et enfin le duplique 48 fois pour former un tableau 48x32
// cela ne fait qu'alimenter chacune des 48 fifos avec les mêmes 32 bits à
ani_fifos.din = 48x{}};

case (state.q) {
  state.IDLE:
    byteCt.d = 0;
    if (qwiic.rx_valid) {          // nouvelles données
      case (qwiic.rx_data) {       // cas selon la valeur de l'adresse I2C
        8hFF: state.d = state.LED; // faire „addresse“ FF pour les LEDs pour le test
        default:
          ani_id.d = qwiic.rx_data[5:0];  // par défaut „adresse“ comme identifiant du moteur
          state.d = state.ANIMATION_STEPS;
      }
    }
  state.LED:
    if (qwiic.rx_valid) {        // si nouvelles données
      state.d = state.IDLE;      // retour à l'état d'inactivité
      ledReg.d = qwiic.rx_data;  // afficher la valeur sur les LED
    }
  state.ANIMATION_STEPS:
    if (qwiic.rx_valid) {                // si nouvelles données
      ani_steps.d = c;  // mémorise l'octet et décale l'ancien octet
      byteCt.d = ~byteCt.q;              // inverse le compteur d'octet
      if (byteCt.q == 1) {               // si deuxième octet
        state.d = state.ANIMATION_DELAY;  // passer à l'état de capture de la tempo
      }
    }
  state.ANIMATION_DELAY:
    if (qwiic.rx_valid) {                // si nouvelles données
      ani_delay.d = c;  // sauvegarde l'octet et décale l'ancien octet
      byteCt.d = ~byteCt.q;              // inverse le compteur d'octet
      if (byteCt.q == 1) {               // si deuxième octet
        state.d = state.ANIMATION_PUT;    // passer à l'état remplissage fifo
      }
    }
  state.ANIMATION_PUT:
    state.d = state.IDLE;                // retour à l'état d'inactivité
```

```
        ani_fifos.wput[ani_id.q] = 1;         // mettre la nouvelle animation dans la fifo correcte
    }

    if (qwiic.stop) {           // si une condition d'arrêt I2C est détectée
      state.d = state.IDLE;     // retour à l'état d'inactivité
    }
  }
}
```

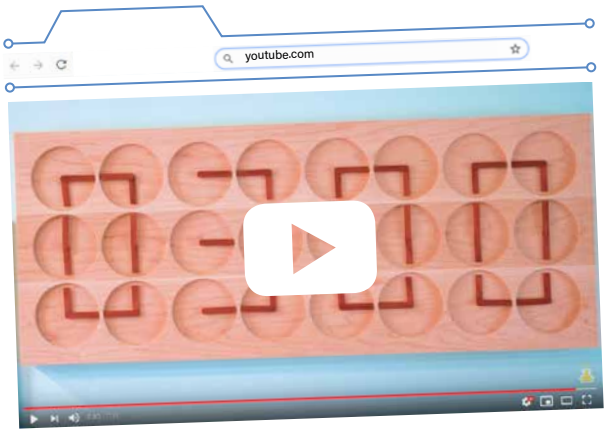
Listage 4.

```
const float digitAngles[10][12] = {{270, 180, 0, 180, 270, 0, 90, 180, 0, 180, 90, 0}, // 0
  {180, 180, 0, 180, 0, 0, 225, 225, 225, 225, 225, 225}, // 1
  {270, 180, 270, 0, 270, 270, 90, 90, 90, 180, 90, 0}, // 2
  {270, 180, 0, 180, 270, 0, 90, 90, 90, 90, 90, 90}, // 3
  {180, 180, 0, 180, 0, 0, 180, 180, 90, 0, 225, 225}, // 4
  {270, 270, 270, 180, 270, 0, 90, 180, 90, 0, 90, 90}, // 5
  {270, 270, 270, 180, 270, 0, 90, 180, 0, 180, 90, 0}, // 6
  {270, 180, 0, 180, 0, 0, 90, 90, 225, 225, 225, 225}, // 7
  {270, 180, 270, 0, 0, 270, 90, 180, 90, 0, 0, 90}, // 8
  {270, 180, 0, 180, 0, 0, 90, 180, 90, 0, 225, 225} // 9
};
```

Vidéo de démonstration

Vous pouvez voir fonctionner le proto de Justin sur YouTube :

https://youtu.be/rNJlQ4Fa9mQ



Un coup ça marche, un coup ça marche pas

Si vous avez besoin d'aide ou d'informations, veuillez parcourir les forums d'Alchitry sur https://forum.alchitry.com. Vous y trouverez probablement la réponse à vos questions ou les personnes susceptibles d'y répondre.

Pour approfondir

1. Fichiers de production :

- CAD File (Fusion 360) [10]
- Alchitry (FPGA) [11]
- Arduino Code (ZIP) [12]

2. Le site web d'Alchitry (https://alchitry.com) propose d'autres ressources intéressantes, notamment des tutoriels et projets, un forum, le schéma Alchitry Au (PDF), le schéma Alchitry Cu (PDF) et le guide d'utilisation de Xilinx Artix 7.

3. Pour approfondir vos connaissances des FPGA et de Lucid, consultez la page *Learning FPGAs: Digital Design for Beginners with Mojo and Lucid HDL* par Justin Rajewski [16]. C'est une excellente ressource pour comprendre et finalement concevoir vos propres applications FPGA.

Sous le capot : Inventor's Kit de SparkFun

Luc Lemmens (Elektor)

Oh non, encore un kit de démarrage Arduino… ? Et si ce SIK, c'est-à-dire l'*Inventor's Kit* de Sparkfun était exactement ce dont vous avez besoin pour vous initier à l'électronique et à la programmation des microcontrôleurs ?

Lâché dans la Forêt des Microcontrôleurs, le débutant désireux de s'initier à l'électronique et à la programmation risque fort de s'égarer en cherchant du matériel adapté. Par quelle carte et quel processeur débuter ? L'Arduino Uno reste l'une des options évidentes et abordables. Sur l'internet on trouve des tonnes de logiciels et de tutoriels Arduino.

Outre la carte avec le processeur, il vous faudra quelques composants supplémentaires (interrupteurs, LED, capteurs, etc.) pour que la carte puisse interagir avec le monde. Et si vous allez du côté de la robotique, il faudra choisir des moteurs et d'autres pièces électromécaniques, ce qui ne facilite pas les choses. Pour prendre un raccourci, on peut opter pour le kit-complet-qui-contient-tout-ce-qui-manque mais là encore l'offre de kits de démarrage Arduino est vaste. Lequel offre le meilleur rapport qualité-prix ? Voyons ce qu'à offrir ce SIK, et je vous dirai pourquoi je le recommande.

C'est quoi ce SIK ?

Comme le décrit SparkFun sur son site [1] : „*L'Inventor's Kit* de SparkFun est un excellent moyen de se lancer dans la programmation et l'interaction matérielle avec le langage de programmation Arduino". Cette affirmation est correcte à tous égards : tout ce dont vous avez besoin (à l'exception du logiciel, que vous devez télécharger vous-même [2], et de quatre piles), y compris un tournevis, se trouve dans ce kit. Le SIK est emballé dans une solide valise en plastique, de sorte que son contenu peut être rangé et transporté en toute sécurité.

Keskia dans ce SIK ?

Le contenu de la boîte (**fig.** 1) est décrit sur le site de SparkFun et dans le manuel **imprimé** joint (*OUI, du vrai papier !*). La liste du matériel (BoM) n'est d'ailleurs pas tout à fait complète, mais c'est une critique mineure. Avant de commencer à assembler un kit, quel qu'il soit, il est bon d'en vérifier le contenu. Une liste exacte est également utile lorsque vous souhaitez commander des pièces de rechange. À part ça, je n'ai que des éloges sur la documentation fournie, j'y reviendrai.

Le SIK comprend la carte RedBoard de SparkFun comme carte à processeur (**fig. 2**), qui – sauf quelques détails mineurs – est compatible avec le célèbre et classique Arduino Uno. La différence la plus frappante est le *connecteur qwiic* de SparkFun, une connexion à 4 fils qui fournit au bus I²C une tension d'alimentation de 3,3 V pour le matériel externe. SparkFun offre une série d'extensions équipées de cette connexion, mais celles-ci ne sont pour l'instant ni utilisées ni discutées dans les projets SIK. Cela laisse la porte ouverte à des possibilités inédites pour le jour où tous les tutoriels pour le SIK auront été réalisés.

La carte processeur RedBoard est entièrement compatible avec l'Arduino Uno et utilisée de la même manière que l'EDI Arduino, mais vous devez installer un pilote USB différent pour qu'elle interagisse avec votre ordinateur. Ne vous inquiétez pas, l'installation complète de l'environnement de programmation Arduino pour Windows, MAC OS et Linux est également expliquée dans le manuel.

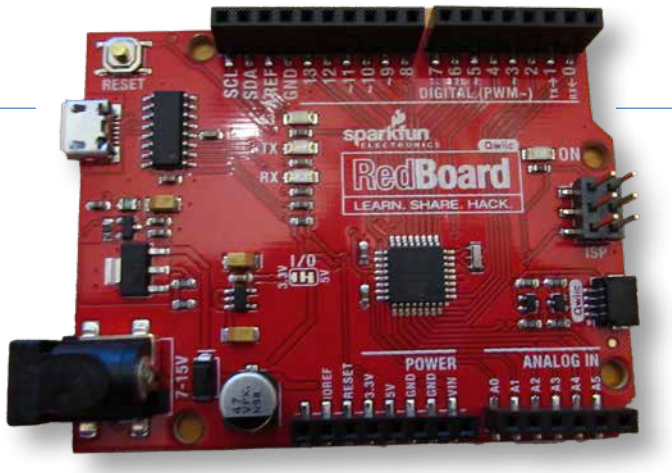


Figure 2. Le RedBoard est un sosie de l'Arduino Uno.

Keski faut savoir ?

Aucun outil de soudure ni aucune compétence particulière n'est nécessaire pour réaliser les projets présentés dans le manuel – tous les composants supplémentaires peuvent être connectés via la plaque d'assemblage (**fig. 3**) fournie dans la boîte et il suffit de les brancher et d'utiliser des fils de connexion, de sorte que même le bricoleur le plus inexpérimenté ne se brûlera pas les doigts ici. Comme promis, vous pouvez commencer avec les projets du SIK sans expérience préalable en électronique ou en programmation.

Manuel de SIK

Le kit de l'inventeur de SparkFun décrit cinq „*projets*" (lumière, son, mouvement, affichage et robot) et chaque projet contient plusieurs *circuits*, 16 en tout. Des termes comme *thèmes* et *didacticiels* m'auraient paru mieux appropriés, mais ce n'est qu'une question de goût. Ça commence évidemment par le clignotement d'une LED et ça se termine par un robot en mouvement qui évite les obstacles à l'aide d'un capteur à ultrasons.

Les instructions pour chaque circuit commencent par une explication des nouveaux composants électroniques ou électromécaniques et des concepts introduits. Elles sont suivies du *guide de branchement* qui explique comment chaque circuit est construit à l'aide de la platine et des fils de connexion, puis des instructions pour télécharger le croquis sur la RedBoard. Ensuite, il y a une section expliquant comment fonctionne le croquis Arduino pour ce circuit et ce que signifient et font les instructions les plus importantes du programme. Elle est suivie de suggestions pour des expériences supplémentaires avec le logiciel (*Coding Challenges*), et la description du circuit se termine par des conseils de dépannage : que vérifier et que faire si le circuit ne fonctionne pas comme décrit. Cette structure est répétée pour chaque circuit du manuel, les instructions deviennent progressivement plus concises, bien sûr, tandis que la complexité des circuits et des schémas augmente.

La qualité du manuel réside dans cette structure cohérente de tous les *circuits*, dans la concision et la simplicité des explications, sans omettre d'informations importantes. C'est donc facile à suivre, même pour les lecteurs non anglophones. La mise en page est claire, et l'utilisation de couleurs permet de distinguer les sections, et les illustrations claires rendent agréable le travail avec le SIK (**fig. 3**). Un bon dessin vaut mieux que des pages d'explication. Un livret relativement mince peut donc contenir une tonne d'informations utiles (**fig. 4**). Quel plaisir, à l'ère de l'information en ligne, de profiter d'une documentation imprimée aussi solide ! Attention, certains *errata* qui n'ont peut-être pas été corrigés dans le manuel imprimé que vous recevez avec l'ISI, mais ils peuvent être trouvés sur l'internet [3] – le manuel affiché là est à jour. Les circuits et les croquis présentés dans le SIK pour se familiariser avec l'électronique et la programmation dans l'environnement de programmation Arduino ne sont pas particulièrement complexes. C'est la combinaison du matériel complet (électronique et mécanique), du logiciel et, peut-être plus important encore, d'une documentation complète et claire qui fait du SIK un outil chaudement recommandé. Elle offre un excellent matériel didactique, tant à l'université que pour l'autoapprentissage. La proposition est très divertissante même pour les électroniciens expérimentés, surtout s'ils cherchent des idées et des réalisations à partager avec un débutant désireux d'apprendre !

(200648-03 VF : Rémy Fasollado)

Figure 1. Beaucoup de choses amusantes dans la boîte du SIK.

Accessoires

Les accessoires mentionnés dans cet article sont disponibles chez SparkFun et Elektor.
www.elektormagazine.com/esfe_sik1

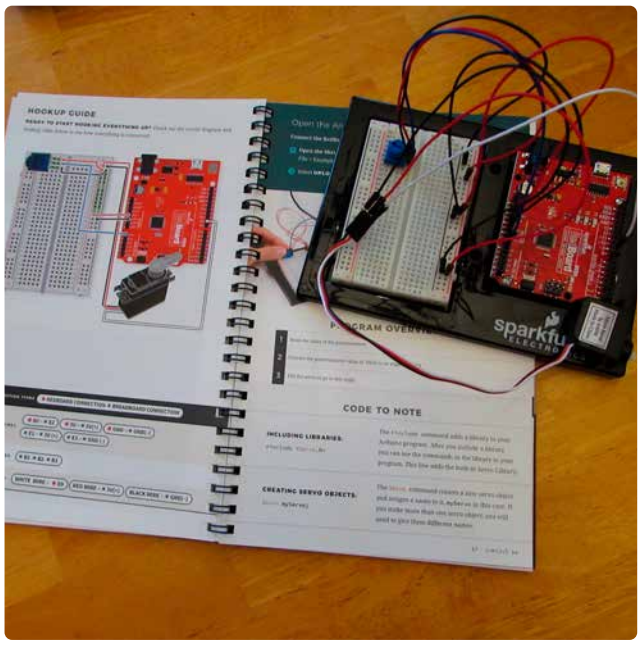


Figure 3. Circuit d'asservissement construit à l'aide de la plaque d'expérimentation du SIK.

LIENS

[1] SIK sur le site de SparkFun :
https://www.sparkfun.com/products/15267
[2] Installation de SIK :
https://bit.ly/2XwWYrh
[3] Manuel de SIK :
https://www.sparkfun.com/SIKerrata

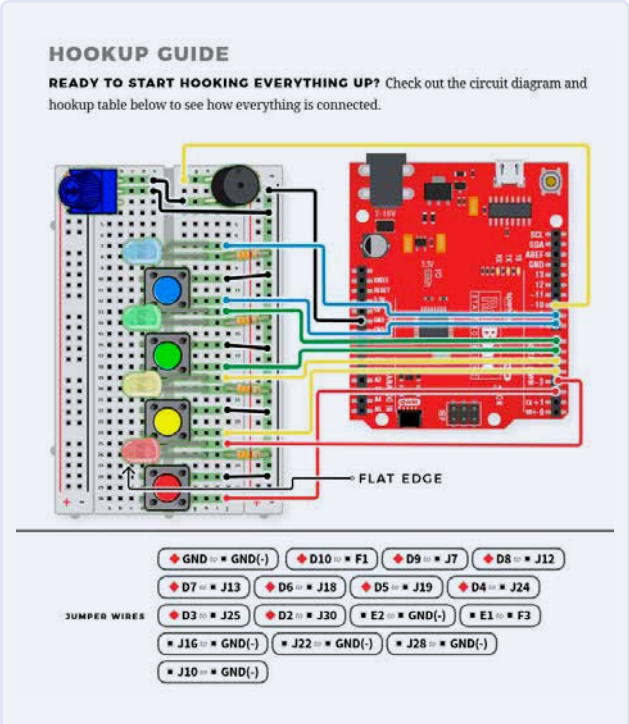


Figure 4. Instructions illustrées pour construire un circuit.

Point de vue de Glenn Samala **nouveaux produits, nouvelles entreprises**

C.J. Abate (Elektor)

En 2020, année difficile entre toutes pour les entreprises, SparkFun a mis en circulation plus de 70 produits originaux. Le PDG Glenn Samala partage ses réflexions sur le processus de développement des produits et sur ce que la communauté des bricoleurs en électronique peut attendre de l'entreprise.

Abate : Vous avez passé neuf ans chez Arrow Electronics avant de rejoindre SparkFun. Qu'est-ce qui vous a conduit à SparkFun ?

Samala : Comme beaucoup de PDG, j'ai été contacté par un recruteur, mais ce qui m'a convaincu, c'est l'histoire de SparkFun et de son fondateur. Un étudiant qui a lancé cette entreprise technique en 2003, dans sa piaule à l'université, sans soutien financier. Un groupe d'employés jeunes et talentueux qui, contre toute attente, pendant la Grande Récession américaine, ont construit une entreprise qui conjugue idées et technique dans ce Nouveau Monde numérique et virtuel d'une manière inédite et percutante. Pas question pour moi de laisser passer l'offre de diriger cette entreprise.

J'ai entrevu la perspective d'un parcours unique pour SparkFun à partir du talent entrepreneurial évident de Nathan Seidle, en le combinant avec mon parcours et mon expérience si différents des siens.

Nathan est un entrepreneur en série qui a réussi à créer une entreprise sans aucun tour de financement. J'ai commencé ma carrière dans une grande entreprise classée *Fortune 100* qui en intégrant des entreprises acquises est passée de 800 millions lorsque j'ai commencé à 9 milliards lorsque je suis parti. Le parcours du fondateur de SparkFun et celui de son nouveau PDG ne pouvaient pas être plus dissemblables que les nôtres, et c'est ce qui m'a captivé.

Toute transition de fondateur à PDG est au mieux délicate, surtout pour le premier PDG qui succède au fondateur. C'est l'un des défis les plus ardus de ma carrière, mais l'un des plus gratifiants.

Abate : Quel a été l'impact de la COVID-19 sur SparkFun ? Pouvez-vous nous parler d'un ou deux ajustements majeurs que vous avez dû faire ?
Samala : Comme pour tant d'autres entreprises dans le monde, la COVID-19 a eu un impact sur nous, depuis la gestion des défis de la chaîne d'approvisionnement mondiale jusqu'à la façon dont nous interagissons et facilitons l'interaction au quotidien. Objectif principal : protéger la santé de nos employés. La plupart, sinon la totalité, de nos ajustements majeurs sont donc axés sur la protection de nos employés. Ce sont les faits et la science qui nous aident à prendre les décisions difficiles.
Le 13 mars 2020, le gouvernement américain a déclaré l'état d'urgence, et le 20 mars, nous avons éloigné 70 % de notre personnel de son lieu de travail habituel. Les seuls employés actuellement admises au siège de l'entreprise sont les équipes de logistique et de fabrication. Auparavant, toutes les équipes fonctionnelles étaient regroupées au siège social à Boulder : informatique, services produits, ressources humaines, ingénierie, marketing, chaîne d'approvisionnement, ventes, développement commercial, finances, administration des opérations, fabrication et logistique, ce qui nous avait permis de forger une culture de travail fortement collaborative. Cette décision d'éloigner sept employés sur dix de leur lieu de travail a donc été difficile mais nécessaire pour protéger tout le monde. Pour l'équipe sur place, nous avons instauré un calendrier échelonné afin de ménager plus d'espace et de permettre la distanciation sociale. Collations, nourriture et boissons sont fournies sur place afin de réduire le trafic à l'entrée et à la sortie du bâtiment. Toujours sur place, des contrôles de température sont effectués quotidiennement et des masques (indispensables), des gants, des lingettes désinfectantes et du désinfectant pour les mains sont mis à la disposition de tous.
Je sais que tout le monde aspire à retourner au bureau, ne serait-ce que pour les habituelles discussions informelles de hasard dans les couloirs dont la valeur est réelle au sein de SparkFun. Nous passons tous à côté d'occasions potentielles, et nous y reviendrons le moment venu.

Le local, c'est là où vous vivez ; les grandes idées sont mondiales, le talent aussi.

Glenn Samala

Abate : Quels ont été vos produits les plus populaires l'année dernière ? Des surprises ?
Samala : Une grande partie de nos progrès en 2020 peut être attribuée à la demande d'outils de prototypage rapide. Tout le monde essaie d'expérimenter les nouvelles techniques et espère atteindre plus rapidement le stade de la validation du concept. C'est là que nous intervenons.
Toute notre gamme de produits Qwiic continue de foisonner, de nos cartes Qwiic si variées aux capteurs Qwiic, avec tout ce qui se trouve entre (accessoires Qwiic, HAT, etc.). Les ressources du prototypage rapide offertes par l'écosystème Qwiic, ainsi que nombre de nos autres produits et services, ne cessent de croître.
SparkFun suscite aussi l'attention avec la sortie d'Artemis, d'ALC, de MicroMod et avec les initiatives de notre Service. L'équipe met en place de nouvelles activités passionnantes, tout en produisant de nouveaux accessoires puissants et originaux signés SparkFun, notamment pour des produits partenaires comme Alchitry, NVIDIA, micro:bit et Raspberry Pi. Pour vous donner une idée de l'ampleur de cette vitalité, en 2020, en plus de toutes ses initiatives de croissance SparkFun a lancé pas moins de 70 produits originaux.
Mes surprises agréables, ce sont les anciens produits originaux de SparkFun, ceux qui ont plus de 7 ans et pour lesquels la demande reste forte. Par exemple, le convertisseur de niveau logique SparkFun sorti en 2013 dont les volumes continuent de faire des jaloux. C'est une preuve éclatante de la valeur de composants de base bien documentés.

Abate : Où voyez-vous des possibilités de croissance dans les six prochains mois ?
Samala : Je pense que cela dépend de la clientèle envisagée. Les clients servis par SparkFun sont très variés, de l'enseignement supérieur (collèges et universités), au hacker dans son garage, en passant par l'entrepreneur en série, les *start*-ups, les sociétés de R&D, sans oublier les grandes entreprises.
Pour ce qui concerne les produits, je pense que tout ce qui est lié aux techniques émergentes, comme l'apprentissage machine à faible consommation et le RISC-V, offre un potentiel d'accélération réelle dans les six prochains mois grosso modo. Je crois aussi que les techniques en voie de maturation, généralement réservées aux entreprises établies, se frayeront un chemin jusqu'à l'ingénieur consommateur - pensez aux techniques géospatiales et au suivi des ressources mondiales.
Comme l'incertitude et l'imprévisibilité de 2020 se prolongeront en 2021, qui sait vraiment où se situera le potentiel de croissance dans six mois ? Ce que je sais, c'est que la COVID n'a pas ralenti le rythme de l'innovation en 2020. Tant que SparkFun s'attachera à rendre intéressantes les nouvelles techniques, et accessibles au grand public, notre croissance se poursuivra.

Abate : SparkFun met l'accent sur la création de contenu : articles, billets de blog et vidéos. Pouvez-vous nous parler de votre approche du marketing de contenu ?
Samala : L'existence de SparkFun repose principalement sur le contenu numérique. Il y a 18 ans, Nathan avait beaucoup de mal à

SparkFun est situé à Niwot, Colorado, USA.

Le plaisir fait partie de la culture de l'entreprise.

trouver quelqu'un qui lui vende une simple carte de développement, et c'est une des deux grandes raisons de l'existence de SparkFun. L'autre besoin criant était l'absence d'une documentation qui dise à l'utilisateur potentiel quoi faire du produit. Nathan s'est donc mis à utiliser le produit, il a appris à le connaître, il l'a documenté et l'a partagé avec le monde entier. Le caractère révolutionnaire de cette démarche de création de contenu peut nous échapper aujourd'hui, mais elle l'était il y a 18 ans. Et c'est elle qui a créé une valeur énorme pour les produits qui en ont bénéficié.
Dans toute entreprise, l'échelle des valeurs est mobile, cela vaut aussi pour le contenu numérique. On s'attend à ce que ce qui avait de la valeur il y a dix ans ou plus ait de la valeur aujourd'hui. C'est pourquoi nous poursuivons le développement de notre contenu numérique afin d'enrichir l'expérience de nos clients. C'est ce qui explique la profusion de contenu autour de nos produits, mais n'oublions pas que cela ne s'est pas fait en un éclair. C'est le résultat de 18 ans d'évolution.
Notre approche du marketing de contenu consiste à créer une plate-forme numérique permettant à tout un chacun de raconter une histoire : ce que fait le produit, ce à quoi vous pouvez l'utiliser et, enfin, quel projet vous avez créé et pourquoi. Le produit est intéressant, certes, mais aussi ce que les gens en font.

Abate : Je suppose que vos ingénieurs et les membres de votre communauté présentent des dizaines de bonnes idées de produits chaque année. Comment votre équipe choisit-elle les idées à concrétiser ?
Samala : Nous recevons chaque année et de toutes parts des dizaines de bonnes idées de produits. Cette profusion est parfois difficile à gérer. Pour les idées qui nous viennent de l'extérieur, c'est Kirk Benell, notre directeur technique, qui décide. Il évalue le potentiel d'intégration d'une idée de produit externe dans notre feuille de route globale et détermine si le produit peut nous ouvrir certains marchés sur lesquels nous voulons opérer.
De nombreuses idées de produits proviennent de l'intérieur, notamment de notre équipe SparkX. SparkX est notre version de Skunkworks. C'est une équipe dirigée par Nathan, notre fondateur, qui jouit d'un degré élevé d'autonomie et de liberté. Indépendamment des considérations liées au marché ou à la feuille de route, si un nouveau produit ou une nouvelle technique leur plaît, ils vont de l'avant et lancent le produit sous notre marque SparkX. Cela nous permet de sonder rapidement le marché et nous aide à déterminer, en fonction de l'intérêt manifesté par la communauté, si ce produit doit passer du statut de SparkX „Black" à celui de SparkFun „Red".
Comme la plupart des processus de production, ce n'est pas toujours aussi simple. Il arrive que Kirk, ne voyant pas l'intérêt d'une idée venue de l'extérieur, prenne l'avis de Nathan. Il arrive aussi que Nathan reçoive une idée qui ne l'intéresse pas vraiment, mais dont il pense qu'elle pourrait s'inscrire dans notre stratégie de feuille de route pour les produits. Le niveau élevé de collaboration entre Nathan et Kirk est élevé, et si nécessaire ils m'impliquent aussi.

Abate : SparkFun est bien connu pour son engagement sur le marché des passionnés et des électroniciens amateurs. Avec la mise au point et le lancement d'un produit comme Artemis et d'un service comme À La Carte, regardez-vous aussi au-delà de ce marché ?
Samala : Je ne dirais pas que je regarde *au-delà* du marché des passionnés, mais que SparkFun se développe avec lui. Nous avons une vaste base de clients enthou-

siastes fidèles depuis 18 ans, et au fil du temps beaucoup d'entre eux ont créé leur propre entreprise. Beaucoup ont également commencé à travailler dans des entreprises de R&D où ils pratiquent le prototypage rapide. Que vous soyez un individu ou une entreprise, le fait est que tout au long de la vie le processus d'apprentissage et de développement ne s'arrête pas. Nous sommes heureux de croître et d'évoluer aux côtés de notre communauté. Que l'on débute dans l'électronique ou que l'on ait progressé au point de vouloir commercialiser sa propre idée de produit, nous voulons être cette plate-forme pour nos clients (anciens et nouveaux) en leur faisant gagner du temps et des ressources, où qu'ils se trouvent dans leur parcours avec l'électronique.

Abate : Vous avez de nombreux clients en dehors des États-Unis ? Qu'en est-il de l'Europe ?
Samala : Oui, en 2020, environ 30 % de notre activité actuelle est internationale, ce qui représente une légère baisse dans le segment international B2B de notre activité en raison de la COVID.

Abate : Pourquoi SparkFun a-t-il lancé „À la carte" ?
Samala : SparkFun a toujours eu pour but de fournir les éléments techniques de base pour aider électroniciens curieux à apprendre dans ce monde merveilleux de l'électronique ou même à concevoir eux-mêmes une idée de produit. Un outil comme À La Carte (ALC) va plus loin en permettant à chacun d'intégrer des blocs standard dans sa propre carte personnalisée.
À La Carte permet de gagner du temps et d'économiser des ressources pendant la phase de prototype. De nombreuses entreprises ne disposent pas du capital, des ressources ou de l'expertise matérielle nécessaires pour accélérer le prototypage, et cette offre de service doit les y aider.
Dans l'enseignement supérieur, nous savons que les kits personnalisés sont précieux pour les professeurs – nous proposons déjà des kits personnalisés adaptés à leurs objectifs de cours. L'ALC va plus loin et donne aux enseignants la possibilité de créer des cartes personnalisées pour leurs programmes d'études ou leurs projets de base.

En somme, ALC offre donc une approche personnalisée, que ce soit dans l'éducation et la formation ou dans le secteur privé.

Les bonnes idées de produits (matériels et logiciels) ne manquent pas, notre mission est d'en faire des succès.

Glenn Samala

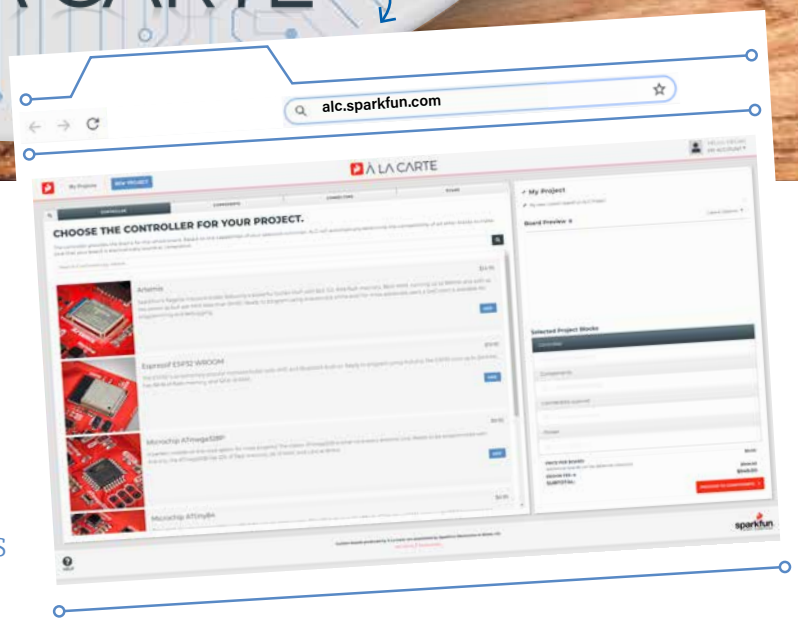
Abate : Pour votre équipe, l'offre À La Carte pose des défis commerciaux uniques. Pouvez-vous partager vos expériences ? Et en quoi SparkFun est-il différent d'un autre fournisseur ?
Samala : Pas de doute. Chaque fois que dans une discussion d'affaires il est question de „sur mesure", les choses se compliquent. Pour l'offre ALC, ce défi incombe à nos équipes opérationnelles, production/fabrication et chaîne d'approvisionnement, et ce sont tout simplement des questions d'échelle.
Pour ceux qui ne le savent pas, nous fabriquons en fait des cartes à notre siège dans le Colorado. Les processus pour les cartes ALC personnalisées sont similaires à celui de nos cartes originales SparkFun. Ainsi, sachant que le processus de fabrication des cartes ALC est notre seconde nature, à mesure que cette offre deviendra plus populaire, notre équipe aura à tenir compte des exigences, du rythme et de l'échelle spécifiques de la fabrication de cartes personnalisées.
Nous ne sommes comparables à aucun prestataire, car nous ne sommes pas un prestataire : notre modèle n'est pas celui de gros volumes pour des produits matures. Je laisse cela aux Flextronics, FoxConns et Jabils du monde entier - ils font ça très bien et SparkFun ne leur fera pas de concurrence. L'offre À La Carte consiste à soutenir les nouvelles idées et à aider les clients à les mettre sur le marché le plus rapidement possible. Avec des initiatives comme ALC nous aidons les autres à se développer dans ce domaine de l'électronique, en particulier dans ce créneau de l'expérimentation et du prototypage rapide. Il y a tant de bonnes idées de produits (matériels et logiciels) et notre intention est d'en faire des succès. Quelle qu'en soit la complexité, notre équipe est attachée à cette vision.

Abate : Parlez-nous un peu de Boulder, Colorado, et de ses environs. Y a-t-il une scène technique là-bas ? Est-ce un biotope pour trouver des talents d'ingénieurs ?
Samala : Le milieu des spécialités technique est riche à Boulder, dans le Colorado, ainsi que dans les environs de Denver. Plus important encore est le degré élevé de coopération, de soutien et d'amitié entre les entreprises ici, une véritable communauté. Boulder même a connu une croissance rapide au cours des 15 dernières années et attire des dizaines de petites et moyennes entreprises techniques, dont beaucoup sont californiennes. L'afflux de talents et d'entreprises au Colorado est un atout pour l'État. Restent la longueur imprévisible d'un phénomène comme la COVID, les bouleversements qui en résultent et les conséquences possibles que nous n'avons même pas encore imaginées.
Quant au talent, je pense que cette question ne saurait être réduite à une vision locale. Le modèle consistant à planter un drapeau d'entreprise dans un foyer technologique régional pour y attirer des talents avait changé depuis un certain temps, et je pense que la COVID a encore accéléré ce changement. Avant la COVID, nous trouvions déjà des talents et des partenariats à l'échelle mondiale. Au fil de l'année 2020, ce mouvement aussi s'est accéléré. Le local, c'est où vous vivez ; les bonnes idées sont mondiales, le talent aussi.

Abate : Comment voyez-vous l'avenir de SparkFun et sa réponse aux besoins de sa communauté ?
Samala : Dans le monde de la technique, notamment dans un environnement libre (*open-source*), vous évoluez en créant une valeur nouvelle dans votre modèle. Parfois la création de cette valeur est la suite logique de ce qui a précédé, d'autres fois c'est un accident.
SparkFun évolue et teste de nombreux domaines qui apportent de la valeur : notre feuille de route de produits, SparkX, les partenariats, les services et les commentaires de la communauté. Nous nous efforçons d'avoir une croissance à la fois logique et fortuite. Nous comptons sur ce moyen, et quelques autres, pour continuer de répondre de façon satisfaisante aux besoins en constante évolution de nos clients actuels et futurs.

(200687-03 – VF Richard Kerr)

Circuits imprimés sur mesure avec le service ALC



Megan Hemmings (États-Unis)

SparkFun conçoit et fabrique des cartes depuis plus de dix ans. Nous avons vu nos clients passer de la recherche d'un accès plus facile aux composants électroniques, à celle de la technologie la plus récente et la plus performante puis à la création et la vente de produits de leur propre conception. Nos clients ont évolué, les offres de SparkFun aussi.

SparkFun À La Carte, ou ALC, est un service de conception et de fabrication de cartes électroniques sur mesure qui assure la transition du prototype à la production. Sa mission : répondre à l'évolution des besoins de nos clients, même si - et surtout **si** - ils n'ont pas d'expérience dans la conception de cartes.
Selon Nathan Seidle, ingénieur fondateur de SparkFun : « Ceux qui ont une idée ne sont pas tous ingénieurs en matériel ou n'en connaissent pas un pour les aider à concrétiser leur idée. La richesse d'ALC, c'est que tout le travail est fait pour vous. Ce que vous devez savoir, ce sont les composants que vous voulez utiliser. Notre système dessinera toutes les pistes et placera les blocs pour vous, ce qui permet à n'importe qui de créer une carte personnalisée ».
Né de la conviction de Seidle qu'il devait y avoir une meilleure façon de créer des cartes personnalisées, ALC élimine la nécessité de couper et de dénuder des fils, de souder des BoB, tout fixer et de tout recommencer pour la carte suivante. ALC (**fig. 1**) offre une interface simple, de type „pointer-cliquer", pour concevoir et fabriquer des cartes répondant aux besoins spécifiques de chacun.

Avec ALC, si votre idée est réalisable avec les BoB de SparkFun, vous recevrez une carte assemblée et personnalisée livrée à domicile en quelques semaines. Finies les séances de soudure nocturnes avant le grand jour. Profitez des conceptions, du savoir-faire en matière de circuits et des machines de SparkFun pour faire fabriquer un appareil professionnel juste pour vous. La **fig. 2** montre les blocs ALC.

L'interface d'ALC

L'interface d'ALC repose sur des blocs. Un bloc ALC contient le module à ajouter à la carte, ainsi que des informations sur les réseaux, les classes, les restrictions de connexion et autres besoins et contraintes de conception. En incluant toutes ces informations, ALC dispense les utilisateurs d'avoir ces profondes connaissances techniques que l'on n'acquiert généralement qu'après de nombreux essais et erreurs. Pour créer une carte avec ALC, les utilisateurs sélectionnent les différents blocs requis dans leur conception et le système se charge du reste, garantissant ainsi une carte électriquement correcte (**fig. 3**).

Gerbers et schémas

Toutes les commandes d'ALC comportent une version téléchargeable des fichiers Gerber et le PDF du schéma. Une fois la carte réalisée, les utilisateurs peuvent en acheter les fichiers source Eagle entièrement routés (.BRD et .SCH). Une fois ces fichiers débloqués, le schéma peut être modifié et personnalisé selon les besoins pour une fabrication

normale par SparkFun ou autre. Bien entendu, si aucune modification n'est nécessaire, l'utilisateur peut très facilement recommander les cartes avec ALC.

Glenn Samala, PDG de SparkFun, précise : « Avec SparkFun À La Carte, nous avons recueilli les années d'expérience de SparkFun en matière de conception et de fabrication de cartes et les avons regroupées dans une plateforme facile à utiliser par tous pour créer des cartes sur mesure. Avec cette offre, nous espérons compléter notre offre de services et rationaliser pour nos clients la démonstration de faisabilité pour mettre des produits sur le marché ».

Consultez le site d'ALC à l'adresse *https://alc.sparkfun.com/* pour démarrer la réalisation d'une carte personnalisée de votre cru,

Les quatre étapes de conception d'une carte dans SparkFun À La Carte

ALC Designer simplifie la conception de cartes personnalisées autant que possible. ALC vous guidera dans le choix du contrôleur, des composants, des connecteurs et de l'alimentation. Il suffit de sélectionner les blocs à ajouter à votre carte. S'il n'y a pas assez de broches pour supporter un composant, ALC ne vous laissera pas choisir ce bloc. Actuellement, ALC exige que toutes les cartes aient au moins un contrôleur, un composant et une alimentation pour commander.

Pour démontrer la simplicité de création d'une carte, voyons comment en fabriquer une pour commander un système de ventilation automatique avec mise en route d'un ventilateur d'extraction et surveillance à distance.

Étape 1 : Choisir un contrôleur. Dans ce cas, prenons le microcontrôleur phare de SparkFun, l'Artemis, qui permettra une extension du projet avec Bluetooth 5.0 et une énorme mémoire flash de 384 K de RAM pour le code (**fig. 4**).

Étape 2 : Ajouter les composants. Pour cette conception, il nous faut :

- L'afficheur **LCD 16×2** pour afficher les mesures
- Le **capteur de qualité de l'air BME680** pour mesurer la température, l'humidité et les COV totaux
- Un **codeur RVB** permettant de passer d'une valeur à l'autre pour la température, l'humidité et les COV totaux
- 3 × **pixel LED -apa102s** pour afficher l'état des indicateurs de COV – vert pour acceptable, jaune pour marginal, et rouge pour malsain
- Un **connecteur Qwiic** pour pouvoir ajouter plus de capteurs et de fonctions plus tard
- Une **LED** pour indiquer la mise sous tension de la carte
- Et un **relais pour activer** le purificateur d'air ou le ventilateur d'extraction lorsque des niveaux élevés de COV sont détectés

Les composants sont illustrés sur la **fig. 5**.

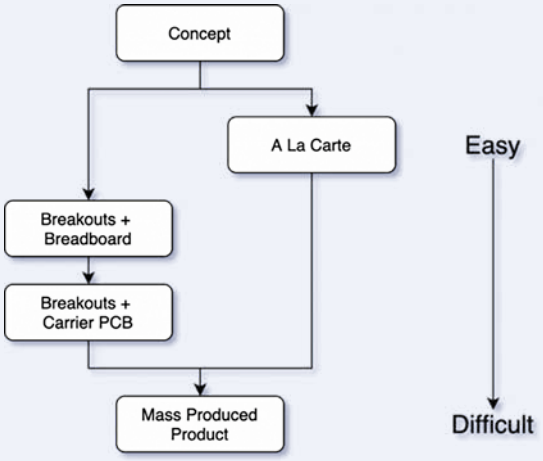


Figure 1. Niveaux de prototypage proposés dans SparkFun à la carte.

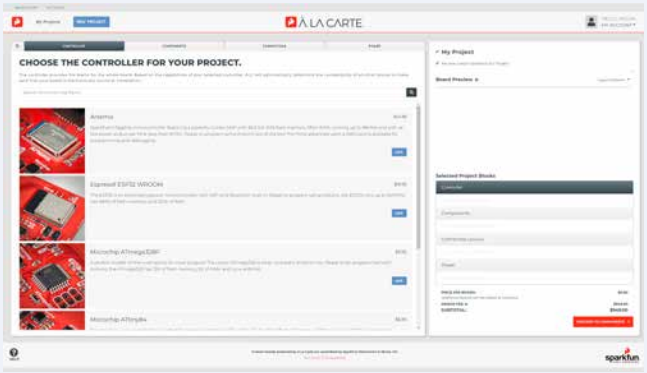


Figure 2. Vue d'ensemble des blocs ALC disponibles pour l'utilisateur.

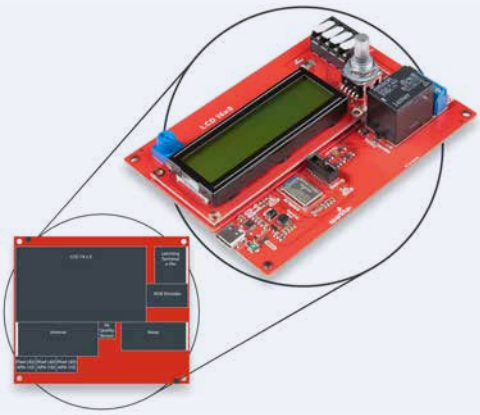


Figure 3. Un exemple de carte dans ALC Design

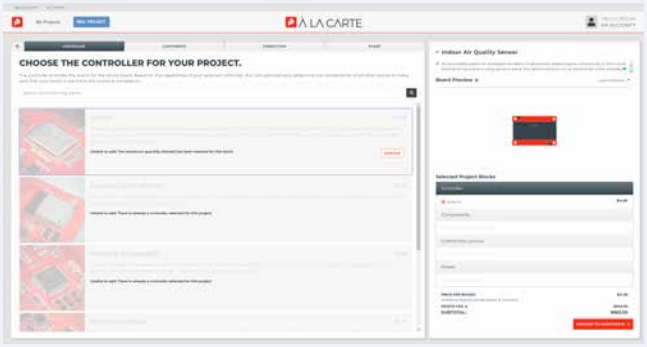


Figure 4. Choisissez votre contrôleur !

Étape 3 : Ajouter des connecteurs (facultatif). Les connecteurs peuvent être utilisés pour rediriger n'importe quel signal sur la carte vers une foule de types de connecteurs différents. Pour ce projet, ajoutons un bornier à verrouillage à 4 bornes (**fig. 6**) pour pouvoir connecter des composants supplémentaires plus tard, par exemple un capteur de particules.

Étape 4 : Ajouter un bloc d'alimentation. Pour ce projet, puisqu'il sera installé dans un bureau, nous utiliserons le **Wall Power** (**fig. 7**).

Voilà, la carte est prête. Comme vous pouvez le voir, ALC a généré un aperçu de la conception de la carte au fur et à mesure (**fig. 8**). Mais quelle excitation quand vous recevez la vraie carte à la fin comme sur la **fig. 9** !

SparkFun À La Carte vous permet d'aborder tout problème ou projet d'un point de vue entièrement personnalisable. Alors, imaginez vos plus grands rêves et faites-en une réalité dans ALC. Lancez votre propre projet sur http://alc.sparkfun.com.

Circuits réalisés avec À La Carte

Avec SparkFun À La Carte, les possibilités de conception sont presque infinies. Voici quelques exemples de cartes créées avec ALC.

Commande d'un four de refusion

Objectif : créer une carte de contrôle de la température dans un four de refusion fait maison (**fig. 10**).

Blocs utilisés :

- Contrôleur : Artemis
- Composants :
  - Codeur RGB - pour permettre la navigation dans les menus et faire des sélections
  - Afficheur LCD 16x2 caractères - pour afficher les informations nécessaires
  - (2) Amplificateur de thermocouple - pour mesurer la température du four
  - (2) Relais - pour commander la température du four
- Alimentation : Alimentation USB

Cuisine

Objectif : un appareil IdO qui peut surveiller et rendre compte de n'importe quel projet en cuisine (**fig. 11**).

Blocs utilisés :

- Contrôleur : ESP32 WROOM
- Composants :
  - Afficheur LCD 16x2 - pour afficher des données telles que minuteries, température, poids et plus
  - Thermocouple - pour mesurer la température de manière sûre et précise
  - Balance numérique - parce que les balances sont toujours pratiques dans une cuisine, pour surveiller l'évolution du poids dans le temps (comme le poids d'une étagère dans mon réfrigérateur), ou le poids d'un morceau de viande en cours de cuisson
  - Connecteur Qwiic - pour permettre des extensions et des ajouts
- Alimentation : par batterie

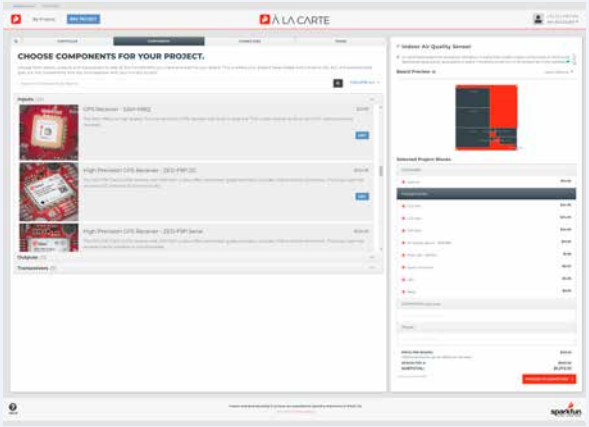


Figure 5. Sélectionnez votre composant !

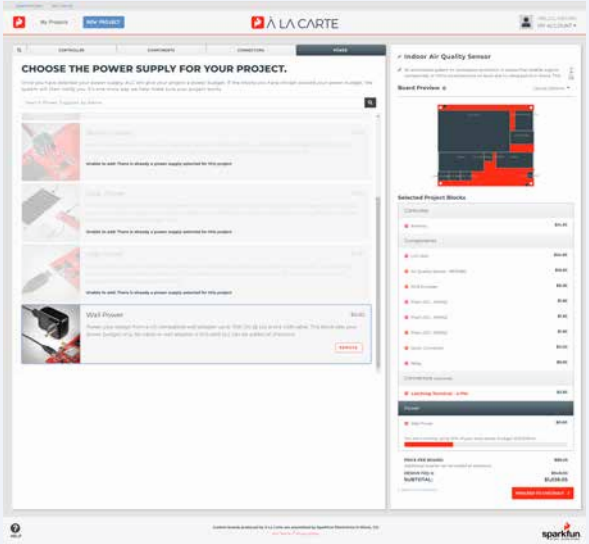


Figure 7. Sélection de l'alimentation électrique – choix d'un chargeur mural.

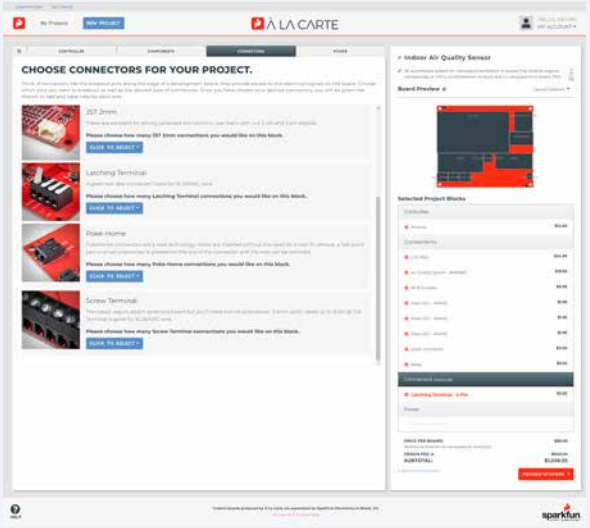


Figure 6. Bornier à verrouillage à 4 bornes.

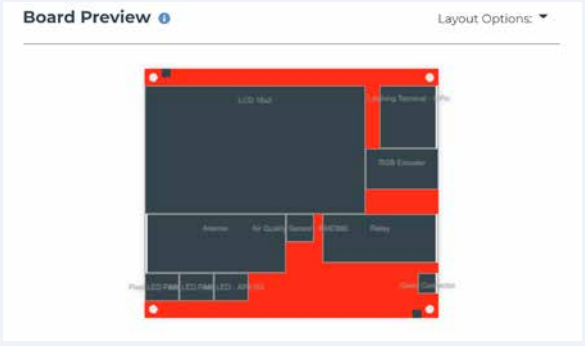


Figure 8. Aperçu d'une carte ALC.

Figure 9. Véritable carte ALC.

Figure 10. Carte pour four de refusion.

Figure 11. Carte Cuisine - Allons cuisiner !

Figure 12. Carte pour station météo.

Figure 13. Carte pour poulailler.

Figure 14. Portier RFID.

Météo Head

Objectif : créer un nœud de capteurs environnementaux pour commander des ventilateurs et permettre l'extension avec d'autres capteurs via Bluetooth (**fig. 12**).

Blocs utilisés :

- Contrôleur : Artemis
- Composants :
  - (2) Bouton - parce qu'il faut des boutons partout
  - Météo - pour permettre l'ajout d'un pluviomètre, d'un anémomètre et d'une girouette
  - (2) LED - pour faciliter le débogage
  - Interrupteur de courant fort - Borniers à verrouillage - pour brancher des accessoires supplémentaires tels qu'un ventilateur et pouvoir les commander via Bluetooth
- Capteur de qualité de l'air - BME680 - pour suivre pression, température, humidité et COV de l'air
- Alimentation : alimentation 12 V de voiture

ChickenNet

Objectif : créer une carte IdO pour surveiller un poulailler à distance et hors de vue (**fig. 13**).

Blocs utilisés :

- Contrôleur : ESP32 WROOM
- Composants :
  - Ruban à LED RVB - pour fournir aux poules la durée d'éclairage nécessaire à la ponte hivernale
  - Balance - pour surveiller le poids d'une boîte de ponte et en déduire si elle est occupée et/ou si un œuf y a été pondu
  - Capteur de mouvement - pour détecter la présence des poules dans le poulailler lui-même
  - BME280 / CCS811 - pour surveiller température, humidité, pression atmosphérique et qualité de l'air
  - RFM95 LoRa Radio - pour des applications sans fil étendues en l'absence de Wi-Fi.
- Alimentation : par USB

Portier RFID

Objectif : un portier simple et programmable (**fig. 14**).

Blocs utilisés :

- Contrôleur : ATmega328p
- Composants :
  - RFID de base - 125 kHz - pour reconnaître et lire les clés RFID
  - Interrupteur de courant fort – avec prise jack - pour activer une gâche électronique
  - Détecteur d'ouverture de porte - pour que le contrôleur puisse savoir si la porte est maintenue ouverte
  - (2) Bouton – on a toujours besoin d'un petit bouton chez soi
- pixel LED - APA102 - pour faire un indicateur d'état
- Alimentation : chargeur mural

(200690-2 VF : Denis LAFOURCADE)

Prise en main du **module BLE Artemis** pour l'apprentissage automatique

Nathan Seidle (États-Unis)

Le module Artemis est le premier module open-source sans-fil pour l'apprentissage automatique (reconnaissance vocale), avec liaison Bluetooth à faible consommation (BLE). Quelle puissance étonnante concentrée dans une carte de 10 × 15 mm ! Voici un tutoriel pour vous guider à travers les possibilités du module Artemis et vous aider à établir les bases de son intégration dans vos propres projets !

SparkFun a longtemps éludé l'obtention d'un certificat *FCC*. La démarche restait opaque et le coût rédhibitoire. Cependant, nous avions déjà travaillé sur les exigences de la *FCC* pour les produits et les projets de loisirs. Avec le développement du *SparkFun Edge* et la création d'*Artemis*, le moment de nous jeter à l'eau était arrivé.

En octobre 2018, nous avons rencontré une équipe de *TensorFlow* pour faire une carte à faible consommation avec la nouvelle *Apollo3* d'*Ambiq*. À l'examen rapide de ses caractéristiques, il apparut que cette carte était aussi stimulante qu'excitante ! 1 Mo de flash, près de 400 Ko de RAM et une microconsommation : nous pouvions rêver de possibilités bien au-delà de l'ancienne Uno. Pour relever le défi du BGA à 81 billes avec un pas de 0,5 mm, il fallait hisser notre savoir-faire d'un cran. Les difficultés rencontrées et surmontées sont relatées dans un blog de 2019 ; *www.sparkfun.com/news/3122*.

Avant d'aborder *Artemis*, nous vous recommandons de consulter ces tutoriels.

- Carte à circuit imprimé, bases. Qu'est-ce ? Ce tutoriel présente les constituants d'une carte à circuit imprimé (PCB) et certains termes d'usage courant dans ce domaine. [1]
- Utilisation d'EAGLE : agencement de la carte. La 2e partie du tutoriel couvre la façon de disposer les composants sur une carte une fois son schéma dessiné. [2]
- Bluetooth, bases. Aperçu de la technologie sans fil *Bluetooth*. [3]
- Programmation *ARM*. Programmation des cartes SAMD21, SAMD51 ou autres processeurs ARM. [4]

Aperçu du matériel

Cette partie couvre les détails techniques d'*Artemis*, comme ses caractéristiques physiques et électriques. Si vous avez déjà une carte de développement dotée d'un module Artemis, vous pouvez sauter cette partie et vous rendre aux **Caractéristiques exclusives**. Cela dit, vous souvenez-vous de ce que cache ce module ? Jetez un coup d'œil à la **figure 1**.

Apollo3

Le noyau du module Artemis est l'*Apollo3* d'*Ambiq* [5]. Il s'agit d'un *ARM Cortex-M4F* (F indique des opérations en virgule flottante) avec 1 Mo de flash et 384 K de RAM. La fiche technique est disponible sur [6].

Antenne BLE

L'Apollo3 comporte une puce RF Bluetooth 5.0 intégrée et le module Artemis a une antenne 2,4 GHz soudée (gain de 2 dBi).

Convertisseur CC abaisseur embarqué

L'Apollo3 peut fonctionner de **3,6 à 1,8 V**. Une plage aussi large, est obtenue par 2 régulateurs CC intégrés qui abaissent la tension VCC d'entrée à la tension de la puce avec un rendement dépassant 80 %. Le module Artemis comprend 2 selfs pour réduire au mieux la consommation d'énergie.

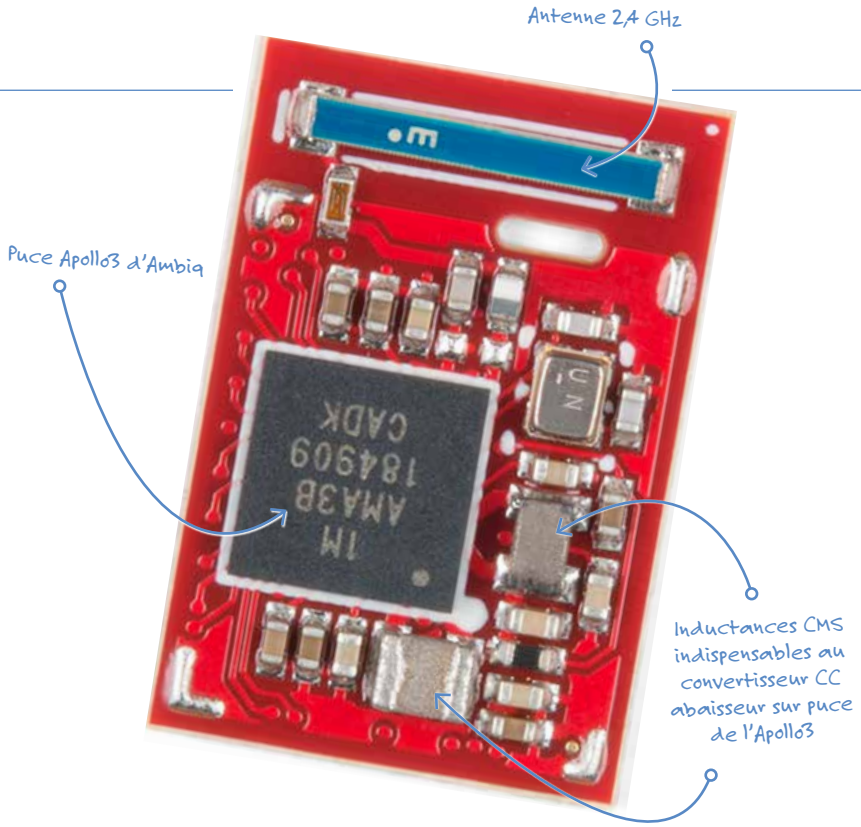


Figure 1. Blindage RF ôté, les éléments remarquables du module Artemis apparaissent.

Figure 2. Artemis, minuscule et léger.

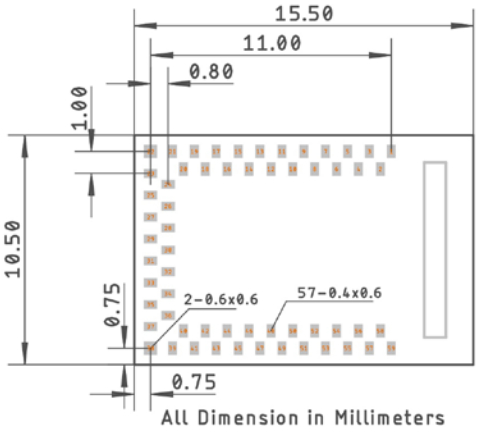


Figure 3. Empreinte CMS recommandée pour le module Artemis. Vue de dessus.

Dimensions

Le module Artemis mesure 15,5 × 10,5 × 2,3 mm et pèse 0,6 g (**fig. 2**).

Empreinte recommandée

La **figure 3** montre la disposition recommandée du circuit imprimé d'accueil du module. Les dimensions et considérations spécifiques sont consignées dans le guide d'intégration Artemis [7]. Si vous utilisez *EAGLE PCB*, clonez simplement l'un de nos modèles open-source qui utilisent Artemis (*RedBoard* [8], *RedBoard Nano* [9], et *RedBoard ATP* [10]) et commencez à dessiner votre carte ! Artemis apparaît vu de dessous sur la **figure 4**.

L'Apollo3 est un CI puissant, mais son boîtier BGA de 0,5 mm nécessite une carte imprimée à 4 couches avec des vias enfouis et remplis d'époxy. Cela rend la carte coûteuse et difficile à produire. Nous avons conçu le module Artemis pour vous débarrasser de ces soucis.

Une carte accueillant Artemis (**fig. 5**) ne nécessite qu'un circuit imprimé à 2 couches et des pistes/espacements de 8 mil. Le routage sous le module est autorisé. Éloignez tous les conducteurs de masse de la zone d'antenne. Si l'exposition mécanique le permet, l'antenne peut être étendue sur le bord de la carte pour améliorer la réception.

Caractéristiques exclusives

Les fonctionnalités d'Artemis sont impressionnantes. Nous allons en donner un aperçu, mais n'oubliez pas d'étudier les exemples inclus dans le noyau d'Arduino ainsi que le SDK d'Ambiq pour en savoir plus.

Souplesse d'affectation des pastilles CMS

Le module Artemis dispose de 48 GPIO à capacité d'interruption et d'une série d'autres périphériques. Les 2 UART matérielles peuvent être réaffectées à une série d'autres pastilles, et il y a 16 sorties PWM totalement indépendantes. Un CAN rapide à 14 bits est connecté à 10 pastilles. Utilisez les maîtres SPI ou I²C disponibles sur 6 jeux de pastilles.

C'est loin d'être tout – la couche d'abstraction matérielle (HAL) met à disposition PDM (modulation de densité d'impulsions), SSC , et DMA (accès mémoire direct). Consultez la carte des fonctions des broches d'Apollo3 [11], la fiche technique d'Apollo3 [12] et le SDK/HAL d'Ambiq [13] pour plus d'informations. Ne craignez rien, nous avons de nombreux exemples d'utilisation des E/S.

Cortex-M4F

Artemis exploite une puce à noyau l'Apollo3 d'Ambiq. Ce dernier comprend un ARM Cortex-M4F fonctionnant à 48 MHz avec un mode rafale optionnel à 96 MHz. Ce puissant noyau peut être programmé avec GCC ainsi qu'avec Keil, IAR, et débogué avec

Figure 4. Module Artemis de dessous.

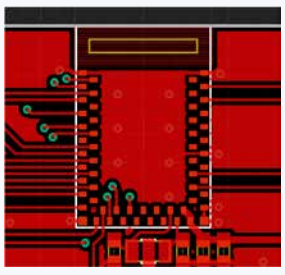


Figure 5. Exemple d'Artemis sur un circuit imprimé.

Ne manquez pas le train Artemis

Pour tout savoir sur Artemis, son historique et sa disponibilité, consultez ces pages :

Création d'Artemis :
www.sparkfun.com/news/3122
Création d'une carte fille RF :
www.sparkfun.com/news/3123
Obtention d'un module certifié FCC :
www.sparkfun.com/news/3124
Communiqué de presse :
https://bit.ly/3b2Zpd7
Création d'un blog RF Shield :
https://www.sparkfun.com/news/3123
Page écosystème :
https://www.sparkfun.com/artemis
Article du blog Hackster :
https://bit.ly/38XRX0i

divers outils JTAG récents. [14]

BLE

Artemis intègre un module Bluetooth 5.0 capable de transmettre jusqu'à 4 dBm, ce qui devrait vous permettre d'atteindre une portée d'environ 70 m. Nous avons vu des commandes RSSI passées à plus de 70 m.

PWM

Avec 31 des 48 broches activées PWM, vous avez de quoi faire ! Consultez la fiche technique graphique [15] et la carte des broches pour vérifier quelles broches ont des capacités PWM.

Interruptions

Chaque broche peut être configurée pour produire une interruption et sortir le processeur du sommeil profond. De plus, 29 des 49 broches ont une résistance de rappel interne activable par logiciel.

CAN à 14 bits

Alors que l'Uno d'origine avait un CAN à 10 bits, Artemis a un CAN à 14 bits - ce qui signifie que l'échelle de conversion *0 à 1023* passe à *0 à 16383*. Cela permettra des lectures plus précises de capteurs analogiques (pression, lumière, son, etc.). Notez, cependant, que le CAN mesure de **0 V à 2 V**. Donc si vous avez un capteur qui sort de 0 à 3,3 V, il est sans danger, mais saturera le CAN dès 2 V. Utilisez la fonction `setResolution` pour modifier la résolution des lectures entre 10 et 14 bits (10 par défaut). De plus, le CAN est **beaucoup** plus rapide (jusqu'à 1,2 Méch/s), ce qui permet d'agréger davantage de données.

Faible consommation

Ambiq, le fabricant de l'Apollo3, a fait des années de recherche sur ce qu'il appelle la technologie *SPOT™* (optimisation de la puissance sous-le-seuil). Ce sigle recouvre une technique très basse consommation obtenue en abaissant les tensions qui matérialisent un 1 ou un 0 logique. En agissant au niveau du silicium, Ambiq a réussi à mettre au point un processeur à 48 MHz consommant moins de 0,5 mA. La surveillance „permanente" de commandes vocales, sans BLE ni connexion à Internet, absorbe environ 6 µA/MHz.

Mode rafale

Parfois, 48 MHz ne suffisent pas. Le processeur peut entrer en mode rafale à 96 MHz, ce qui permet d'effectuer des calculs internes et un contrôle en deux fois moins de temps.

Note : les opérations sur les broches externes sont limitées à 48 MHz.

JTAG

La puissance des puces actuelles exige des outils de débogage modernes. Artemis est basée sur un Cortex-M4F qui possède un port JTAG réservé au débogage. Un débogueur JTAG permet de définir des points d'arrêt, d'inspecter les registres et de voir quelles instructions (assembleur et C) sont exécutées. C'est un outil dont vous n'aurez pas souvent besoin, mais si c'est le cas, il vous sortira d'affaire.

PDM

La reconnaissance vocale „toujours active" est un des atouts remarquables d'Artemis. Les microphones MEMS numériques sont plus sensibles et plus faciles à utiliser que leurs ancêtres analogiques. Artemis comporte un port PDM qui permet d'utiliser jusqu'à deux microphones MEMS, soit en mode bivoie, soit en mode de filtrage spatial.

Les résistances de rappel internes

Chaque broche a un faible rappel interne au + (activé par logiciel). En outre, les broches configurées comme ports I2C ont des résistances de rappel au + sélectionnables par logiciel (1,5 kΩ, 6 kΩ, 12 kΩ, 24 kΩ). Il n'y a plus besoin de résistances externes sur SDA et SCL.

Courant délivré sur les broches

Une exclusivité Artemis : le courant de sortie de toutes les broches GPIO est programmable. Pour chaque broche un courant maximal de 2, 4, 8 ou 12 mA peut être sélectionné. En outre, les pastilles 3 et 36 permettent de sélectionner des commutateurs de puissance côté VDDH, (R_{on} ~1 Ω). Les pastilles 37 et 41 permettent de sélectionner des commutateurs de puissance côté VSS, (R_{on} ~1 Ω).

> *« La transition du prototype au produit final a toujours été chaotique – Artemis y met bon ordre en proposant un même module du prototype à la production. Ajoutez à cela l'efficacité de la plateforme d'apprentissage machine TensorFlow® et la puissance du microcontrôleur Apollo3 d'Ambiq®, et vous obtenez un outil séduisant ».*
>
> Nathan Seidle, ingénieur, fondateur de SparkFun

Jargon (wikipedia.org)

BGA : Matrice de billes
SPOT : Subthreshold Power-Optimized Technology
HAL : Couche d'abstraction matérielle
HAL : Hardware abstraction
PDM : Pulse-density modulation
Make : Makefile

Une grande première

Approbation de la FCC/IC/CE !

Le module Artemis de SparkFun a reçu l'approbation de la FCC (Commission fédérale des communications), de l'IC (Industrie Canada) et le label européen CE, ce qui en fait le premier module BLE open-source, fabriqué aux États-Unis et certifié FCC/IC/CE. Cette certification du module Artemis permet aux concepteurs de produits d'utiliser le même module du prototype à la production, et facilite notablement l'apprentissage machine à faible consommation.

Sécurité

Le *Cortex-M4F* d'Artemis possède plusieurs couches de sécurité, dont le démarrage sécurisé, l'OTA, le stockage de clés, ainsi que le cryptage/décryptage à la volée de mémoire flash externe (comme une carte SD).

Programmation

Artemis se programme via l'interface JTAG standard ou un chargeur d'amorçage série (*serial bootloader*). Le chargement série est confié à un CH340 relié au connecteur USB [16] (**fig. 6**) ou, pour une programmation et un débogage avancés, diverses cartes de développement de SparkFun ont une empreinte JTAG. Pour plus d'informations sur la programmation ARM, dont les interfaces JTAG, consultez notre *tutoriel de programmation ARM*. [17]

Chargeur d'amorçage SparkFun

Notre chargeur d'amorçage à débit flexible (*SVL*) est lancé à chaque mise sous tension. Que signifie exactement *à débit flexible* ? L'ordinateur et le chargeur ouvrent la communication à 9600 Bd, puis s'accordent à choisir un débit plus rapide pour le transfert du gros des données binaires. La vitesse de téléchargement peut atteindre 921600 Bd et réduire beaucoup le temps de téléchargement. En cas d'erreurs de communication à débit élevé, le *débit flexible* permet de sélectionner le débit qui fonctionne le mieux. Pour télécharger des croquis et des programmes, ce chargeur est l'outil préféré de ceux qui veulent un moyen rapide et fiable de charger de nouveaux programmes dans Artemis.

Une fois une carte cible Artemis sélectionnée, des options supplémentaires apparaîtront par la suite à l'ouverture du menu *Outils*. Les options **SVL Baud Rate** (**fig. 7**) permettent de choisir la vitesse de téléchargement. 921600 bps est la vitesse recommandée car c'est la plus rapide pour mettre à jour des croquis. Cependant, il existe certaines plateformes (Linux) où les pilotes USB-série CH340 standard ne fonctionnent pas toujours bien à plus de 115200 Bd. Donc, si vous avez des erreurs de téléchargement, pensez à réduire la vitesse. Vous trouverez d'autres informations sur le forum [18], et pensez à mettre à jour les pilotes pour Mac OSX [19] ou Linux [20]. Tout comme sur les classiques Arduino Uno, Arduino Mega, etc., le chargeur est activé en réinitialisant la carte. Un simple condensateur de 0,1 µF entre les broches DTR et Reset suffit pour qu'Artemis se réinitialise et entre en mode chargement. Si au bout de 50 ms aucun nouveau microprogramme n'est détecté, le code utilisateur est exécuté. Si les discussions spécialisées (par ex. sur les chargeurs d'amorçage) vous intéressent, vous pourrez en savoir plus sur celui d'Artemis [21].

Chargeur d'amorçage Ambiq

En plus du chargeur SparkFun Artemis, nous programmons chaque Artemis avec le *Secure Bootloader* (SBL) Ambiq. Ce chargeur sera mis à profit pour les mises à jour de bas niveau des périphériques qui nécessitent une provenance sécurisée. Le chargeur est activé à la réinitialisation si la broche 47 est à 1. Il communique à 115200 Bd. Le chargeur attend alors indéfiniment de nouvelles données binaires. Un outil python et un exécutable pour commu-

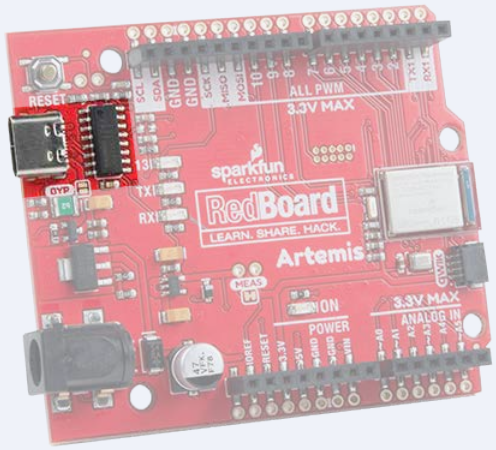


Figure 6. Connecteur USB sur RedBoard Artemis, pour le chargeur série doté du CH340.

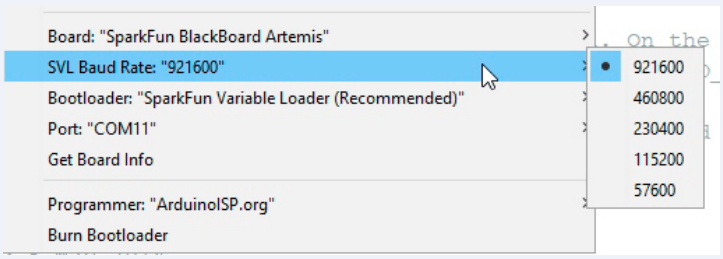


Figure 7. Téléchargez votre croquis très rapidement avec le SVL réglé sur 921600 Bd.

niquer avec ce chargeur sont fournis.
Ce processus de chargement est différent de celui auquel vous êtes peut-être habitué. Sur la plupart des Arduino avec *ATmega328*, le chargeur *STK500* est lancé à l'initialisation et à la fin du temps imparti, le code de l'utilisateur est exécuté. Le chargeur Artemis est similaire mais une autre broche (*Bootload*) doit être maintenue à 1. Pour une mise en œuvre d'Artemis aisée et bon marché, nous avons conçu un petit circuit RC à ajouter à votre réalisation : un circuit intégré USB-série doté du minimum de broches de contrôle (le CH340E n'a que le RTS) et permet néanmoins l'activation du chargeur d'amorçage Ambiq. Si vous devez modifier le chargeur SparkFun Artemis (décrit ci-dessus) ou utiliser la chaîne d'outils de chargement sécurisé, le circuit de la **figure 8** convient pour le chargement et utilise une seule broche (DTR ou RTS sont pris en charge).

Attention ! Vous n'endommagerez jamais l'Artemis, mais l'utilisation des outils Ambiq Secure Bootloader écrasera le bootloader SparkFun, supprimant la possibilité de téléchargement plus rapide. L'utilisation du chargeur Ambiq Secure Bootloader pour la programmation générale d'Arduino n'est pas recommandée **(fig. 9).**

« Dans notre domaine, commercialiser un produit a toujours été et reste difficile. Les démarches sont compliquées, les obstacles nombreux, et là, nous pouvons vous aider. Artemis facilite le passage rapide de grandes innovations à l'échelle industrielle ».

Glenn Samala,
PDG de SparkFun

Pour charger un nouveau code sur votre module Artemis avec la chaîne d'outils du chargeur Ambiq, sélectionnez l'option *Ambiq Secure Bootloader* dans le menu Arduino *Tools -> Bootloader*. Ces outils modifieront votre binaire, et l'encadreront de différents en-têtes de sécurité. Le chargement du code à 115200 Bd peut échouer. Si c'est le cas, cliquez à nouveau sur *upload*.

Fonctionnement du circuit RC à une broche

Le module est réinitialisé en mettant DTR (ou RTS) à 0. Au bout de 10 ms, le logiciel met DTR à 1. La broche de chargement reste à 1 pendant 100 ms, ce qui permet d'exécuter le chargeur. L'ouverture d'un port série met DTR à 0, ce qui réinitialise le module, mais comme DTR reste à 0 pendant les communications série normales, le module ne lance pas le SBL et continue à faire fonctionner le Bootloader Artemis de SparkFun.
Nous avons modifié l'outil de chargement python d'Ambiq pour que DTR et RTS soient pilotés à l'identique. Ainsi vous pouvez utiliser soit RTS soit DTR pour lancer le chargeur Artemis. Nos outils Ambiq SBL [22] font ensuite passer DTR/RTS à 1 pour lancer le chargeur d'Ambiq.
Si vous préférez, un bouton peut être inséré sur la broche de chargement coupée. Si l'utilisateur maintient ce bouton enfoncé et réinitialise la carte, elle entre en mode de chargement et y reste jusqu'à ce qu'un cycle de chargement se termine ou qu'une réinitialisation se produise. Cette méthode fonctionne bien, mais nécessite l'intervention de l'utilisateur à chaque fois qu'un programme doit être chargé.

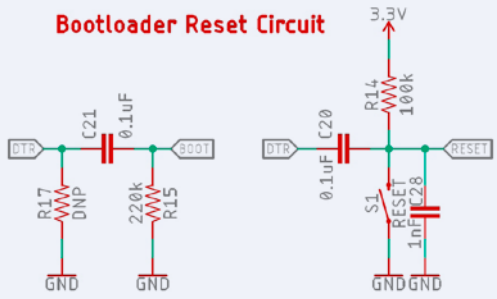


Figure 8. Ce schéma d'initialisation et d'amorçage à 1 broche pour Artemis est idéal pour tout convertisseur USB-série dont les broches de contrôle sont accessibles (CH340, CP210x, FT232, etc.).

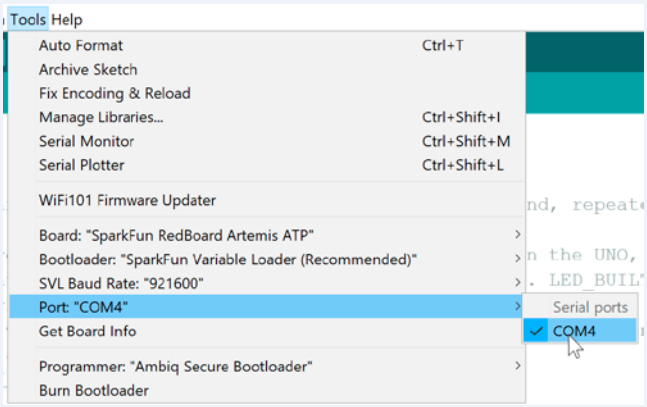


Figure 10. Sélection du port COM.

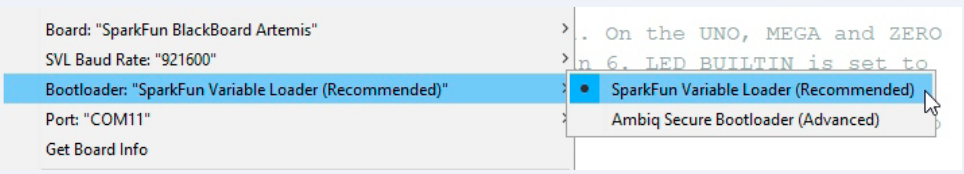


Figure 9. Ne cochez pas Ambiq Secure Bootloader si vous n'êtes pas sûr de vous.

Diagnostic des pannes

Besoin d'aide ? Si votre produit ne fonctionne pas comme prévu ou si vous avez besoin d'assistance technique ou d'informations, allez sur les forums SparkFun [23]. Vous pourrez demander et trouver de l'aide. Si c'est votre 1ère visite, il faudra créer un compte [24] pour parcourir les *forums produits* et y poster des questions. Le lien [25] ouvre directement les forums Artemis de SparkFun.

Q. Par erreur, j'ai utilisé le bootloader Ambiq. Maintenant, le SparkFun Variable Loader ne fonctionne plus. Que dois-je faire ?

R. Vous n'avez pas pu vous en empêcher et vous avez chargé du code avec le chargeur Ambiq comme nous l'avons indiqué **figure 9**. C'est bon ! Pour remettre le chargeur d'amorçage SVL SparkFun, procédez comme suit :

Étape 1 : Sélectionner la bonne carte et le bon port COM dans l'EDI Arduino.

Vérifiez que vous avez sélectionné la bonne carte et le bon port COM dans le menu *Outils*. COM 4 est indiqué sur la **fig. 10**, mais votre port COM peut être différent.

Accessoires

Les principaux accessoires cités dans cet article sont disponibles chez Elektor et SparkFun !

Module SparkFun Artemis - Apprentissage machine avec Cortex-M4F et BLE à faible consommation

RedBoard Artemis

RedBoard Artemis Nano

RedBoard Artemis ATP

OpenLog Artemis

Carte de développement Edge - Apollo3 Blue

www.elektormagazine.fr/esfe-en-artemis

Developpement Artemis avec Arduino

Avec le noyau Arduino-Artemis des puissantes cartes RedBoard Artemis, RedBoard Artemis Nano et RedBoard Artemis ATP), vous mettrez moins de 5 min pour faire clignoter une LED !

Guide de raccordement de la RedBoard Artemis

Débutez avec la RedBoard Artemis – toutes les fonctions du module Artemis logées une carte au format Uno.

Étape 2 : Sélectionner *Graver la séquence d'initialisation* dans le menu *Outils*.
La **figure 11** indique la sélection qui amènera Arduino à utiliser le chargeur Ambiq pour recharger le SparkFun Variable Loader via le port série.
Étape 3 : Sélectionner à nouveau SVL comme chargeur.
Ne recommençons pas, d'accord ? Choisissez le chargeur SVL (**fig. 12**). Vos croquis seront de nouveau téléchargés à grande vitesse.

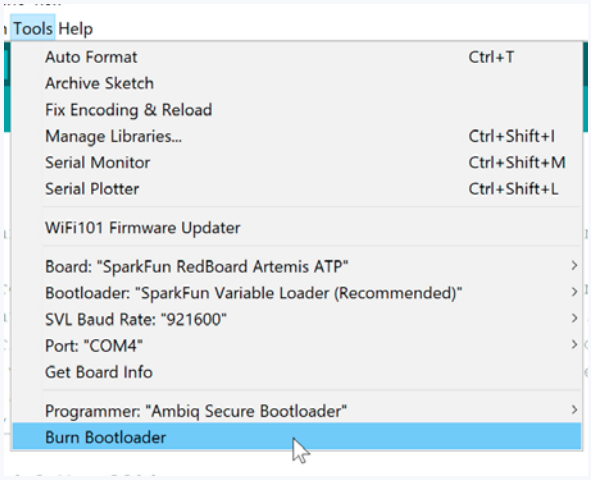


Figure 11. Dans le menu Outils, sélectionnez : Graver la séquence d'initialisation.

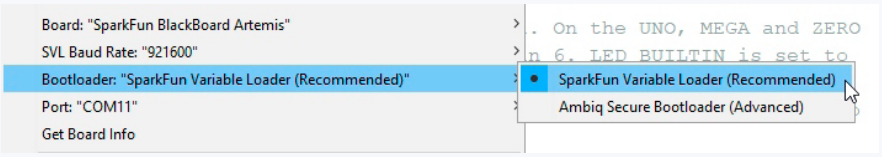


Figure 12. Oui, c'est comme la figure 9. Cochez cette fois **chargeur variable SparkFun**.

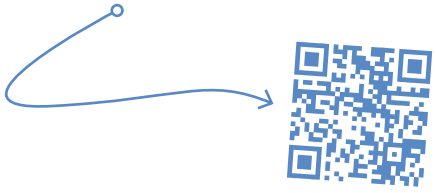
Pour approfondir

Les liens suivants (voir à la fin de l'article) sont particulièrement utiles pour accéder rapidement aux ressources d'Artemis : [5], [7], [11], [12], [13], [14], [17], [21] et [22].
Prêt à utiliser Artemis dans l'EDI Arduino ? Consultez le tutoriel sur le *développement Artemis avec Arduino*. [27]
Êtes-vous à l'aise avec un fichier *make* et un autre EDI ? Suivez le tutoriel sur la configuration du SDK Ambiq Apollo3. [28]

Artemis a été conçu et validé pour toute la gamme de plus de 100 cartes Qwiic de SparkFun. Consultez notre site, inspirez-vous des exemples et branchez simplement capteurs et sorties désirés pour réaliser une application que, sans ces accessoires, vous auriez crue hors de votre portée !

(200685-01 – VF Yves Georges)

Espace Elektor en ligne pour cet article :
www.elektormagazine.fr/esfe-en-artemis

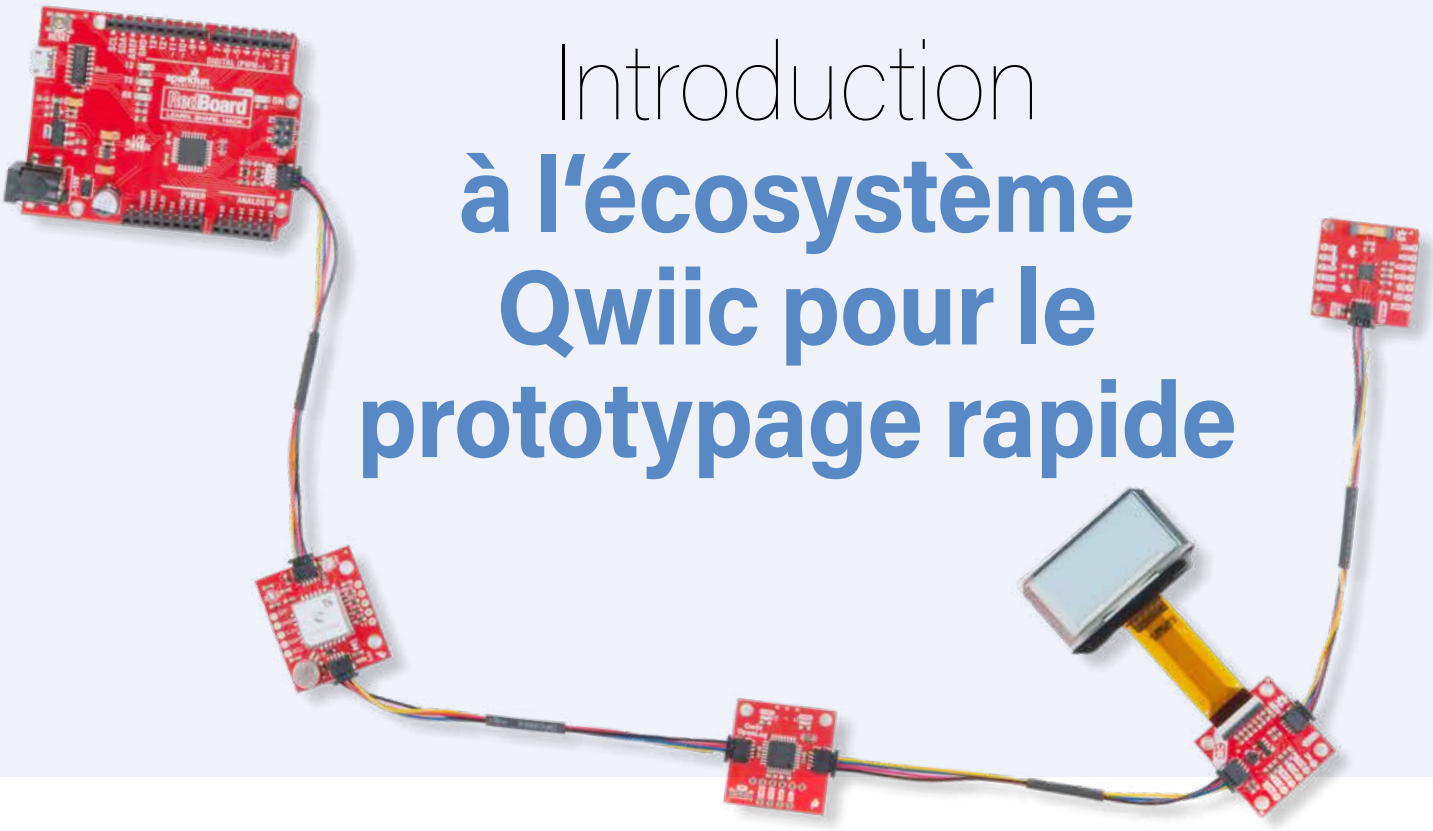


TensorFlow ambiq micro

L'union fait la force !
Artemis est le fruit d'une intense collaboration des équipes de TensorFlow et d'Ambiq Micro.

LIENS

[1] Cartes à circuit imprimé, bases : https://learn.sparkfun.com/tutorials/pcb-basics
[2] Utilisation d'EAGLE : https://bit.ly/3rLBDYW
[3] Bluetooth, bases : https://learn.sparkfun.com/tutorials/bluetooth-basics
[4] Programmation ARM : https://learn.sparkfun.com/tutorials/arm-programming
[5] Ambiq: https://ambiq.com
[6] Fiche technique d'Ambiq Apollo3 : https://bit.ly/3b8uxb8
[7] Guide d'intégration Artemis : https://bit.ly/3rLALn8
[8] RedBoard Artemis: https://www.elektormagazine.fr/esfe-en-artemis2
[9] RedBoard Artemis ATP: https://www.elektormagazine.fr/esfe-en-artemis3
[10] BlackBoard Mega: https://www.elektormagazine.fr/esfe-en-artemis4
[11] Fonctions des broches d'Apollo3 : https://bit.ly/3hHtY9F
[12] Fiche technique d'Apollo3 : https://bit.ly/2LkjIIh
[13] Ambiq SDK/HAL: https://ambiq.com/apollo3-blue/
[14] Outils JTAG : https://www.sparkfun.com/categories/tags/jtag
[15] Fiche technique graphique d'Artemis :: https://bit.ly/39eBG7p
[16] Chargement via série-USB CH340 : https://bit.ly/3rNBCnz
[17] Tutoriel de programmation ARM : https://learn.sparkfun.com/tutorials/arm-programming
[18] Article de forum : https://bit.ly/3rUDJ8O
[19] Pilotes pour Mac OSX : https://bit.ly/354YgOb
[20] Pilotes pour Linux : https://github.com/juliagoda/CH341SER
[21] Le chargeur d'amorçage Artemis : https://bit.ly/392ZEIA
[22] Outils Ambiq SBL : https://bit.ly/38a1qIM
[23] Forums SparkFun : https://forum.sparkfun.com/index.php
[24] Créer un compte sur le forum : https://forum.sparkfun.com/ucp.php?mode=register
[25] Forum Artemis : https://forum.sparkfun.com/viewforum.php?f=163
[26] Tutoriel de conception Artemis avec Arduino : https://bit.ly/2XbG8hk
[27] Installation du SDK Ambiq Apollo3 : https://bit.ly/3ncdbfX
[28] Cartes SparkFun Qwiic : https://www.sparkfun.com/qwiic



Introduction à l'écosystème Qwiic pour le prototypage rapide

Chris McCarty (États-Unis)

En avril 2017 sortaient les premières cartes Qwiic de SparkX, le labo de SparkFun. Le point de départ de Qwiic c'est le fait que le fondateur de SparkFun, Nathan Seidle, en avait assez de souder les mêmes quatre barrettes mâles, puis de relier les quatre fils à une carte de développement. Il voulait des petits connecteurs avec détrompage et des fils à code de couleur, adaptés au montage en chaîne et facilitant la communication entre cartes via I²C.

À son lancement, plutôt modeste, le système Qwiic Connect ne comptait que sept produits. Depuis, la gamme Qwiics'est étoffée et compte plus de 150 cartes, kits et connecteurs originaux produits par SparkFun. De nombreuses entreprises partenaires ont suivi le mouvement, soit en créant leurs propres cartes Qwiic, soit en fournissant à leurs utilisateurs les moyens de connecter des appareils existants à l'écosystème. Les équipes de développement, les prototypistes et les amateurs ont adopté Qwiic comme système pour simplifier le câblage et limiter les soudures.

Qwiic Connect est un écosystème de capteurs, d'actionneurs, de shields et de câbles I²C qui accélèrent le prototypage et le rendent plus sûr. Grâce au connecteur JST à 4 broches avec détrompage pour éviter la permutation accidentelle des liaisons SDA et SCL sur une carte d'expérimentation, le choix parmi la multitude de cartes-compatibles Qwiic n'est pas difficile.

Avec au moins deux connecteurs sur la plupart de nos cartes Qwiic, il est facile de les enchaîner par le bus I²C. Cela permet la connexion de milliards de combinaisons de capteurs, d'accessoires et de cartes de développement Qwiic.Voulez-vous lire les données sur un afficheur LCD à partir d'un capteur environnemental avec des capacités denavigation à l'estime ? Rêvez-vous de fabriquer vous-même un combiné et casque de réalité virtuelle ? Ce ne sont là que quelques exemples de ce que vous pouvez faire avec Qwiic sans souder quoi que ce soit.

On nous interroge souvent sur l'envergure de Qwiic. Cet écosystème est divisé en six grandes catégories : cartes de développement, périphériques, capteurs, accessoires, kits et câbles. Chacune comprend plusieurs sous-sections qui définissent plus précisément ce dont vous pourriez avoir besoin pour un projet spécifique. Voyons de plus près ce qu'offre chaque catégorie Qwiic !

Cartes de développement

Les cartes de développement Qwiic sont votre point d'entrée de tout l'écosystème. On trouve dans cette catégoriedes cartes au facteur de forme familier de l'Arduino R3 avec un ATmega328 (**fig. 1**) jusqu'aux SAMD21/51, RISC-V, SparkFun Artemis et plus encore. Vous pourrez également utiliser les cartes Thing Plus compatibles *feather* pour rester connecté au monde merveilleux de l'IdO, avec les puces ESP32 WROOM et nRF52840, ainsi que les outils de développement pour FPGA et apprentissage machine.

Périphériques

Les périphériques Qwiic peuvent paraître compliqués à classer, mais pas d'inquiétude, ils sont faciles à repérer. Un périphérique Qwiic se définit par tout ce qui se connecte à une plateforme de développement comme les shields pour Arduino (**fig. 2**), les HAT ou pHAT pour Raspberry Pi (**fig. 3**), les cartes de support pour micro:mod, etc. Ces cartes sont idéales pour équiper une carte de développement qui n'a pas encore de capacités Qwiic, et vous éviter d'acheter un tout nouvel Arduino ou Thing Plus chaque fois que vous voulez utiliser Qwiic.

Capteurs

Parmi les capteurs Qwiic il y a évidemment de tout, depuis le capteur de mouvement ou de paramètres environnementaux jusqu'aux cartes GPS et autres (**fig. 4**). Avec les capteurs Qwiic, vous pourrez enchaîner un jeucomplet de cartes d'imagerie et obtenir un spectre plus large que la vision humaine avec les IMU - parfait pour les projets de robotique et de drones.

Accessoires

Les accessoires Qwiic sont une sorte de fourre-tout de tout ce qu'il faut pour afficher des informations, alimenter en énergie et ajouter des options d'E/S aux commandes d'un projet (**fig. 5**). Le choix de cartes dans cette catégorie est vaste : déclencheur MP3, kits de relais statiques à fort ampérage, afficheurs OLED transparents (ou opaques) et bien d'autres.

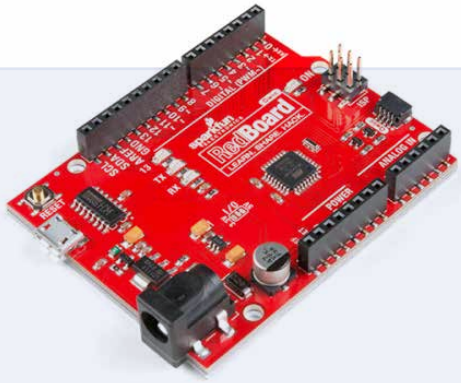


Figure 1. Au facteur de forme familier de l'Arduino R3, RedBeard est compatible Qwiic.

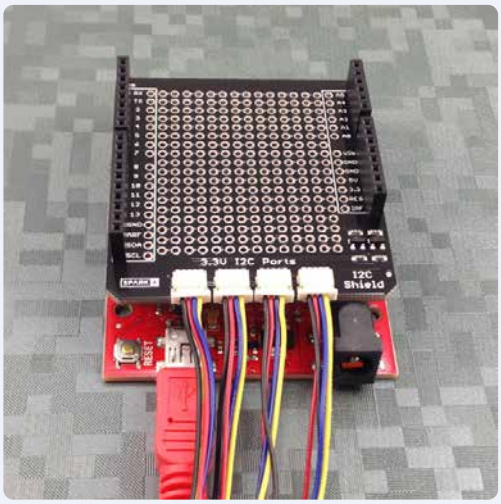


Figure 2. Shield Qwiic de style „Uno".

Figure 3. pHat v.3,0 pour Raspberry Pi.

Figure 4. Capteurs avancés sur Qwicc avec le BoB du ZED-F9, module GPS-RTK de haute précision.

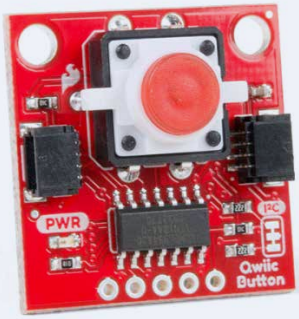


Figure 5. Ajout d'une fonction d'E/S : Carte à bouton-poussoir compatible Qwiicc.

Kits

Nous avons très vite compris qu'un choix trop vaste pourrait devenir un inconvénient de Qwiic, surtout pour les gensqui ont besoin d'un tremplin pour démarrer. D'où l'idée des kits Qwiic. Du développement à la robotique ou au Raspberry Pi - peu importe ce dont vous avez besoin, il existe un kit Qwiic pour vous aider à démarrer, à élargir le choix ou à faire progresser un de vos projets en cours (**fig. 6**).

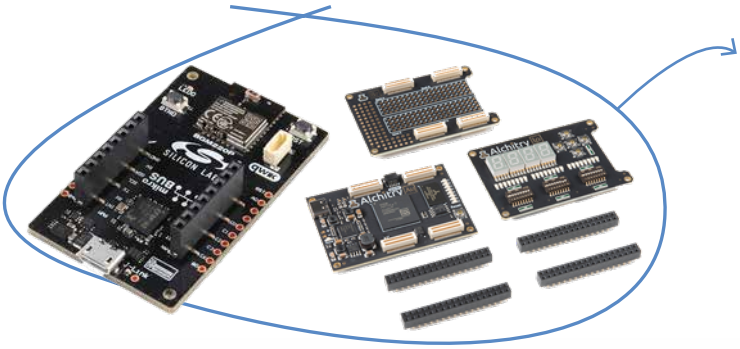
Câbles

Encore une catégorie simple et complète, avec un vaste choix de longueur de câble (**fig. 7**), ainsi que des adaptateurs pour relier le Qwiic à d'autres types de connexion. De plus, cette catégorie comprend les connecteurs qui font du Qwiic ce qu'il est. Donc, si vous voulez créer votre propre carte compatible Qwiic, c'est la catégorie qu'il vous faut !

Perspectives

Nous sommes fiers de notre écosystème. C'est un moyen simple et rapide d'ajouter des options I²C au circuit de votre choix. Trois ans après son lancement, adopté par des électroniciens de plus en plus nombreux dans le monde entier, l'écosystème Qwiic prospère et nous sommes impatients de voir comment il va progresser.

(200691 – VF : Denis LAFOURCADE)



Accessoires

Vous trouverez chez SparkFun et Elektor les accessoires mentionnés dans cet article !

www.elektormagazine.com/esfe-en-qwiic1
Kit de démarrage Qwiic pour Raspberry Pi (Figure 6)

www.elektormagazine.com/esfe-en-qwiic2
Connecteurs Qwiic chez nos partenaires (Figure 7)

Connecteurs Qwiic chez nos partenaires

Depuis l'apparition de l'écosystème Qwiic, des entreprises du secteur l'ont bien soutenu. Citons le kit d'exploration BGM220 de Silicon Labs, qui utilise leur périphérie mikroBUS, la ligne de produits FPGA d'Alchitry, la ligne complète de capteurs Zio de Hong Kong, ou encore le nouveau kit d'évaluation RA6M4 de Renesas Electronics : chacun de ces fabricants produit des cartes maintenant toutes équipées de connecteurs Qwiic. De plus en plus de confrèresadoptent dans l'industrie ce standard de connexion I2C facile, en particulier nos amis de QuickLogic et de Circuit Dojo qui proposeront bientôt des produits en coopération avec nous ! Il est gratifiant de voir les collègues incorporer Qwiic à leurs innovations.

Figure 6. Kit de démarrage Qwiic.

Figure 7. Kit de câbles Qwiic.

sparkfun
START SOMETHING

Qwiic®

Jamais il n'a été aussi facile de se brancher avec l'I²C

L' **écosystème** Qwiic Connect de SparkFun utilise des connecteurs **JST à 4 broches** pour l'interfaçage instantané de vos cartes de développement avec des capteurs, des afficheurs LCD, des relais et autres.

Le prototypage est devenu un jeu (de cartes) !

Cette gamme en constante évolution vous permet d'interconnecter rapidement cartes, capteurs, accessoires et bien davantage. Ajoutez ou modifiez vos composants en fonction des besoins de votre projet. Insérez votre nouvel élément et c'est reparti !

Plus d'infos, de tutoriels et de sources ici

www.elektormagazine.com/qwiic

elektormagazine.com/posters

elektor
design > share > sell

Sous le capot : Superchargeur & booster pour LiPo en kit

Alex Wende (États-Unis)

Savoir souder est une des compétences essentielles que tout amateur d'électronique doit acquérir très tôt. Pour passer de l'apprentissage de la construction d'un circuit sur une plaque d'essais à la conception de votre propre circuit imprimé, il y a les kits à souder. Leur objectif étant de permettre d'apprendre à souder, ils sont souvent assez rudimentaires : un clignotant avec quelques LED ou un bruiteur quelconque.
Il existe aussi des kits qui permettent de construire quelque chose d'utile, comme celui que l'électronicien youtubeur GreatScott! a conçu en collaboration avec Elektor. Je l'ai testé pour vous.

Contrairement à d'autres kits, celui du *LiPo Supercharger* permet d'apprendre à souder les composants montés en surface (CMS). Une fois assemblé, c'est une carte qui convertit l'énergie d'une seule batterie lithium-ion ou polymère, dont la tension est normalement d'environ 3,7 V, et la porte soit à 5 V avec un courant jusqu'à 1,52 A, soit à 12 V avec un courant jusqu'à 0,76 A.
Ce n'est pas tout. Comme la batterie devra être rechargée à terme, ce module est également un circuit de charge pour la batterie incorporée, avec un courant maximum de 1 A. La charge se fait soit par un connecteur USB-C, soit par une alimentation reliée aux bornes de 5 V et de masse. Ce kit est doté d'un circuit de protection de la batterie contre les inversions de tension, d'un verrouillage en cas de sous-tension et de surtension, ainsi qu'une protection contre les courts-circuits à la sortie de la carte.

Déballage et premières impressions

Le kit ne contient pas les outils nécessaires (soudure, fer à souder et pince), mais il y a tout le reste. Le manuel est accessible à partir d'une URL ou d'un code QR, ainsi que le dessin du circuit imprimé et le plan d'implantation des composants. La carte est livrée avec quelques composants déjà soudés, tous les circuits intégrés, et même une petite carte de liaison pour le connecteur USB (**fig. 1**), soudé à la carte principale par des pastilles beaucoup plus grandes sur le bord. La valeur des résistances est imprimée en clair sur ces composants, mais ce n'est pas le cas des condensateurs et des LED, Ceux-ci sont donc fournis dans des sachets individuels, étiquetés avec leur valeur, ce qui facilitera l'implantation.
Les CMS ne sont pas bien grands, notamment les résistances et les condensateurs, ce qui peut intimider un débutant. Cependant, ils ne sont pas tellement plus petits que les résistances ¼ W courantes dans les kits à composants traversants (TH). Cette taille est idéale pour commencer à se frotter à la soudure de CMS (**fig. 2**).
Le manuel commence très bien en expliquant le fonctionnement de la carte et le rôle de chacun des principaux composants. Pour qui soude pour la première fois des CMS, il explique la lecture des codes numériques sur les résistances qui indiquent où chaque composant doit être placé. Après avoir passé en revue les outils requis, il fournit quelques conseils illustrés sur l'ordre de soudage des composants. GreatScott! complète ses explications dans une vidéo [1] qui donne non seulement un aperçu détaillé de la carte, mais aussi un tutoriel de soudure de CMS (**fig. 3**).

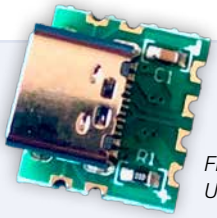


Figure 1. Carte de liaison USB-C incluse dans le kit.

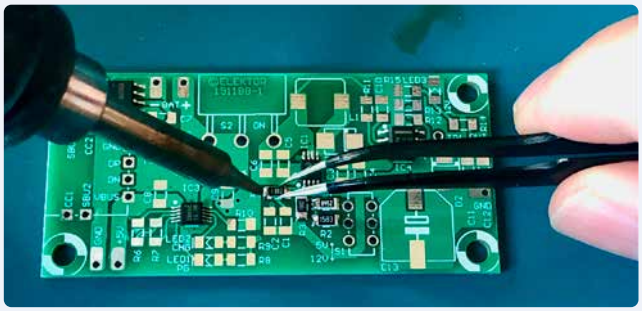


Figure 2. Soudage des résistances.

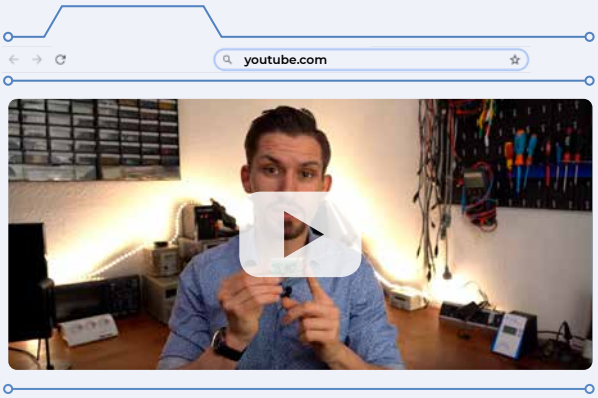


Figure 3. GreatScott! présente son kit de SuperCharger LiPo sur YouTube.

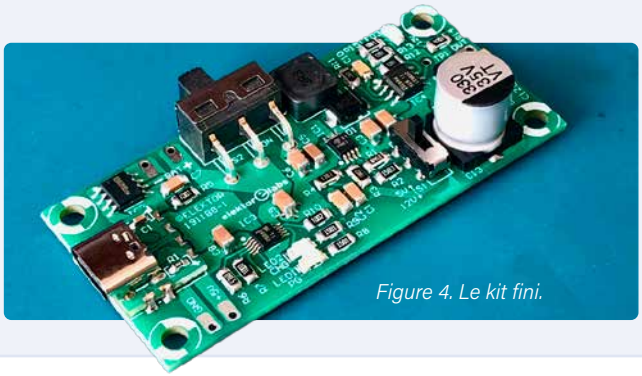


Figure 4. Le kit fini.

Tester le produit final

Une fois le kit assemblé (**fig. 4**), j'ai branché le chargeur pour m'assurer d'avoir une batterie chargée. Aussitôt, j'ai vu s'allumer les voyants d'alimentation et de charge. Une fois la batterie chargée à bloc, il était temps de mettre la carte à l'épreuve.

La première étape a consisté à m'assurer que j'ai soudé comme il faut. Heureusement, mon multimètre m'a indiqué une tension assez proche de 5 V et 12 V. Les alimentations à découpage sont appréciées pour leur petite taille et leur rendement, mais, si elles ne sont mal conçues, elles ont pour effet secondaire d'être assez bruyantes en sortie. Après une série de tests, j'ai été heureux de constater que le bruit produit restait inférieur à 40 $mV_{c.àc.}$ sous une lourde charge,mais encore beaucoup plus faible la plupart du temps.

Quel que soit le type de régulation de tension, le grand ennemi, malheureusement inévitable, est la dissipation thermique. J'ai été surpris par le rendement de la carte, d'autant plus qu'elle augmente la tension de la batterie au lieu de la diminuer. Sous de lourdes charges, elle chauffe un peu cependant. Dans une charge continue, le courant maximum mesuré sous 5 V était d'environ 1,3 A, et sous 12 V, de 0,4 A. Par salves, l'intensité du courant de sortie atteint facilement 1,52 A et 0,76 A.

Réflexions générales

Pour le débutant, le concepteur de la carte a fait un excellent travail. Les CMS les moins faciles à trouver sont déjà soudés sur la carte. La taille des composants à souder est suffisante même pour « la première fois ». La description du kit ne laisse planer aucun doute. Il y a quelques douzaines de composants à souder, mais, grâce à la nomenclature interactive "cliquez et localisez" sur le site web d'Elektor [2], je n'ai pas eu à scruter longuement la carte pour y trouver p. ex. "R7" ou "C3".

Ce que j'ai le plus apprécié dans ce kit, c'est qu'une fois soudé, je vais l'utiliser. Les projets alimentés par piles qui ont besoin d'une source d'énergie régulée ne manquent pas. Et pour un circuit comme une chaîne de LED, ou avec quelques petits moteurs, il est pratique de pouvoir augmenter la tension de la batterie à 5 V ou 12 V. Avec son circuit de protection, son circuit de charge et son interrupteur bien pensés, je sais que n'ai aucun souci à me faire pour ce super-chargeur et pourrai donc me concentrer sur les circuits qu'il alimente.

(200693-03 VF : Délenda Carthago)

Accessoires

Vous trouverez chez Elektor les accessoires décrits dans cet article !

- **Kit de super-chargeur LiPo à assembler soi-même (par GreatScott !)**
  www.elektor.com/diy-lipo-supercharger-kit-by-greatscott

LIENS

[1] Vidéo du kit SuperCharger LiPo : https://youtu.be/6LxRnf6sQNQ
[2] Article sur le kit LiPo SuperCharger et BoM : https://www.elektormagazine.fr/articles/superchargeur-lipo-booster-en-kit

Électronique mémorable du passé de SparkFun

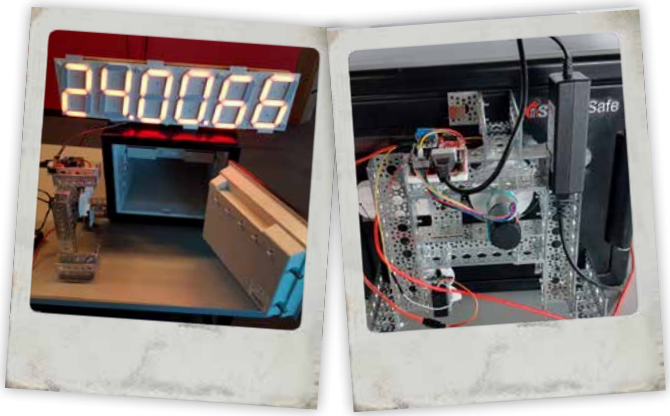
l'équipe de SparkFun

Même à la pointe du progrès, on a ses moments de nostalgie. Dans Elektor il y a même une rubrique pour ça. Depuis 2003, avec sa culture décalée, SparkFun a enchaîné les produits fous et les projets audacieux. Pour cet épisode de *Rétronique*, les vétérans de SparkFun parlent de ce qui leur vient à l'esprit pour évoquer le riche passé de SparkFun.

2005 : Le fameux téléphone Port-O-Rotary

Note de Nathan Seidle (fondateur) datée du 17 janvier 2005 : „Oui, tu as bien lu, Port-O-Rotary. On a piraté un téléphone à cadran rotatif. Il sonne, mais il est sans fil. Imagine la tête des gens quand ils voient et entendent ça !"

Ce projet déjanté associait un cadran téléphonique rotatif de près de 50 ans, aux techniques les plus en vue en 2005 pour faire un téléphone cellulaire, basé sur module quadribande GM862 de Telit, une carte à PIC (18 broches) modifiée, avec le 16F88, une antenne cellulaire tribande, une batterie lithium-ion polymère de 3,7 V, répondant à des commandes AT simples. Ce projet fou a eu un tel succès que nous en avons fait un produit… qui s'est vendu !



2017 : Robot casseur de coffre-fort

La compagne de Nate lui a un jour offert un vieux coffre-fort dont elle ignorait le contenu, mais qu'elle n'avait pas payé cher car personne n'en connaissait la combinaison. Ce fut la rencontre entre une technique antique de protection et SparkX, notre équipe expérimentale, qui a employé une nouvelle technique pour forcer le coffre sans intervention humaine en… 40 min et 42 s avec un robot autonome fabriqué avec le RedBoard, des servomoteurs, une commande de moteur, un capteur de courant, un buzzer, un châsssis fait de pièces Actobotics et un bouton rouge „go" (tutoriel complet [1]). Le coffre-fort était vide, mais cela n'avait pas d'importance.

Quelques mois après, l'équipe se rendait au DEF CON 25 pour renouveler l'expérience sur scène devant environ 800 casseurs informaticiens. On a débarqué, acheté un coffre-fort sur place (impossible de voyager avec l'ancien), on l'a déballé sur scène sans avoir la moindre idée de son fonctionnement. Moins de 25 min plus tard, le coffre-fort était ouvert ! Imaginez l'ambiance dans la salle pleine de *hackers* silencieux pendant que le robot était à la manœuvre ?

2009-2018 : Dix ans de concours de véhicules autonomes

Le concours est né d'un pari sur la construction d'un véhicule capable de se déplacer dans les locaux de SparkFun. En 2009, cette tâche n'était pas des plus faciles. La communauté SparkFun a relevé le défi. La première année, ce sont 16 véhicules qui ont participé. Les drones de Chris Anderson ont gagné en bouclant le parcours en 36 s ; au moins un des drones a fini dans un très grand arbre (oui, il y avait des engins volants) ; l'expérience a été inoubliable. C'était la première fois qu'un tel drone était utilisé, l'événement était hautement expérimental. Nul ne savait à quoi s'attendre. Tous ont beaucoup appris.

En 2018, la finale de l'AVC a réuni près de 200 équipes, soit près de 500 concurrents ! Il y avait à la fois le parcours des véhicules autonomes et l'arène des robots de combat (*battle-bots*). Au fil des ans, l'expertise des concurrents a progressé rapidement, de même que l'accès aux prodiges techniques, le LIDAR surtout et d'autres capteurs. À tel point que SparkFun a dû relever le niveau de difficulté du parcours, avec des obstacles mobiles, des rampes, des épingles à cheveux, et d'autres complications que nos ingénieurs ont dû imaginer.

En 2018, face à la prolifération de compétitions similaires, s'est imposée la décision de mettre fin à cette aventure de dix ans, inscrite maintenant dans l'histoire de SparkFun.

Sélection de projets légendaires, farces et bizarreries

2006 : Commande d'inclinaison à distance Roomba®. Quand iRobot® a ouvert sa plateforme Roomba au bidouillage, SparkFun a proposé quelques outils de développement Roomba et une commande d'inclinaison. Celle-ci, appelée RooTilt, amalgame divers jouets de SparkFun. Nous avons pris un accéléromètre sans fil Bluetooth, modifié un peu le micrologiciel et l'avons combiné avec la liaison sans fil RoboDynamics RooTooth pour le Roomba. Et voilà ! Pilotez votre Roomba.

2006 : Horloge murale GPS de 30 cm. C'est le projet qui n'a cessé de prendre de l'ampleur. Nous avons combiné quelques barres lumineuses à LED avec un simple contrôleur et un récepteur GPS pour en faire une très grande horloge qui se règle toute seule, affiche les heures/minutes/secondes, et est précise à 100 ns.

2007 : Contrôleur NES géant. Oui ! Nous nous sommes présentés à la Maker Faire avec une manette Nintendo fonctionnelle de 40 kg et 1,5 m.

2010 : Concours Antimov. Basé sur les trois lois de la robotique d'Isaac Asimov, le concours Antimov mettait les participants au défi de concevoir un robot qui primo enfreint les lois de la robotique (sauf blesser des êtres vivants), deuxio accomplit une tâche triviale de la manière la plus inefficace et la plus laborieuse possible et troisio se détruit lui-même en accomplissant cette tâche. Chose étonnante, nous avons eu des participants…

2012-2013 : Tournée nationale. La tournée nationale de SparkFun avait pour objectif de partager notre passion pour l'électronique avec les élèves et les enseignants de tout le pays. À bord du véhicule récréatif SparkFun, l'équipe a parcouru le pays pendant un an, faisant étape partout où nous devions animer des ateliers d'électronique.

2013 : Plongée inaugurale dans les poubelles. Entre les prototypes, les échantillons, les retours de clients et tous les vieux trucs restés trop longtemps sur les étagères, nous accumulons les rebuts. En 2013, quelqu'un a eu une idée géniale : puisque la plupart sont encore utilisables, ne les recyclons pas, remettons-les entre les mains des gens. Ainsi naquit le Dumpster Dive Sale ou les soldes-poubelles. Vous achetez des pochettes surprises remplies d'électronique dont vous ignorez tout. Il faut les déballer pour en découvrir le contenu. On en fait encore de temps en temps, lorsque nos surplus débordent et qu'on se souvient qu'ils pourraient faire la joie d'autrui !

2015 : Capteur de vitesse SparkFun. Avec cet afficheur à chiffres géants, le RedBoard et le LIDAR, ce radar de vitesse est installé dans notre quartier général. Les gens (surtout les visiteurs) sont curieux de leur vitesse et courrent donc vers lui… souvent si vite qu'ils n'arrivent pas à s'arrêter avant le mur. Dangereux mais très amusant.

2017 Skimmer Scammer App. L'équipe de SparkX a coopéré avec des enquêteurs pour effectuer l'ingénierie inverse de matériel utilisé pour pirater les cartes de crédit dans les pompes à essence. Ce matériel, découvert dans certaines pompes au Colorado, mais utilisé par des fraudeurs dans tout le pays, pouvait être détecté par Bluetooth. L'équipe de SparkX a écrit une application gratuite pour alerter les gens lorsqu'un tel pirate de carte de crédit était détecté à une pompe à essence. L'application a été téléchargée plus de 200 000 fois. Nous l'avons retirée après son inactivation par une mise à jour d'Android en 2019.

2018 : Joystick géant. Le lancement par Microsoft de la manette adaptative Xbox nous a stupéfiés et enthousiasmés. Quel meilleur moyen qu'un banc d'essai technique et un projet génial pour faire passer le mot à notre communauté ? Puisque Microsoft a permis au contrôleur de fonctionner avec des accessoires tiers, l'équipe de SparkFun a construit un gigantesque joystick inspiré librement des anciens joysticks Atari. Ce projet se distinguait par sa créativité, ses boutons big-dome et quelques micro-interrupteurs.

(200695 — VF Tobé Gotokal)

LIENS

[1] Tutoriel du robot forceur de coffre-fort : https://learn.sparkfun.com/tutorials/building-a-safe-cracking-robot

Stationnement parfait **avec LiDAR**

Rob Reynolds (États-Unis)

Quoi de plus génial que les capteurs de distance ? Ils sont souvent le premier pas dans la réalisation d'un robot autonome, mais servent aussi bien dans d'autres applications divertissantes, éducatives ou utilitaires. Quand j'ai eu l'idée d'un système d'assistance pour garer ma voiture, j'ai déménagé le projet avec ses capteurs de distance de ma paillasse dans le garage et suis parti pour de nouvelles bidouilles !

D'abord, inventer un besoin

Je n'ai jamais eu ni balle de tennis, ni planche de surf ou ni aucun autre repère accroché au plafond de mon garage pour m'indiquer l'endroit précis où arrêter ma voiture lorsque je la rentre au garage. Maintenant que j'ai un adolescent d'âge à se mettre au volant, il est temps d'y songer. Comme je ne joue pas au tennis, mais à l'électronique, c'est d'elle que viendra donc la solution. Le plus simple serait de suspendre un RPi au plafond du garage à la place de la balle de tennis, mais je veux utiliser un capteur de distance associé à un petit feu tricolore. À l'entrée du garage, la voiture est accueillie par une LED verte ; lorsqu'elle s'avance, le feu passe au jaune puis au rouge lorsqu'elle atteint le point d'arrêt idéal. Ce feu tricolore, je vais le concevoir et l'imprimer en 3D. Ça fera un coffret sympa !

Quel est le capteur adéquat ?

Avec cette profusion de capteurs de distance/proximité (**fig. 1**), comment savoir lequel convient le mieux à votre projet ? Il y a un bon nombre de caractéristiques à évaluer, dont la portée, la résolution, le type d'interface, la fréquence de mesure et le coût. Il y en a qui ne sont pas critiques (faut-il vraiment 635 mesures par seconde ?) et d'autres qui sont incontournables (mon professeur affirme qu'il faut utiliser des composants I²C). Entre porte et mur du fond, mon garage fait environ 8,6 m . Je pourrais probablement m'en tirer avec l'un de nos appareils à ultrasons XL-MaxSonar, d'une portée d'environ 8,3 m, mais, pour cette réalisation, je vais utiliser le module TFMini - Micro LiDAR [1]. Je l'utilise avec le kit Qwiic, car il comprend une carte amplificatrice. Comme le TFMini fonctionne sous 5 V mais communique sous 3,3 V, la carte amplificatrice fournie avec le kit Qwiic élimine le besoin de conversion de niveau logique si l'on utilise une carte 5 V, ou d'une alimentation bi-tension si l'on utilise une carte 3,3 V. En l'associant au Redboard, avec son connecteur Qwiic intégré, je peux faire de la moitié capteur de cet assemblage un dispositif *brancher-jouer* simple (**fig. 2**). Ajoutez quelques LED à forte luminosité, des résistances et une alimentation, et le tour sera joué !

Qwiic = assemblage rapide et facile des composants I²C

Le système Qwiic simplifie cette réalisation. La version TFMini Qwiic est livrée avec la carte amplificatrice et une paire de câbles. Un câble relie le module à la carte amplificatrice, l'autre la carte amplificatrice

au connecteur Qwiic de votre RedBoard. Les LED verte, jaune et rouge sont sur les broches 8, 9 et 10. Pour le code, j'ai légèrement retouché le croquis *LidarTest.ino* disponible sur la page du *Guide de branchement du TFMini Qwiic* [2]. Le **listage 1** montre le programme retouché. J'ai également réalisé un modèle vite fait de feu tricolore que vous pouvez télécharger, ainsi que le croquis Arduino, depuis mon dépôt Github. [3]

JE VOUS AURAI PRÉVENU ! En raison de la consommation du TFMini et du système de gestion de l'énergie du RedBoard, si vous essayez d'alimenter votre projet final par la prise jack, il se figera, la LED verte restera allumée et votre chauffeur – trompé par sa confiance dans votre assistant – foncera droit dans le mur. Alors qu'avec l'alimentation par le connecteur microUSB, ce problème ne se pose pas. Oubliez donc cette prise jack si vous ne voulez pas d'ennuis !

À vous de jouer !

Ma réalisation est simple, voire expéditive, vous pourrez donc faire autrement et mieux. Lancez-vous avec le code et les fichiers .STL [3] ! Quelques suggestions :

- ajouter des boutons pour faciliter le réglage du point d'arrêt optimal ;
- améliorer le coffret ;
- augmenter le nombre de LED pour accroître la visibilité.

Avec de l'ambition, vous pouvez même remplacer le RedBoard par un RPi et programmer ce projet en Python, en ajoutant un écran pour plus d'interaction visuelle.
Y a-t-il un avantage à utiliser un ordinateur mono-carte plutôt qu'un microcontrôleur pour ce projet ? Absolument aucun. Mais pourquoi faire simple quand on peut faire compliqué ? Alors, au travail, faites-vous plaisir, les amis !

(200698 VF Helmut Müller)

[suite au verso]

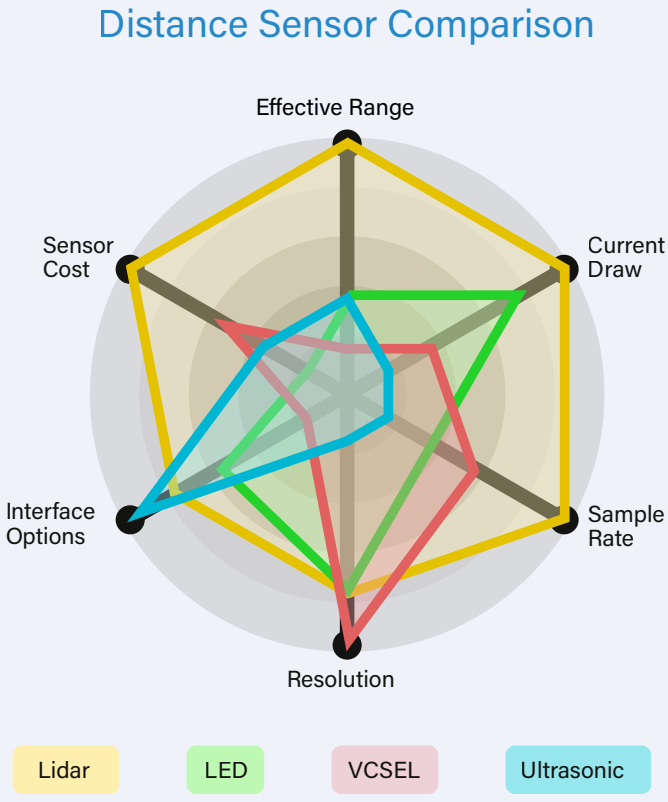


Figure 1. Pour choisir le bon capteur de distance pour votre application, évaluez les performances des technologies disponibles par rapport à vos objectifs de conception.

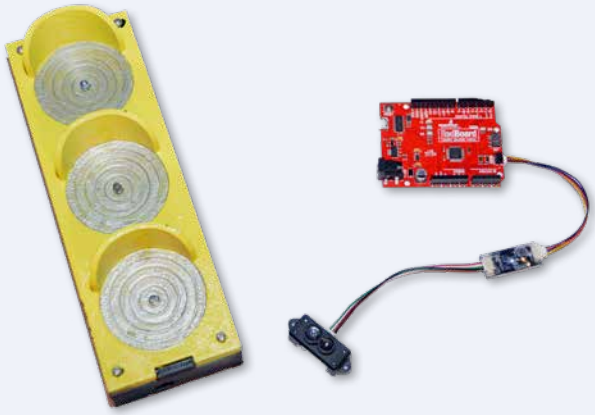


Figure 2. Les composants électroniques et le coffret imprimé en 3D de ce projet

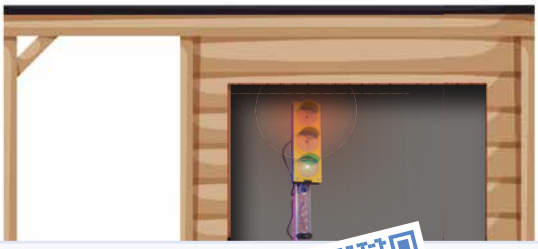
Listage 1. Programme Perfect Parking with LiDar

```
/*
  TFMiniStopLight.ino
  Rob Reynolds, November 19, 2018

  This code is a small practical demonstration
application of the TFMini Lidar Module. The 3D
files can be found in the Github repository, here
  [ https://github.com/ThingsRobMade/TFMini_Stop_
Light ]
  Based heavily on the previous collaborative work
done by Nate Seidle and Benewake. The original
example sketch for the Qwiic Enabled TFMini can be
found here:
  (https://www.sparkfun.com/products/14786)

  This code is free, but if you find it useful,
and we meet someday, you can buy me a beer
(Beerware license).
*/

#include <Wire.h>

uint16_t distance = 0; //distance
uint16_t strength = 0; //signal strength
uint8_t rangeType = 0; //range scale
/*Value range:
  00 (short distance)
  03 (intermediate distance)
  07 (long distance) */

boolean valid_data = false;
//ignore invalid ranging data

const byte sensor1 = 0x10;
//TFMini I2C Address

// Define pins for LEDs
const int greenLED = 8;
const int yellowLED = 9;
const int redLED = 10;
int stopLimit = 160; //Change this number of cm to
adjust stop distance

void setup()
{
  Wire.begin();

  Serial.begin(115200);
  Serial.println("TFMini I2C Test");
  //For testing using Serial Monitor

  // Set LED pins as outputs
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}

void loop()
{
  if (readDistance(sensor1) == true)
  {
    if (valid_data == true) {
      Serial.print("\stopLimit[");
/* These Serial.print lines remain for testing and
adjustment purposes */
      Serial.print(stopLimit);
      Serial.print("\tDist[");
      Serial.print(distance);
      Serial.println();

      if (distance <= stopLimit) {
        digitalWrite(redLED, HIGH);
        digitalWrite(yellowLED, LOW);
        digitalWrite(greenLED, LOW);
      }
      else if (distance > stopLimit && distance
< (stopLimit + 200) ) { //change this number to
increase distance yellow stays lit
        digitalWrite(redLED, LOW);
        digitalWrite(yellowLED, HIGH);
        digitalWrite(greenLED, LOW);
      }
      else if (distance > (stopLimit + 199) ) {
//change this number to adjust when yellow LED
illuminates
        digitalWrite(redLED, LOW);
        digitalWrite(yellowLED, LOW);
        digitalWrite(greenLED, HIGH);
      }
    }

    //  else {
    //   Serial.println("Read fail");
    //  }

    delay(50);
//Delay small amount between readings
  }
}


//Write two bytes to a spot
boolean readDistance(uint8_t deviceAddress)
{
```

Et si on remplaçait le RedBoard par un RPi pour programmer ce projet en Python ?

Rob Reynolds

```
  Wire.beginTransmission(deviceAddress);
  Wire.write(0x01); //MSB
  Wire.write(0x02); //LSB
  Wire.write(7); //Data length: 7 bytes for
distance data
  if (Wire.endTransmission(false) != 0) {
    return (false); //Sensor did not ACK
  }
  Wire.requestFrom(deviceAddress, (uint8_t)7); //
Ask for 7 bytes

  if (Wire.available())
  {
    for (uint8_t x = 0 ; x < 7 ; x++)
    {
      uint8_t incoming = Wire.read();

      if (x == 0)
      {
        //Trigger done
        if (incoming == 0x00)
        {
          //Serial.print("Data not valid: ");
          //for debugging
          valid_data = false;
          //return(false);
        }
        else if (incoming == 0x01)
        {
          Serial.print("Data valid:     ");
          valid_data = true;
        }
      }
      else if (x == 2)
        distance = incoming;
//LSB of the distance value "Dist_L"
      else if (x == 3)
        distance |= incoming << 8; //MSB of the
distance value "Dist_H"
      else if (x == 4)
        strength = incoming; //LSB of signal
strength value
      else if (x == 5)
        strength |= incoming << 8; //MSB of signal
strength value
      else if (x == 6)
        rangeType = incoming; //range scale
    }
  }
  else
  {
    Serial.println("No wire data avail");
    return (false);
  }

  return (true);
}
```



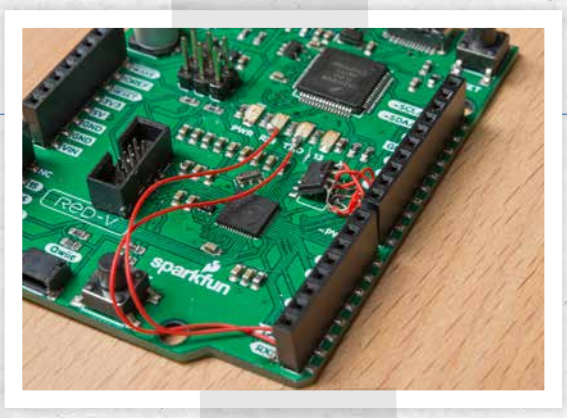
Accessoires

Vous trouverez chez Elektor et SparkFun les principaux articles mentionnés dans cet article.

- **RedBoard - Programmé avec Arduino**
  www.elektormagazine.com/esfe-en-perfectparking1
- **TFMini - Micro Module LiDAR**
  www.elektormagazine.com/esfe-en-perfectparking2

LIENS

[1] TFMini - Module Micro LiDAR : https://www.sparkfun.com/products/14588
[2] Guide de connexion de TFMini Qwiic : https://bit.ly/2Xzun4r
[3] Sketch de Perfect Parking, fichiers .STL du coffret : https://github.com/ThingsRobMade/TFMini_Stop_Light



Trépanation d'un circuit imprimé à pastilles inaccessibles

Nathan Seidle (États-Unis)

De nombreuses entreprises ne reconnaissent jamais leurs erreurs, leur vie est trop courte. Ici, nous ne nous laissons pas abattre par nos échecs, nous les clouons au mur et nous en parlons à tout le monde pour éviter de refaire nos erreurs.

« Tout le monde peut dessiner un circuit imprimé, mais pour en faire un bon, il faut de solides compétences ». C'est avec de telles maximes que je me trouve des excuses quand j'en rate un. Dans notre labo SparkX, nous avons un coin de mur sur lequel nous affichons nos ratés (**fig. 1a**) qui racontent chacun une histoire. Bien des entreprises ne reconnaissent jamais leurs erreurs, leur vie est trop courte. Ici, nous ne nous laissons pas abattre par nos échecs. Nous les clouons au mur et en parlons à tout le monde, cela nous évite de les refaire.

Le circuit fonctionne-t-il ?

Je conçois des cartes (et j'en rate) depuis près de deux décennies. J'ai appris de nombreuses astuces, mais la leçon majeure est de m'assurer que, avant de lancer un nouveau lot de cartes et quel que soit le nombre de pistes à couper ou à rajouter, le circuit en cours d'étude fonctionne

Figure 1a: Mur des ratés dans la zone SparkX.

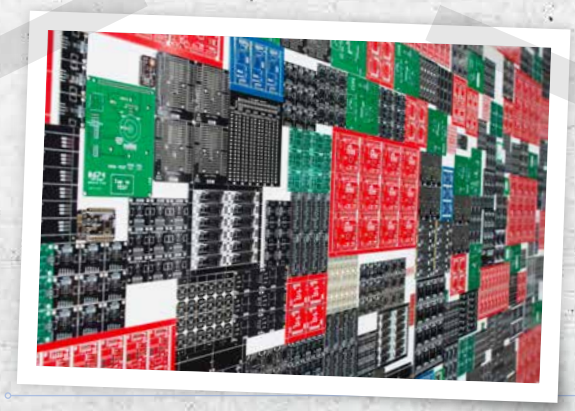


Figure 1b: Admirez quelques-uns de ces mémorables ratés.

Figure 2 : Comment rajouter une connexion sur cette pastille ?

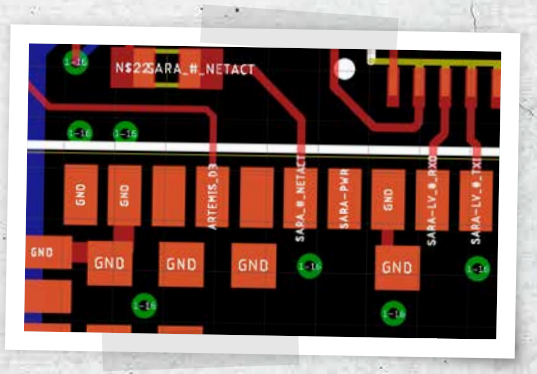


Figure 3 : La pastille SARA-PWR oubliée.

à 100 %. Si vous trouvez une erreur et que vous commandez aussitôt de nouvelles cartes, eh bien, vous ne vous serez débarrassé que cette erreur-là. Sur un projet récent, j'ai dû faire une intervention chirurgicale inédite pour moi à un point tel que je voudrais la partager (**fig. 2**). Le circuit à l'étude était une simple combinaison d'Artemis et de notre module cellulaire LTE. Comme pour la plupart des protos, j'ai commencé avec un RedBoard Artemis et le bouclier LTE pour m'assurer que tout fonctionnerait. J'ai ensuite dessiné une carte et fait réaliser quelques prototypes. Lors du test des différentes parties de la carte, tout a fonctionné, sauf le cellulaire. Je ne manque pas d'excuses (trop de projets en même temps, manque de temps, etc.), mais j'avais tout bêtement oublié de relier la broche d'alimentation du module SARA à Artemis (**fig. 3**).

Or, le module cellulaire SARA LTE NB-IoT (**fig. 4**) offre un incroyable choix de fonctions et, comme de nombreux modules cellulaires, pour qu'il démarre, sa broche d'alimentation doit être maintenue au niveau bas pendant deux ou trois secondes. Il ne restait donc qu'à rajouter cette connexion oubliée. Habituellement les pattes d'un circuit intégré sont accessibles, il est donc facile de souder un fil sur un SOIC ou un TSSOP. Avec un peu de colle (pour maintenir le fil en place), du flux, du fil de câblage et un peu de tresse à dessouder, c'est possible aussi sur un QFN. Les pastilles du module SARA sont disposées sous le module et donc inaccessibles. Heureusement, comparées à celles d'un QFN, elles sont assez grandes. Que faire ? Corriger le dessin du PCB et en faire graver un nouveau lot, ou tenter d'y percer un trou par l'autre face, en m'arrêtant juste au moment où la mèche atteint la surface de la carte *sous* le module SARA. J'ai opté pour cette intervention consistant à passer un fil *à travers* le circuit imprimé pour le souder sur la pastille de la broche de commande de l'alimentation sous le module SARA.

Perçage, soudage et collage

Pour localiser l'endroit exact où percer la carte par en dessous, j'ai rajouté sous le dessin du circuit un rond de 1/16" (1,6 mm) à l'endroit où je voulais faire la connexion, en prenant soin qu'il soit à cheval sur la pastille qui m'intéressait et le plan de masse à proximité, pour que le bord droit de celui-ci me serve de repère pour centrer la pastille de commande de l'alimentation (**fig. 5**).

J'ai imprimé ce dessin à l'échelle 1, découpé le rectangle et l'ai disposé sur le dos du proto. À l'aide d'un pointeau à ressort, j'ai marqué avec précision l'endroit où percer (**fig. 6**).

Figure 4 : L'avant/arrière du module Ublox SARA.

Figure 7 : Forets standard de la caisse à outils du bricoleur.

Figure 5 : Position de la pastille vue de l'arrière (ou du dessous) de la carte.

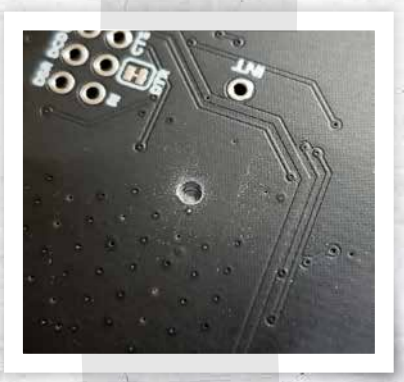


Figure 8 : Perçage initial avec un foret de 1/16".

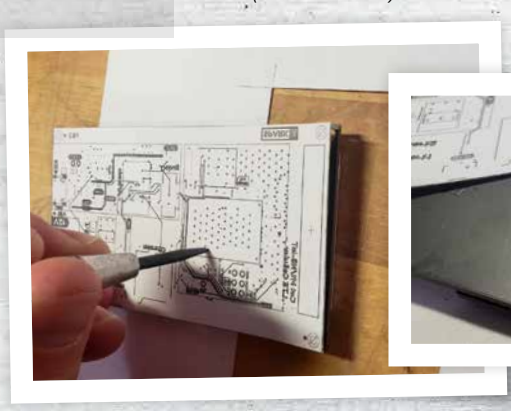


Figure 6a: Le pointeau à ressort marque l'endroit.

Figure 6b: C'est le marquage qui fait toute la différence.

Pour le premier perçage, j'ai utilisé deux tailles de foret standard : 1/16" (1,6 mm) et 3/32" (2,38 mm) (**fig. 7**) et une perceuse à colonne (**fig. 8**). J'ai traversé les couches inférieures de vernis épargne, de cuivre et de verre époxy FR4. Comme il s'agit d'une carte à deux couches, j'ai arrêté de forer à la machine juste avant d'atteindre la couche de cuivre supérieure pour finir à la main (**fig. 9**). Le FR4 se laisse travailler avec une mèche bien affûtée tenue à la main. J'ai arrêté dès que j'ai senti que le foret „accrochait" la couche de cuivre supérieure.

Je suis ensuite passé au foret 3/32" (2,38 mm) pour agrandir le trou (**fig. 10**) assez pour y passer une sonde de multimètre. J'ai gratté puis testé la continuité entre le bord du fond du trou (où j'espérais établir un contact avec le cuivre du plan de masse supérieur) et une pastille de masse à proximité. Satisfait d'avoir atteint sans dommage la couche de cuivre supérieure, du moins le grand plan de masse, j'ai repris le rognage avec le foret de 1/16" légèrement en biais.

Quand j'ai eu l'idée d'utiliser de l'alcool isopropylique avec un coton-tige pour nettoyer les rognures de fibre de verre du trou, quelle ne fut pas ma surprise de voir apparaître les pastilles (**fig. 11**) ! Le plan de masse était facile à repérer, mais laquelle des deux autres était la pastille de commande de l'alimentation qui m'intéressait ? J'ai vu sur le dessin que c'était celle de gauche (**fig. 12**). L'alcool avait rendu transparente la fibre de verre pendant un moment, mais une fois évaporé, la FR4 est redevenue opaque (**fig. 13**). Il devait donc rester de la fibre de verre au-dessus de la pastille.

Après avoir continué de forer un peu plus à la main, j'ai vu apparaître du brillant. Le trou (**fig. 14a**) était assez grand pour y passer la panne de mon fer à souder, mais ce n'était pas encore gagné. L'ajout d'un peu de flux a eu deux effets bénéfiques : excellent pour faire couler la soudure là où il faut, il contient aussi un acide léger qui nettoie les contacts oxydés. Juste ce qu'il fallait ici pour enlever assez de FR4 et créer un point soudable sur la pastille.

Après quelques tentatives, j'ai réussi à chauffer suffisamment le fil nu de 30 AWG pour faire fondre la soudure à son extrémité sur la pastille. C'est du bricolage de dingue (**fig. 14b**), mais la soudure résiste bien à une (très légère) traction. Quelques vérifications de continuité supplémentaires au multimètre m'ont rassuré sur l'absence de court-circuit avec le plan de masse.

Le sommet du trou était entièrement encerclé par le plan de masse, avec lequel le fil nu ne devait pas entrer en contact. J'ai immobilisé le fil avec du ruban isolant (**fig. 14c**) pendant que j'ai soudé son autre extrémité au microcontrôleur (l'Artemis). Après une ultime vérification et avec beaucoup de précautions de manipulation du circuit imprimé, le module cellulaire a démarré !

Inquiet de casser la soudure ou de détériorer la pastille, j'ai – après un ultime test électrique – inondé de colle chaude le trou et le fil à proximité (**fig. 14d**), apportant protection et tenue mécanique au fil et à la soudure.

Un peu de chance, ça aide

Cette méthode n'a évidemment fonctionné (**fig. 15**) que parce que j'ai eu de la chance : la taille des pastilles permettait la soudure manuelle ; il se trouvait sous les pastilles aucune piste vitale qui m'aurait empêché

Figure 9 : Fin de perçage à la main.

Figure 11 : On voit des pastilles !

Figure 10 : Pas grand-chose à voir.

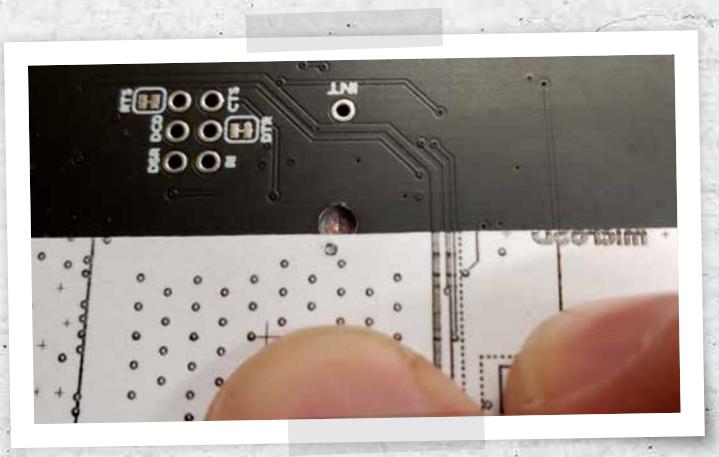


Figure 12 : La pastille d'alimentation est à gauche.

de percer par l'arrière. Une fois trouvé et réglé le problème de la broche d'alimentation, j'ai repris l'examen du circuit et j'ai trouvé d'autres bourdes. Quatre pour être précis, qui seraient passées inaperçues à ce stade si, après avoir corrigé le premier problème, je m'étais lancé aussitôt dans une nouvelle version de la carte. Les quatre autres problèmes auraient fini par se manifester, mais bien plus tard. La prochaine fois que vous travaillerez sur une carte défectueuse, prenez le temps de mener rigoureusement chaque étape de vérification à son terme.

200700-03

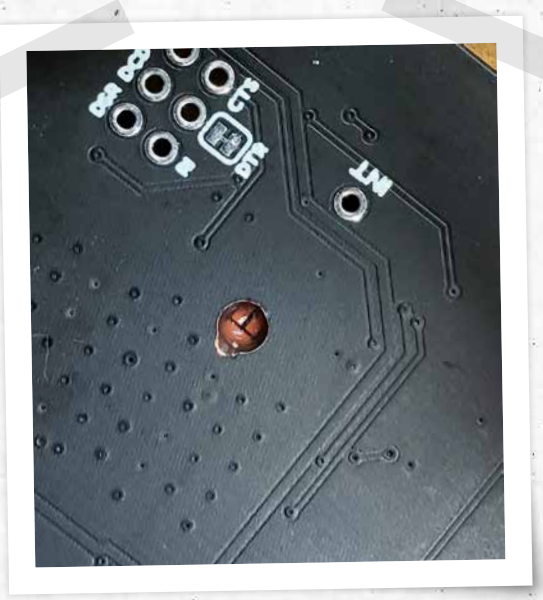


Figure 13 : L'alcool rend le FR4 transparent.

Figure 14a : Le trou est assez grand pour y passer la panne de mon fer à souder.

Figure 14b: Pas très joli.

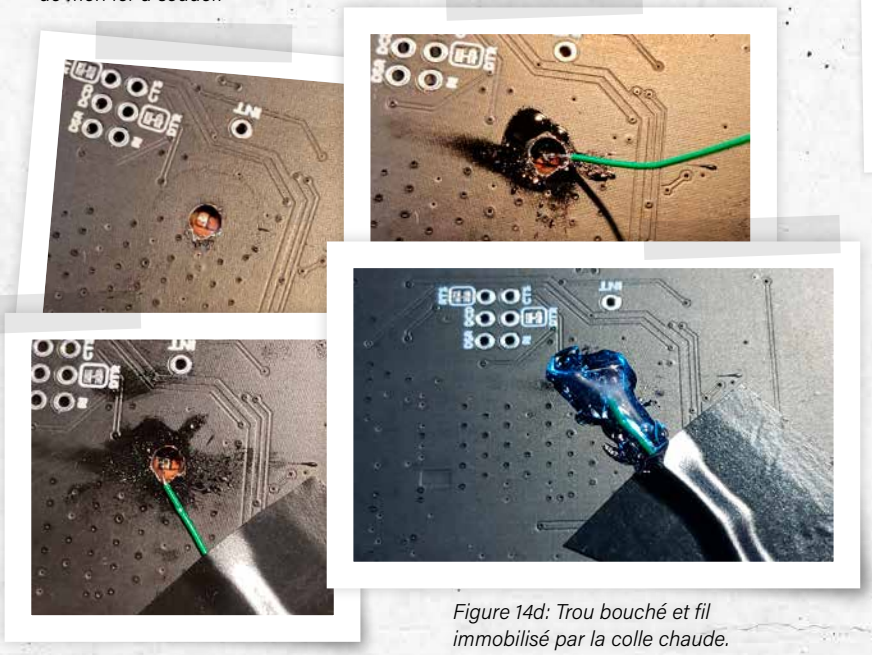


Figure 14c: Le fil a été soudé avec succès à travers le circuit imprimé.

Figure 14d: Trou bouché et fil immobilisé par la colle chaude.

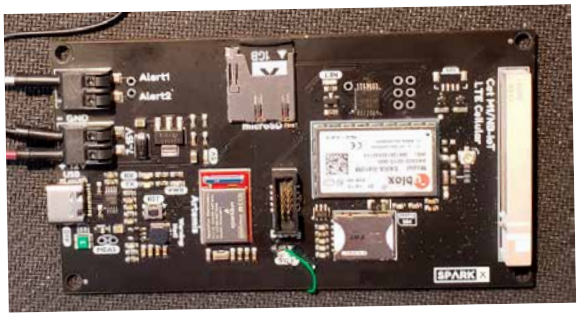


Figure 15 : Et voilà le travail !

Accessoires

Voici les principaux accessoires mentionnés dans cet article d'Elektor :

- **Module SparkFun Artemis - Cortex ML BLE de faible puissance-M4F**
  www.elektormagazine.fr/esfe-en-erroranalysis1
- **SSparkFun RedBoard Artemis**
  www.elektormagazine.fr/esfe-en-erroranalysis2
- **Bouclier SparkFun LTE CAT M1/NB-IoT - SARA-R4**
  www.elektormagazine.fr/esfe-en-erroranalysis3

Ad astra per aspera

L'ambition de la série d'articles *Analyse d'erreurs* proposée par Elektor est d'encourager les électroniciens, quel que soit leur niveau, à tirer le plus possible de leçons de leurs propres erreurs en les partageant avec toute la communauté. Ainsi, chaque membre de notre équipe reste à l'affût d'erreurs et de leçons à partager. C'est exactement cet esprit que nous avons reconnu sur le mur des ratés du labo SparkX.

Concevoir pour vendre : RTK Surveyor de SparkFun

RTK Surveyor est un récepteur GNSS utilisé pour la géolocalisation de précision. Nous vous proposons de découvrir le processus de mise sur le marché d'un tel outil de levé. En français, *surveyor* se traduit par géomètre ou topographe.

Chris McCarty (États-Unis)

RTK est le sigle de *real-time kinematic*, c'est-à-dire *cinématique en temps réel*, autrement dit l'étude des mouvements. Le RTK Surveyor est un récepteur GNSS, système mondial de navigation par satellite, facile à utiliser pour un positionnement au centimètre près. Parfait pour la topographie, cet appareil préprogrammé peut également être utilisé pour la conduite autonome, la navigation, le suivi des biens et toute autre application pour laquelle il y a une vue dégagée du ciel. Le RTK Surveyor peut également, d'un simple clic, servir de station de base, tandis que deux RTK Surveyor peuvent être utilisés conjointement pour créer un système RTK d'une *précision horizontale de 14 mm*. La connexion Bluetooth intégrée sur une ESP32 WROOM permet à l'utilisateur d'utiliser le ZED-F9P du RTK Surveyor avec l'application SIG (= système d'information géographique) de son choix sur un téléphone ou une tablette. La batterie intégrée permet une utilisation sur le terrain

31 Chronologie

15 août 2020
Le fondateur de SparkFun, Nathan Seidle, assemble le premier prototype du RTK Surveyor. À ce stade, c'est un ESP32 WROOM Thing Plus relié par quelques fils à une carte de liaison GPS-RTK-SMA. C'est modeste, mais ça fonctionne, et même bien !

20 août 2020
Les premiers circuits imprimés pour le RTK Surveyor sont commandés. La V0.1 restait proche du circuit initial, soudé par Nate quelques jours avant. Nous avons remarqué des faiblesses, notamment des problèmes de dissipation de chaleur avec le u-blox ZED-F9P, un bruit de rétroaction qui a perturbé le récepteur GNSS, et quelques mauvaises pistes dans la carte elle-même. Rien d'exceptionnel à ce stade.

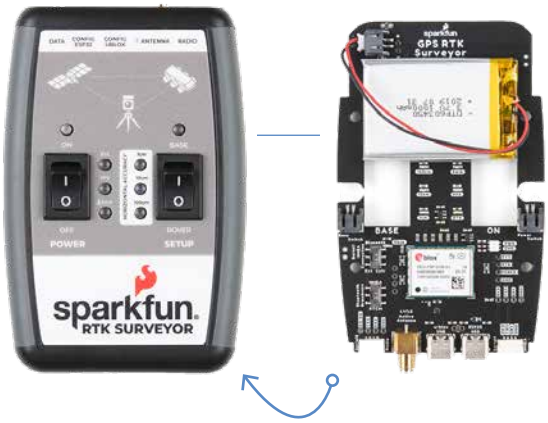
23 août 2020
Les premiers PCB pour le RTK Surveyor arrivent. C'est la première fois qu'un outil d'arpentage sur mesure se trouve au siège de SparkFun. En fait, la toute première fois qu'un véritable outil d'arpentage de la classe du RTK Surveyor se trouvait dans le bâtiment, c'était sur le chantier lors de sa construction il y a quelques années !

16 septembre 2020
La V0.2 du PCB pour le prototype RTK Surveyor est commandée. La correction des problèmes de bruit et de dissipation découverts dans la version précédente a permis d'obtenir un arpentage d'une précision impressionnante. La prochaine commande de cartes sera celle des circuits imprimés destinés au Surveyor lui-même !

14 et 15 octobre 2020
Les tutoriels "*Installation d'un système RTK de base pour engins roulants*" et "*Comment construire une station de référence GNSS autonome*" sont en ligne sur le site *SparkFun Learn*. Ce sont sur SparkFun.com les premiers bons exemples destinés aux utilisateurs intéressés par la topographie GNSS. Vous pouvez les lire dans ce numéro !

19 octobre 2020
Achat d'un boîtier pour le RTK Surveyor. Nous avions une idée de sa forme puisque nous sommes partis d'un modèle générique, il ne restait qu'à trouver un fournisseur.

20 octobre 2020
Conception et impression de l'autocollant pour le Surveyor. Une fois le boîtier connu, nous avons pu demander à notre graphiste de dessiner l'autocollant en profitant d'une accalmie dans son calendrier chargé. Aussitôt dessiné, aussitôt commandé.



jusqu'à 4 h (ou plus avec les banques de batteries USB courantes). Le RTK Surveyor abaisse le seuil d'accès pour les outils d'arpentage, tant sur le plan économique qu'en termes de facilité d'utilisation. Là où le prix d'appareils homologues est exorbitant ou qu'il s'agit de logiciel propriétaire, le RTK Surveyor, avec du logiciel libre, est disponible à une fraction du prix. D'où est née l'idée de proposer cet équipement de topographie ? À quoi ressemble le calendrier d'un nouveau produit expérimental de cette nature, et au fait, comme se présente-t-il ?

Inspiration

L'inspiration pour le RTK Surveyor est née du souhait de Nathan Seidle de combiner un ESP32 et une puissante carte GNSS RTK. Il n'existait malheureusement aucune option fiable à la fois praticable financièrement et à l'abri d'une gaine protectrice non conductrice. À cet objectif principal du Surveyor s'est ajouté le besoin grandissant d'autres fonctions. Nate savait ce qu'il attendait d'un outil de topographie, et il a mis sur la table tout ce que SparkFun avait dans son jeu. Pas de codage, coût faible, code libre, tout semblait se goupiller sans complication. Il a suffi de quelques mois pour sortir un RTK Surveyor d'usage public, le tout sans course effrénée à la baisse. Quel changement de rythme rafraîchissant que cette quête d'un outil d'arpentage accessible au plus grand nombre.

De toute évidence, la quantité de travail requise par la mise au point et la production d'un produit de SparkFun est considérable, surtout s'il s'agit d'un produit de grande précision pour un marché limité. Le calendrier confirme que c'est bien le lundi suivant la sortie du RTK Surveyor que nous avons commencé à travailler sur sa prochaine itération. En fait, il n'est pas encore question d'une nouvelle version immédiatement après le lancement initial, mais le pli est pris, nous ne cessons d'innover sur ce que nous créons. C'est vrai aussi pour le RTK Surveyor de SparkFun.

Regard sur l'avenir

Quelle sera la prochaine étape pour le RTK Surveyor ? Nous progressons déjà avec de nouvelles itérations de l'outil GNSS en équipant toutes les nouvelles constructions avec la V1.1 de la carte interne. La certification FCC et CE devrait suivre peu après avec quelques changements de fonctions supplémentaires. Ensuite le RTK Surveyor sera validé par SparkX pour les versions de production complètes, avec support technique et logo de la marque. Avec un peu de chance, la version 2.0 du RTK Surveyor arrivera fin 2021 ou début 2022. Suivez ce projet, nous préparons des mises à jour impressionnantes. Son prix restera autour de sa marque actuelle, bien plus économique que les versions à plus de 4 000 $ que vous voyez ailleurs.

(200701 – VF : Richard Kerr)

Accessoires

Elektor et SparkFun tiennent à votre disposition les accessoires mentionnés dans cet article.

› **RTK Surveyor de SparkFun**
www.elektormagazine.com/esfe-en-rtk1

› **23 octobre 2020**
La première série des PCB de production du RTK Surveyor est commandée. Une fois réceptionnés au siège de SparkFun la version 1.0 des PCB, tous les composants, le boîtier et les autocollants, nous sommes à pied d'œuvre pour constituer le stock initial du RTK Surveyor. C'est un processus compliqué pour SparkX en raison de la nature même de l'étude et de la mise au point d'un outil de type GNSS, surtout pendant une pandémie.

27 novembre 2020
Le premier lot de RTK Surveyor est assemblé et testé. Tout est prêt sept jours avant le jour J. C'est excitant !

30 novembre 2020
Le RTK Surveyor est soumis à la certification FCC et CE. Ce n'est pas la première fois que nous soumettions une étude originale de SparkFun à des certifications complètes pour le vendre dans la plupart des pays du monde, et nous avons beaucoup appris avec la première série. Avec Artemis, nous avions été contraints d'admettre que nous ne maîtrisions pas le temps requis pour obtenir l'approbation. Il s'est avéré que le développement d'un outil de levé GNSS peut prendre encore un peu plus de temps !

3 décembre 2020
Les PCB V1.1 de RTK Surveyor sont commandés, maintenant en rouge SparkFun ! Si vous connaissez Nathan Seidle, vous savez qu'il ne cesse jamais de travailler, et la mise au point du RTK Surveyor n'a pas fait exception. Et ce n'est pas fini !

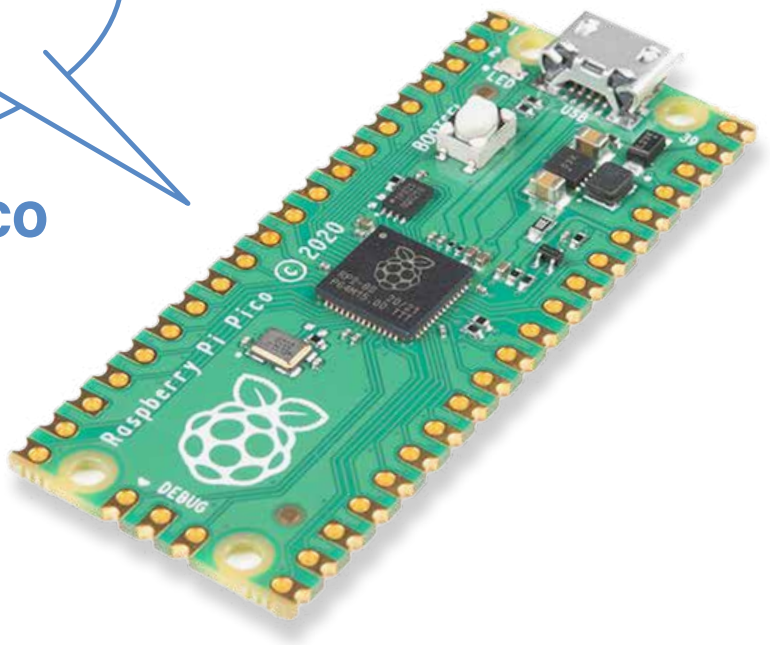
4 décembre 2020
Équipé des PCB V1.0, le produit RTK Surveyor est mis en vente encore dans la semaine. Le grand jour est arrivé, le RTK Surveyor est disponible pour tout le monde.

7 décembre 2020
On commence à travailler sur la V2.0 du RTK Surveyor.

Hello World

Raspberry Pi Pico et RP2040

Pleins feux sur les premiers microcontrôleur et micro-processeur de Raspberry Pi



Avra Saslow (États-Unis)

Confiante dans son rôle à jouer dans l'avenir de l'informatique, la Fondation Raspberry Pi s'est lancée dans le développement de produits révolutionnaires : un microcontrôleur et un microprocesseur RPi parfaitement compatibles avec MicroPython et C/C++. Bienvenue dans le nouveau monde de Raspberry Pi … le RPi Pico et le microprocesseur RPi RP2040.

Le RPi Pico (**fig. 1**) est idéal pour les projets de moindre complexité qui ne font tourner qu'un programme à la fois – situation où vous n'avez pas besoin de la taille et de la puissance d'un RPi complet - mais pour lesquels une plate-forme Arduino ne convient pas. Il s'accompagne d'une documentation incroyablement complète pour ses kits de développement logiciel (SDK) MicroPython et C/C++. Si vous êtes plus familier avec la programmation en Python, ou si avec votre connaissance d'Arduino vous pouvez passer au C++ (dont Arduino est une variante), alors le Pico pourrait bien devenir votre nouveau microcontrôleur préféré.

Présentation de RPi Pico

Allons au fait et examinons les spécifications techniques !

Le microprocesseur est le nouveau RP2040 conçu par Raspberry Pi. Le nom reprend les initiales de la fondation, suivies du nombre de cœurs du processeur (2), du type de processeur (M0+), de *floor(log2(ram/16k))*, et enfin de *floor(log2(nonvolatile/16k))*. La **figure 2** illustre l'algorithme de nommage !

La puce intègre une amorce UF2 qui permet de flasher le microcontrôleur avec le micrologiciel via USB (par glisser-déposer), ainsi que des routines de calcul en virgule flottante. Elle est équipée de cœurs de processeur Dual Cortex M0+ avec une horloge variable jusqu'à 133 MHz. Couplée à une matrice de bus à haute performance, elle peut obtenir la pleine puissance sur les deux cœurs en même temps. Ce niveau de performance pourrait autoriser à terme des modèles d'apprentissage machine avec TensorFlow pour MicroPython. Elle dispose d'une vaste RAM interne (264 Ko de SRAM), mais utilise une mémoire flash externe (2 Mo de mémoire flash embarquée), ce qui permet à l'utilisateur de choisir la mémoire qu'il lui faut.

- Modes de fonctionnement à faible puissance (veille et veille profonde).
- Un USB intégré qui peut servir à la fois de dispositif et d'hôte.
- Divers périphériques numériques dont 2x UART, 2x I²C, 2x SPI, jusqu'à 16 canaux PWM, un minuteur avec quatre alarmes et un compteur en temps réel.
- 30 GPIO multifonctions dont quatre utilisables en CA/N (**fig. 3**).
- Un capteur de température.
- 8× automates finis d'E/S programmables (PIO) pour la prise en charge de périphériques personnalisés. Sur la plupart des microcontrôleurs, il faut faire du *bit-bang*, c'est-à-dire utiliser l'unité centrale et du logiciel pour activer et désactiver directement des broches. Cela fonctionne mais prend

Listage 1. Commandes de terminal pour l'installation de MicroPython.

```
echo „Obtaining MicroPython“;
  cd ~/;
  mkdir pico;
  cd pico;
  git clone -b pico git@github.com:raspberrypi/micropython.git;

  echo „Obtain additional tools“;
  sudo apt update;
  sudo apt install cmake gcc-arm-non-eabi;
echo „Building MicroPython“;
  cd micropython;
  git submodule update --init --recursive;
  make -C mpy-cross;
  cd ports/rp2;
  make
```

du temps, surtout si vous utilisez des interruptions, et cela consomme de précieuses ressources de calcul dont vous pourriez avoir besoin pour autre chose. Pour traiter les données qui entrent et sortent, on peut utiliser les E/S programmables ou les 2 blocs PIO de quatre automates finis du Pico, et le décharger ainsi des contraintes de traitement pour la gestion des protocoles de communication.

- Et bien davantage !

En ce qui concerne les logiciels, le RP2040 est compatible avec les environnements de développement multiplateforme C/C++ et MicroPython, y compris un accès facile au débogage. Même si le RPi Pico ne bouleverse pas forcément les références informatiques, les documentations du RP2040 et du Pico élaborées par la Fondation Raspberry Pi sont d'une profondeur technique inégalée. Avec de telles caractéristiques, le RPi Pico est puissant, incroyablement extensible et sa taille est (presque) à la hauteur de son nom "Pico". Il s'aligne sur tous les produits Raspberry Pi tant par sa taille que par son accessibilité en termes de caractéristiques techniques et de coût. La Fondation Raspberry Pi a réussi son entrée sur le marché des µC !

MicroPython sur le Pico

Vous avez peut-être remarqué que le langage pris en charge en passant du RPi à RPi Pico a changé de Python à MicroPython. C'est parce que Python est trop gourmand en ressources pour fonctionner sur de petits microcontrôleurs. Pour les microcontrôleurs comme le Pico, on utilise donc un portage de Python, appelé MicroPython. Presque identique à Python, il ne contient qu'un sous-ensemble des modules de la bibliothèque standard, de sorte qu'il se contente de peu de RAM (environ 16 K selon MicroPython).

Il faut quelques étapes pour faire fonctionner MicroPython sur le RPi Pico. Le processus commence par le clonage du dépôt Github de MicroPython et l'installation de CMake et de *GNU Embedded Toolchain* pour Arm afin d'aider à la construction du logiciel. Les commandes du Terminal (écrites en script shell) construisent les programmes exécutables et les bibliothèques à partir du code source de MicroPython (**listage 1**).

À partir de là, l'installation du logiciel MicroPython se fait par glisser-déposer, où vous faites glisser le microprogramme .uf2 sur la carte, vue comme périphérique de stockage de masse USB. Il faudra maintenir le bouton BOOTSEL enfoncé pour la faire passer dans ce mode de stockage de masse USB. Une fois le microprogramme chargé sur la carte, vous pourrez vous connecter au REPL (*Read Evaluate Print Loop*) de MicroPython, qui est un moyen simple de tester

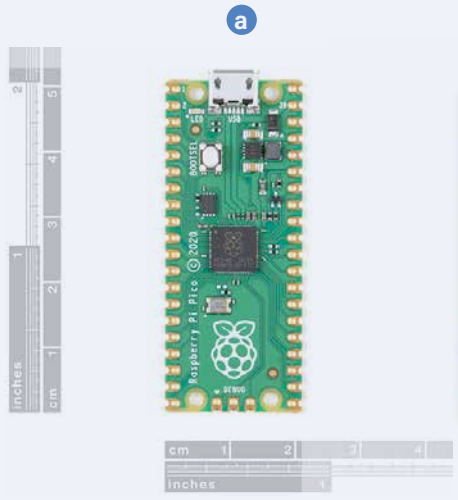


Figure 1a. Le Raspberry Pi Pico est une excellente solution lorsque vous n'avez besoin ni de la taille ni de la puissance d'un RPi complet. Fig.1 b : Regardez le dessous du Pico.

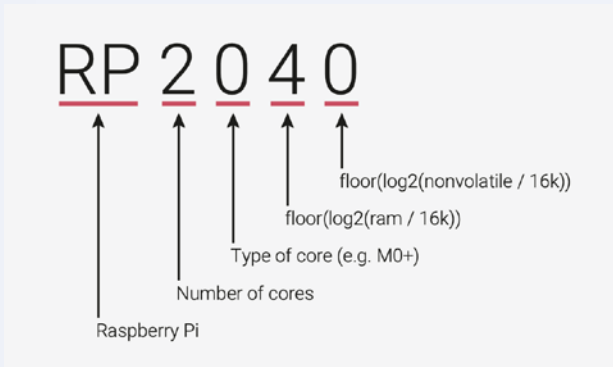


Figure 2. L'algorithme de nommage

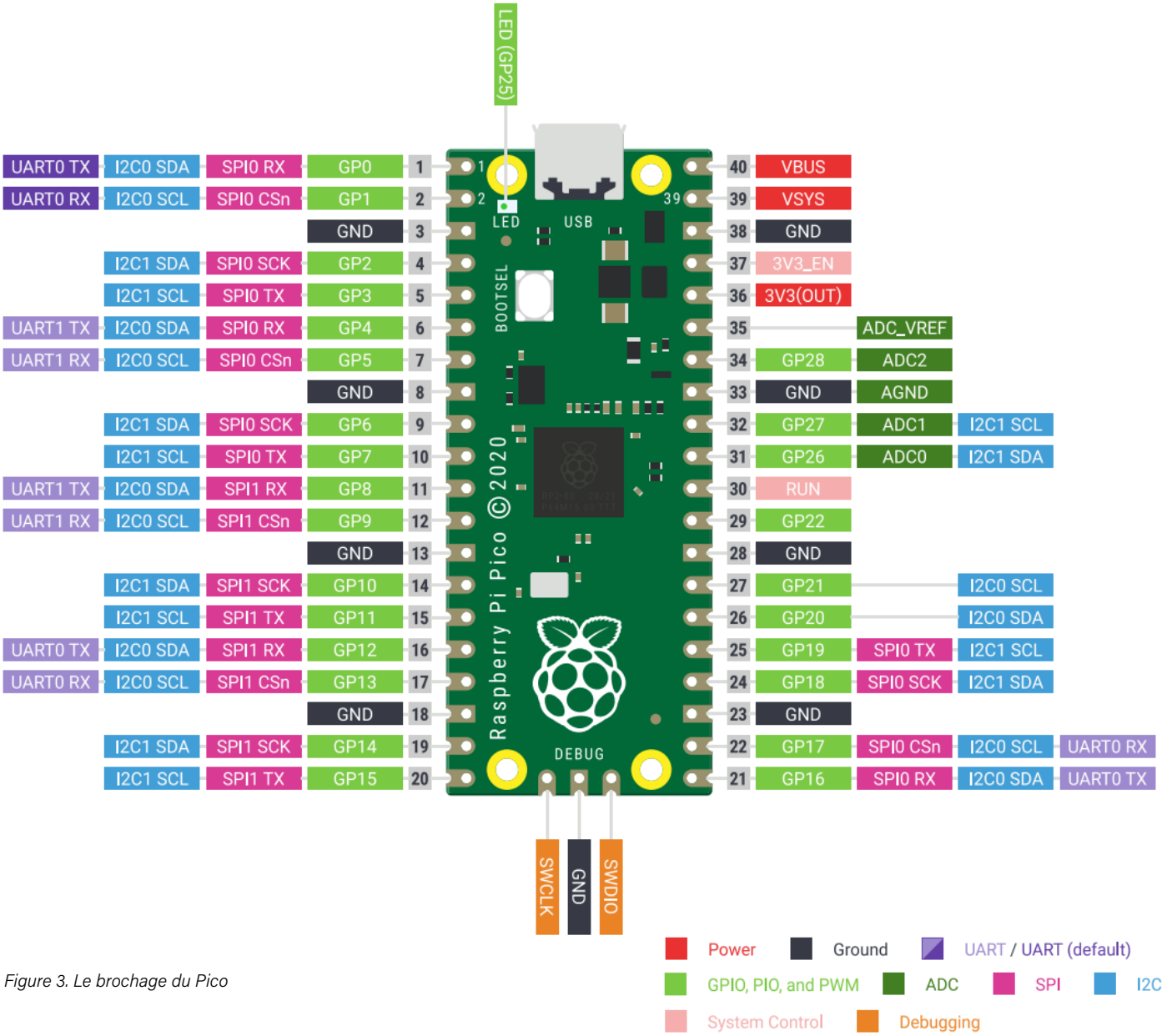


Figure 3. Le brochage du Pico

le code et d'exécuter des commandes. En utilisant l'exemple de code de Raspberry Pi, j'ai pu facilement mettre en place un code qui utilise un minuteur pour faire clignoter la LED embarquée (**listage 2**).

Un peu de morse ?

Comme je travaillais déjà sur le Pico avant sa sortie alors qu'il était encore *top secret*, je voulais communiquer avec lui de manière un peu cryptée. Aussi, à partir du code « *Blink an LED* », j'ai fait un script MicroPython qui compose en morse le message *Hello World* (**listage 3**). Comme il est difficile d'illustrer sur papier le clignotement d'une LED, je lui ai aussi fait imprimer la traduction du code morse sur le terminal (**fig. 4**). Et ça a donné : Hello World, RPi Pico !

Listage 2. Une minuterie pour faire clignoter une LED.

```
##credit of the Raspberry Pi Foundation
 from machine import Pin, Timer
led = Pin(25, Pin.OUT)
 tim = Timer()
 def tick(timer):
 global led
 led.toggle()

 tim.init(freq=2.5, mode=Timer.PERIODIC,
callback=tick)
```

.... . .-.. .-.. --- / .-- --- .-. .-.. -..

Figure 4. "Hello World" en morse

Listage 3. Un script MicroPython qui produit „Hello World“ en morse.

```
#-------------------------------------------------------------------------------
# Importation des bibliothèques nécessaires, connexion à la LED intégrée,
règlage de la fréquence de la LED
#-------------------------------------------------------------------------------

import time
from machine import Pin
led=Pin(25,Pin.OUT)# la LED du Pico est sur la broche 25
BlinkRate=0.25

#-------------------------------------------------------------------------------
# Fonctions pour la durée du signal en code Morse et code lui-même

#-------------------------------------------------------------------------------

def dash():
    led.value(1)
    time.sleep(4*BlinkRate)
    led.value(0)
    time.sleep(BlinkRate)

def dot():
    led.value(1)
    time.sleep(BlinkRate)
    led.value(0)
    time.sleep(BlinkRate)

def pause():
    time.sleep(BlinkRate)


code = {'A':'.-','B':'-...','C':'-.-.','D':'-..','E':'.','F':'..-.','G':'--.',
    'H':'....','I':'..','J':'.---','K':'-.-','L':'.-..','M':'--','N':'-.',
    'O':'---','P':'.--.','Q':'--.-','R':'.-.','S':'...','T':'-','U':'..-',
    'V':'...-','W':'.--','X':'-..-','Y':'-.--','Z':'--..',
    '0':'-----','1':'.----','2':'..---','3':'...--','4':'....-',
    '5':'.....','6':'-....','7':'--...','8':'---..','9':'----.',
    '.':'.-.-.-',
    ',':'--..--',
    '?':'..--..',
    '/':'--..-.',
    '@':'.--.-.',
    ' ':' / ',

}

#-------------------------------------------------------------------------------
# Fonction qui renvoie simplement la phrase en morse à partir de la phrase
anglaise en majuscules
#-------------------------------------------------------------------------------

def convertToMorseCode(sentence):
  sentence = sentence.upper() #transformation en majuscule pour que le
dictionnaire reconnaisse le caractère
    secretSentence = „“ #effacement de la phrase à ajouter
    for i in sentence: #pour chaque caractère de la phrase, le chercher dans
le dictionnaire et le remplacer par le code Morse
    secretSentence += code[i] + „ „
    return secretSentence

#-------------------------------------------------------------------------------
# Fonction principale qui fait clignoter la LED selon la phrase en morse
#-------------------------------------------------------------------------------

while True:
    sentence = „Hello World“
    secretSentence = convertToMorseCode(sentence)
    for i in secretSentence:
        if i == „.“:
            dot()
        elif i == „-“:
            dash()
        else:
            pause()
```

Le futur antérieur de Raspberry Pi

Avec son premier ordinateur monocarte Raspberry Pi a ouvert en 2012 un nouveau monde informatique. Jamais auparavant un PC polyvalent aussi petit et à un coût aussi faible n'avait été produit. La Fondation Raspberry Pi aura su démontrer que ses capacités et fonctions interactives étendues permettaient de l'utiliser pour autre chose que la simple navigation sur le web. Avec l'informatique physique permise par un GPIO à 40 broches et Python préchargé, le RPi a littéralement brisé la barrière des coûts et des compétences liés à la conception de projets automatisés basés sur l'IdO. Aujourd'hui n'importe qui, quelles que soient ses compétences techniques, peut connecter des capteurs et des affichages au RPi et le programmer en Python pour faire du calcul, effectuer des tâches d'entrée/sortie multiples ou afficher des données sur un serveur web.

La liaison durable avec Python

Python est resté un pilier de Raspberry Pi au cours de la dernière décennie ; malgré la mise à niveau de composants physiques comme la puissance de calcul, le graphique et les E/S par rapport aux modèles précédents, le support du langage lui-même n'a pas changé. Je pense que c'est en partie parce que Python se marie si bien, par nature, avec le prototypage et l'open source du RPi. Il existe des douzaines de bibliothèques Python qui permettent le traitement rapide des données, l'analyse et l'apprentissage machine. Tout comme l'extensibilité offerte par le GPIO au RPi, Python est extensible, modulaire et flexible. Les applications web comme Jupyter Notebooks ouvrent une vaste gamme de flux de travail dans les domaines de la science des données, du calcul scientifique et de l'apprentissage machine, avec des greffons qui ajoutent de nouveaux composants et s'intègrent aux composants existants. Il n'est donc pas surprenant que Python s'associe si bien avec RPi.

Présentation des produits **SparkFun RP2040** !

Le RPi Pico est construit autour du puissant RP2040, et donc beaucoup des qualités indéniables du microcontrôleur proviennent des caractéristiques techniques de la puce. Pour le RP2040 Sparkfun propose trois nouvelles cartes qui utilisent toutes les caractéristiques de calcul offertes par la puce, ainsi que les environnements de développement multiplateforme C/C++ et MicroPython.

Nous avons ajouté à nos cartes pour le RP2040 une petite touche d'innovation propre à SparkFun :

- SparkFun RP2040 Thing Plus (**fig. 5**)
- Pro Micro RP2040 (**fig. 6**)
- carte processeur MicroMod RP2040 (**fig. 7**)

Toutes ces cartes ont huit fois plus de flash que le Pico. 16 Mo de mémoire flash, c'est bien sûr beaucoup pour sauvegarder des données hors ligne ! Avec autant de mémoire flash sur chaque carte, des modèles d'apprentissage machine avec TensorFlow utilisant le RP2040 sont parfaitement réalisables. Les cartes partagent également toutes les caractéristiques techniques du RP2040, notamment :

- Deux processeurs ARM Cortex-M0+ (jusqu'à 133 MHz)
- 264 ko de SRAM intégrée dans six banques
- Six E/S dédiées pour SPI Flash (avec XIP)
- 30 GPIO multifonctions : matériel dédié pour les périphériques les plus courants ; E/S programmable pour prise en charge de périphériques supplémentaires ; et quatre canaux CN/A 12 bits avec un capteur de température interne
- Développement multiplateforme C/C++ et MicroPython avec accès facile au débogage et à l'amorçage UF2

Chacun de ces microcontrôleurs RP2040 construits par SparkFun a cependant ses propres avantages, alors observons-les de plus près.

SparkFun RP2040 Thing Plus

Au facteur de forme *feather*, avec 18 broches GPIO, la RP2040 Thing Plus de SparkFun est une excellente carte pour les projets qui peuvent nécessiter de la mobilité ou qui intègrent un jeu complet de périphériques. Elle est équipée d'un connecteur JST pour batterie LiPo à cellule unique, et sert à la fois de circuit de charge et de jauge. Elle comporte aussi un emplacement pour carte micro-SD pour l'enregistrement de données ou un stockage supplémentaire. La Thing Plus RP2040 comprend une LED WS2812 RGB adressable, des broches JTAG PTH, quatre trous de fixation et, bien sûr, un connecteur Qwiic qui facilite grandement le prototypage avec des capteurs et des affichages.

Pro Micro RP2040

La Pro Micro RP2040 est au format Pro Micro et offre une compatibilité USB-C complète. Comme la Thing Plus RP2040, elle comprend une LED WS2812 RGB adressable, un connecteur Qwiic, ainsi que des pastilles crénelées (*castellated pads*), et un bouton de démarrage et de réinitialisation. Les vias crénelées font de ce modèle un candidat idéal pour des projets tels que la création d'un contrôleur pour des programmes ou jeux informatiques ou la construction d'un clavier. Plus précisément, cette carte pourrait vous permettre de construire un clavier entièrement personnalisable pour l'adapter à vos besoins spécifiques. Si vous cherchez de l'inspiration pour un clavier qui permet des actions particulières, je recommande le projet de Jason Rudolph : https://github.com/jasonrudolph/keyboard. En fait, tout périphérique avec des E/S dédiées s'intégrerait parfaitement avec la Pro Micro RP2040. De plus, on peut construire tout cela sans rien perdre du confort de MicroPython.

Carte processeur MicroMod RP2040

Enfin, SparkFun sort une carte à processeur MicroMod pour le RP2040. L'écosystème MicroMod, est un système modulaire qui permet d'échanger les cartes porteuses (ou cartes de support) et les cartes à processeur. Informez-vous sur www.sparkfun.com/micromod. Ainsi, au lieu d'acheter des cartes qui ont un processeur prédéfini ou des types spécifiques d'entrées et de sorties, MicroMod vous permet de changer d'avis en cours de projet, soit avec le processeur, soit avec la carte porteuse.
SparkFun propose diverses cartes de support, équipées chacune d'un connecteur Qwiic, ce qui vous permet d'ajouter toujours plus de capteurs et d'afficheurs à votre projet. Dès à présent, les cartes porteuses offrent de multiples options pour tirer parti du RP2040.

- ATP (*all the pins* = liaisons à toutes les broches)
- Saisie et affichage
- Enregistrement de données
- Apprentissage machine
- Météo

Désormais, comme avec les autres cartes processeurs (avec les puces ESP32, Artemis, nRF52840 et SAMD51), vous pouvez utiliser

le RP2040 avec la carte à processeur MicroMod RP2040. Vous tirez ainsi pleinement parti de la puce RP2040 et pouvez ajouter les périphériques nécessaires pour votre projet. Si vous souhaitez vous familiariser rapidement avec la puce RP2040 et voir comment elle effectue diverses tâches, la carte processeur MicroMod RP2040 est aussi simple que possible pour l'expérimentation et le prototypage.

Dernières réflexions

Plus que jamais, la Fondation Raspberry Pi poursuit la création de produits performants et accessibles, et encourage les projets créatifs et open source. Le Raspberry Pi Pico, le RP2040, et tous les produits associés de SparkFun disposent non seulement d'une vaste documentation inégalée, mais ils sont à l'origine d'un environnement de développement avec MicroPython innovant. La seule question est de savoir comment commencer vos créations avec le RP2040. Est-ce par le biais du Pico, très avantageux ? Ou la méthode de prototypage modulaire de MicroMod ? Ou peut-être la petite taille du Pro Micro. Ou bien encore grâce à un projet portable utilisant la carte RP2040 Thing Plus. Peu importe votre choix, allez-y ! Bâtissez des projets en utilisant toutes les capacités de MicroPython et des logiciels *open source*. Dites *hello* à votre nouveau microcontrôleur favori à la sauce Raspberry Pi !

200711-04 – VF : Denis Lafourcade

Revendeurs agréés Raspberry Pi

Les produits à base de Raspberry et les accessoires RPi vous intéressent-ils ? SparkFun et Elektor sont tous deux revendeurs agréés de Raspberry Pi.

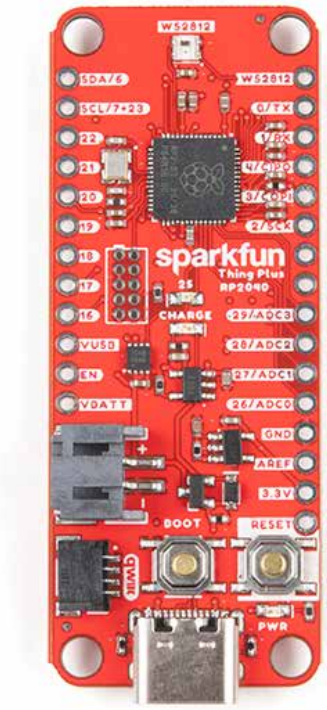


Figure 5. SparkFun Thing Plus

Figure 6. SparkFun Pro Micro

Figure 7. Carte processeur MicroMod RP2040

Accessoires

Vous trouverez chez SparkFun et Elektor les principaux accessoires mentionnés dans cet article :

- **Carte à processeur MicroMod RP2040**
  www.elektormagazine.com/esfe-en-rpipico1
- **Pro Micro RP2040**
  www.elektormagazine.com/esfe-en-rpipico2
- **Carte à microcontrôleur Raspberry Pi Pico**
  www.elektormagazine.com/esfe-en-rpipico3
- **RP2040 Thing Plus de SparkFun**
  www.elektormagazine.com/esfe-en-rpipico4

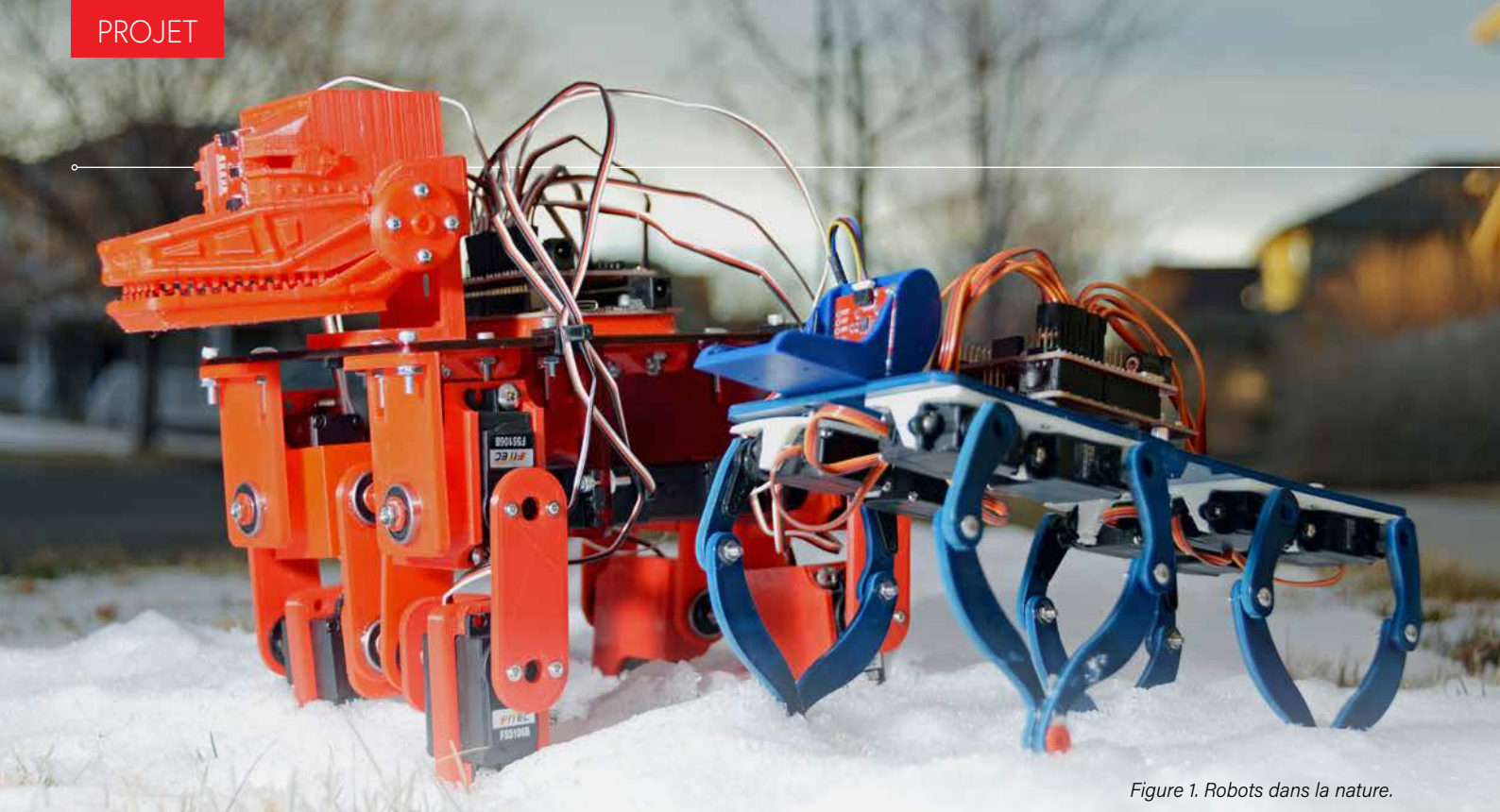


Figure 1. Robots dans la nature.

Fabriquer soi-même des robots à quatre pattes

Rob Reynolds (États-Unis)

Que diriez-vous de construire un robot quadrupède ? Oui, bien sûr ! Je propose de construire deux plates-formes extensibles, avec de multiples points de fixation pour le montage de servos, l'ajout de capteurs et la mise en place de microcontrôleurs.

Je ne peux pas parler au nom de tous les électroniciens, mais on peut dire que dès l'enfance, lorsque la plupart d'entre nous imaginaient des robots, nous pensions d'abord aux humanoïdes bipèdes dotés de compétences interactives que nous appellerions aujourd'hui l'IA, sans toutefois la fluidité des mouvements des humains. Il s'agissait de rêves ou de créatures imaginées pour le cinéma, animées par des acteurs en costume métallique, des marionnettistes cachés, ou des génies comme Ray Harryhausen et son hibou robot Bubo du *Choc des Titans*. Me voici adulte (chronologiquement du moins), et bien déçu de constater que ce que nous appelons robots domestiques sont ici un aspirateur ou là un robot pâtissier, c'est-à-dire un simple mixeur sur un socle. Si c'est ça la robotique, il ne me reste plus qu'une chose à faire : créer mes propres robots (**fig. 1**).

Conception du projet

L'un des aspects que je préfère dans la création de quelque chose comme un robot - ou, de n'importe quel autre projet d'ailleurs – c'est le point de départ. Tout est encore possible. Dans la comédie musicale de Stephen Sondheim « Sunday in the Park with George », le personnage de George dit en se souvenant de son grand-père, le peintre George Seurat : « Blanc : une page ou une toile vierge. Son préféré - tant de possibilités ». À ce stade, les possibilités sont illimitées, peut-être bêtement parce que je n'en sais pas assez – et espérons que je n'en saurai jamais assez – pour savoir ce qui est impossible. Dans mon esprit, tout est donc possible. À quoi mon robot devrait-il ressembler ? Pour l'effet « wow », il pourrait s'agir d'un robot humanoïde bipède longiligne, comme ceux du film « *I, Robot* ». En pratique, si je veux un robot qui puisse facilement se déplacer chez moi et interagir avec moi ou avec son environnement, il serait plus simple de le monter sur quatre roues. Ce qui nous ramènerait en gros au stade du « robot-aspirateur ». Il faut chercher un juste milieu, quelque chose d'un peu pratique, mais d'assez impressionnant pour attirer l'attention. Je voudrais aussi pouvoir le faire évoluer. J'ai décidé qu'il serait donc quadrupède.

Pour m'inspirer, il y a les travaux de *Boston Dynamics* sur les quadrupèdes, de Big Dog en 2005, massif et un peu maladroit, qui crapahute en terrain accidenté, à Spot en 2018, élégant et agile, qui ouvre les portes et

danse sur *Uptown Funk*. C'est un bel exemple de progression et d'itération. Pour mon quadrupède, je voulais une plate-forme de base avec des possibilités d'extensions, voire d'échange de modules. J'ai envisagé plusieurs configurations, et finalement opté pour deux conceptions différentes. Après coup, il est évident que l'une a été inspirée par Big Dog, et l'autre par Spot, avec chacune ses propres capacités, mais extensibles l'une et l'autre.

J'ai commencé par le robot inspiré de Spot, plus petit et plus élégant. Avec en tête la mécanique que je voulais utiliser, j'ai fait une série de lignes droites sur mesure, qui a donné des pièces sans intérêt. Ensuite j'ai réfléchi un peu plus au Spot de Boston Dynamics, et j'ai réalisé que dans ce cas, je pouvais laisser la forme suivre la fonction. Je pouvais me permettre quelques courbes, pour améliorer un peu l'apparence de ce robot. En fait, j'ignore s'il est plus beau, mais il est un peu moins anguleux, et aussi moins stable. Disons que j'ai réussi à le rendre moins disgracieux.

J'ai ensuite commencé à travailler sur le deuxième robot, le robot Big Dog, d'inspiration plus industrielle. Après plusieurs heures de travail de conception, je me suis demandé à quoi bon réinventer la roue. J'ai cherché des modèles utilisables, du moins comme point de départ. J'ai trouvé une superbe réalisation de *Technovation* décrite sur *Instructables* [1]. Les deux plates-formes ont de multiples points de fixation pour le montage de servos, l'ajout de capteurs et la mise en place du microcontrôleur. Je prévois d'utiliser le Redboard Artemis de SparkFun, qui a l'empreinte familière de l'Arduino Uno. Le Redboard Artemis offre plus de mémoire et de vitesse que l'Uno, plus le BLE embarqué, et en utilisant ce schéma de perçage, je peux facilement passer au SparkFun Artemis ATP si j'ai besoin de plus de broches, car il partage ce même schéma de perçage. J'utiliserai aussi le *shield* de pilotage de moteur sans fil de SparkFun.

Les pattes du robot quadrupède

Pour les pattes, j'ai fait deux choix différents pour les deux robots. Pour le plus petit - appelons le Bluesette - j'ai utilisé une liaison à cinq barres, qui n'est peut-être pas idéale pour un quadrupède, surtout avec des servomoteurs. Cela vaut la peine de l'expérimenter juste pour apprendre et se frotter au mouvement et à la cinématique inversée. La liaison à cinq barres est un mécanisme à deux degrés de liberté dans lequel les cinq barres sont reliées en boucle (**fig. 2**).

Ce mécanisme contrôle les coordonnées x et y de l'articulation D, appelée point final ou *effecteur* final, (ou dans notre cas, le pied du robot) en ajustant simultanément les angles d'entrée thêta 1 et thêta 2, contrôlant ainsi l'angle des barres B2 et B5. En déplaçant l'effecteur final de chaque patte le long d'une trajectoire elliptique, nous pouvons faire marcher le robot en avant et en arrière, ajuster sa hauteur, et même tourner en rond.

Comme je l'ai dit, ce n'est peut-être pas la meilleure locomotion avec des servomoteurs pour un quadrupède, mais si vous passez à un ensemble de huit moteurs sans balais à haute performance, cette configuration peut devenir très efficace. Le projet Stanford Doggo en est un bon exemple [2] : grâce à la vitesse, à la puissance et à la commande des moteurs sans balais, ce petit quadrupède peut sauter à plus d'un mètre de hauteur et même faire des sauts périlleux arrière (**fig. 3**).

Mon robot plus grand et plus gros, appelé Big Red, utilise le mécanisme le plus courant pour les pattes de quadrupèdes, connu sous le nom de manipulateur série. Chaque articulation a son propre moteur, de la base à l'actionneur final. Ces articulations sont souvent celles de l'épaule, du coude et du poignet, en particulier lors de la création de tout type de structure de bras anthropomorphe, comme un automate de placement mono-bras, par exemple sur une chaîne de montage automobile. En

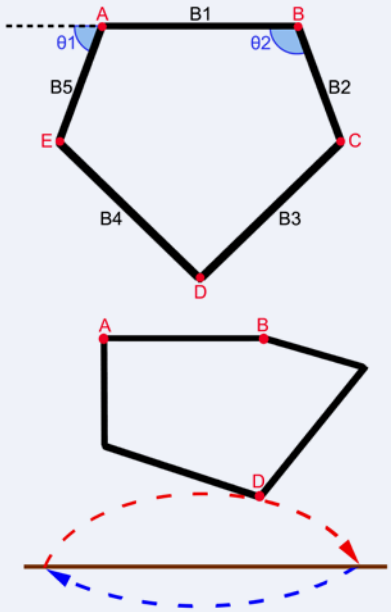


Figure 2. Mécanismes à cinq barres.

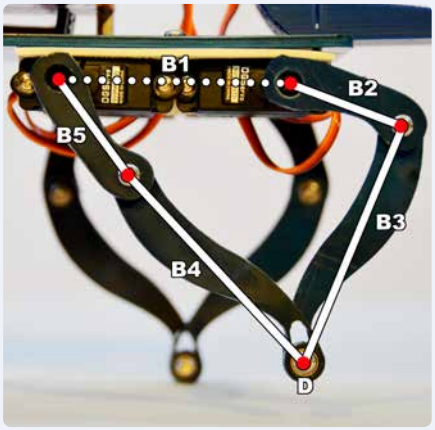


Figure 3. Les deux systèmes utilisent deux servos pour contrôler la position des pattes, mais de manière très différente.

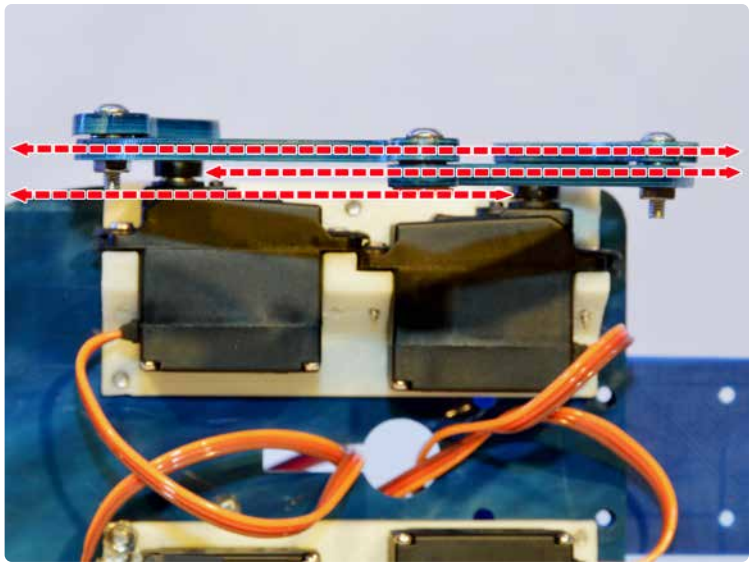


Figure 4. En décalant les supports de servo de l'épaisseur du matériau de la patte, tout reste aligné.

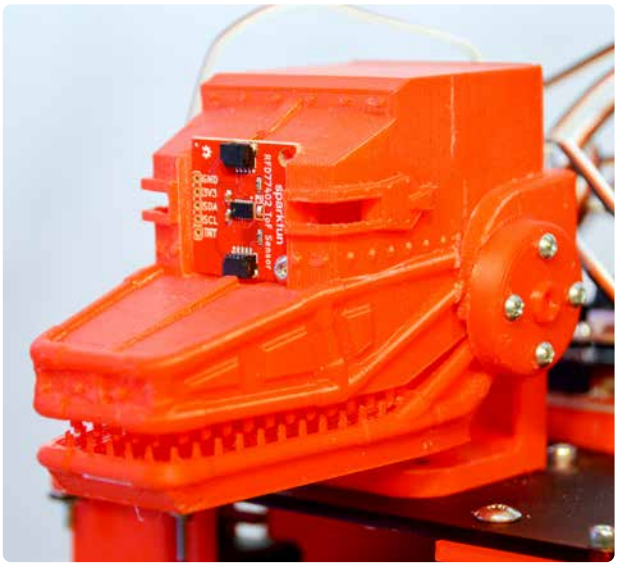


Figure 5. Tête de chien.

gardant les bras courts pour cette première version de Big Red, je peux disposer d'une force et d'une stabilité impossible à atteindre avec les pattes de Bluesette.

Pour la construction et les matériaux, j'ai utilisé une combinaison de pièces en acrylique de 3 mm découpées au laser et de pièces en PLA imprimées en 3D. De nombreux ateliers partagés disposent d'imprimantes 3D, et certains proposent aussi des découpeuses laser. Si vous n'avez pas accès à la découpe laser, il reste le contreplaqué de 3 mm, facile à travailler avec des outils manuels. Les plus importantes sont les pièces imprimées en 3D, en particulier les supports de servo. Même sans imprimante 3D, vous pouvez fixer vos moteurs solidement avec des serre-câbles ou de la colle thermofusible, du moins pour le robot avec pattes à cinq barres. Soyez créatif. Après tout, ne sommes-nous pas des bidouilleurs ici ?

Même s'ils ne sont pas complètement nécessaires, j'utilise des roulements pour toutes les articulations des deux robots. Les mouvements sont beaucoup plus doux et cela réduit les frottements au niveau des articulations et donc la consommation de courant des servomoteurs. Une autre caractéristique importante de la conception de Bluesette est le décalage de 3 mm (ou de l'épaisseur du matériau utilisé pour les pattes) des moteurs pas à pas. Si les moteurs pas à pas sont placés sur un même plan (**fig. 4**), les réunir en boucle entraînera une torsion latérale au niveau de l'une des articulations, ce qui nécessitera à nouveau plus de courant pour entraîner les servomoteurs.

> *La base de mes quadrupèdes, c'est une plate-forme avec des possibilités d'extensions et même d'échange de modules.*
>
> Rob Reynolds

Il me fallait encore une sorte de support frontal pour les capteurs. J'aurais pu concevoir vite fait bien fait un support de montage rudimentaire, mais j'y ai vu une autre occasion de me montrer créatif. Pour Bluesette, j'ai juste assemblé quelques formes primaires, chanfreiné les arêtes, ajouté quelques trous de montage et imprimé le tout. Big Red, en revanche, a reçu un peu plus de mon temps et de mon attention. Je me suis souvenu que dans le film « L'île aux chiens » il y avait de très beaux chiens robots, donc en utilisant des images fixes du film, j'ai fait jouer mes muscles de conception 3D et j'ai recréé les têtes des chiens robots, avec des ajustements pour pouvoir placer la plupart de nos cartes de capteurs Qwiic (**fig. 5**). Je craignais que cela n'alourdisse l'avant de ce robot, mais je me suis dit que cela pourrait être compensé avec une grosse batterie à l'arrière.

Pour l'assemblage des deux robots, j'ai utilisé des vis M3 de différentes longueurs (sauf pour les entretoises, qui ont des 4x40). J'ai imprimé en 3D des bagues pour les roulements de Big Red et les articulations des poignets de Bluesette.

Une note sur le codage et les tests. Si vous concevez et construisez quelque chose qui peut bouger, il va bouger, mais généralement au moment où vous vous y attendez le moins. J'ai appris cette leçon pour la première fois alors que je travaillais sur une petite voiture autonome. J'avais ajouté du code à mon croquis et le téléchargeais sur mon véhicule. Dès le téléchargement terminé, ma petite voiture est partie à toute vitesse, s'envolant de mon établi et se fracassant contre le mur du fond. Nathan, le fondateur de SparkFun, a eu le même problème lors de l'une de ses constructions, mais il travaillait sur un véhicule autonome suffisamment grand pour y monter. Quand il a décollé, il a arraché son ordinateur portable de son bureau. Il faut donc toujours s'assurer que les pièces qui font bouger votre création ne touchent pas le sol. Si vous travaillez sur une voiture, posez-la sur une boîte plus petite que l'empattement du véhicule et plus haute que sa garde au sol. Pour ces robots, je disposais de quelques petits profilés en aluminium rainurés en T MicroRax, que j'ai utilisés avec une pièce sur mesure vite faite en 3D pour garder les pieds décollés du sol (**fig. 6**). Dans l'UE ou au Royaume-Uni, vous trouverez des accessoires similaires chez MakerBeam.

Accessoires

Vous trouverez certains des accessoires mentionnés dans cet article chez SparkFun et Elektor !

- **Elektor MIT App Inventor Bundle**
  www.elektor.com/elektor-mit-app-inventor-bundle
- **SparkFun RedBoard Artemis**
  www.elektormagazine.com/esfe-en-qrobot1
- **SparkFun Wireless Motor Driver Shield**
  www.elektormagazine.com/esfe-en-qrobot2

Le code

Après l'assemblage, commençons l'écriture du code (**listage 1**). Notre principale ressource sera la bibliothèque Servo. À mesure que nous avancerons et ajouterons des capacités supplémentaires, d'autres bibliothèques entreront en jeu. Pour commencer, il faut juste que notre croquis crée et relie entre eux tous les objets servo. Un moyen simple et rapide est de partir de quelque chose comme le croquis **Servo Sweep** et de faire quelques ajustements mineurs. Pour rester simple, à la fois pour aujourd'hui et pour plus tard, on peut énumérer les servos dans une boucle `for`. Si vous prévoyez de poursuivre avec ces robots ou d'autres de conception similaire, c'est un excellent point de départ, car il suffira d'ajuster le nombre de boucles au nombre de servos à créer et relier. Nous définirons ensuite une position de départ pour chaque servo et lui ferons passer un court test de balayage, juste pour nous assurer qu'ils communiquent entre eux et sont correctement alimentés. Lorsque je travaille sur un code dont je sais qu'il continuera à être développé, et peut-être même à avoir de multiples itérations, je trouve qu'il vaut mieux le décomposer en fonctions plutôt que de tout avoir dans la `void:loop()`. Cela facilite les choses pour trouver, modifier, et copier/coller dans d'autres itérations du croquis ou dans des croquis complètement différents qui ont juste besoin d'un petit bout d'un code existant. Sans doute parce que je suis assez vieux pour me souvenir de la programmation en BASIC et de l'utilisation de sous-programmes.

Vous voici avec un point de départ. Les prochaines étapes, du moins pour moi, iront par ordre croissant de difficulté. Abaisser la hauteur du robot jusqu'à peu près la moitié de sa hauteur totale est assez simple et peut donner l'impression qu'il se tapit. La prochaine étape serait logiquement une paire de fonctions de marche, à la fois en avant et en arrière. Puis tourner des deux côtés. Ensuite, on peut passer à la commande par BLE ou au mouvement autonome, selon l'objectif final. Connecter votre ordinateur portable ou tablette via Bluetooth et envoyer des commandes série à un seul caractère serait un moyen simple et rapide pour commencer à commander votre robot à distance, ou bien le coupler avec une carte micro:bit comme télécommande portable. Pour commander votre robot depuis un téléphone Android, vous pourriez utiliser l'appli Inventor du MIT (https://appinventor.mit.edu/) et créer une interface sur un appareil que vous avez toujours avec vous. Et si vous hésitez, vous pouvez toujours écrire une fonction pour chaque commande, puis parquer celles que vous décidez de ne pas utiliser. Des capteurs de proximité équipent couramment les robots, ils constitueront l'étape suivante la plus pertinente. Peut-être souhaitez-vous ajouter un capteur d'environnement pour des données comme la température ou l'humidité. La dernière étape pour cette fois sera d'ajouter un peu d'apprentissage machine ou d'intelligence artificielle de base. Avec TensorFlow, il sera assez facile d'entraîner votre robot à répondre à des commandes vocales simples : stop, marche, assis, attaque, tout ce que vous voulez !

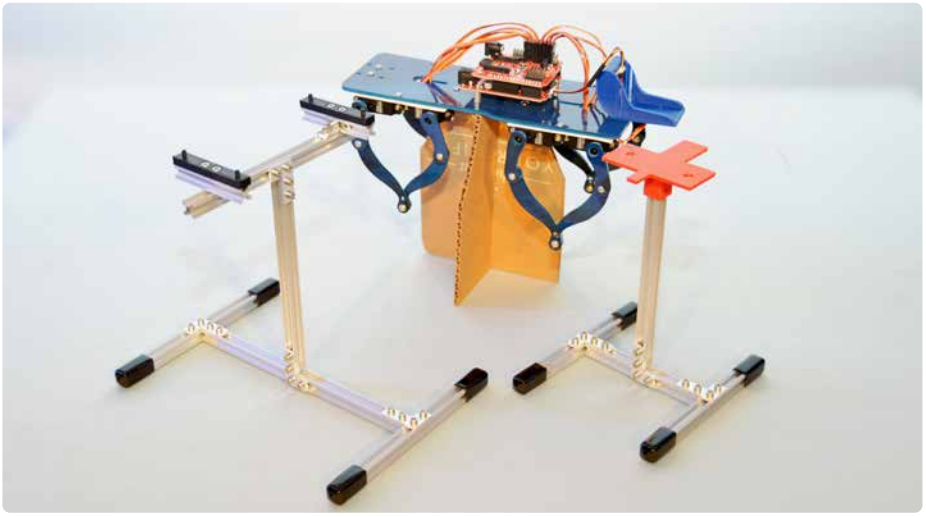


Figure 6. J'ai fabriqué deux supports en alu, mais du carton rigide aurait aussi fait l'affaire.

À vous maintenant de construire le vôtre !

Comme toute étude en cours, ces deux robots vont évoluer et ils auront encore bien changé quand paraîtra cet article. Je dépose tout sur Github (https://github.com/ThingsRobMade/QuadrupedRobots) et poursuivrai la mise à jour au fur et à mesure de l'ajout de fonctions à ces deux quadrupèdes. Je vous encourage à construire les vôtres sans attendre ce que je fais. Lancez-vous et commencez à programmer. L'un des avantages de la communauté open-source est la présence d'une intelligence collective : à chaque obstacle,il se trouvera toujours quelqu'un pour émettre une idée brillante qui à son tour inspirera de nouvelles méthodes ou de nouvelles directions. D'ici là, ne cessez jamais ni de rêver ni d'inventer. Bonne bidouille !

(VF : Denis Lafourcade 200712-02)

LIENS

[1] Technovation, "3D Printed Arduino Powered Quadruped Robot," Instructables, 2020. : https://www.instructables.com/3D-Printed-Arduino-Powered-Quadruped-Robot/

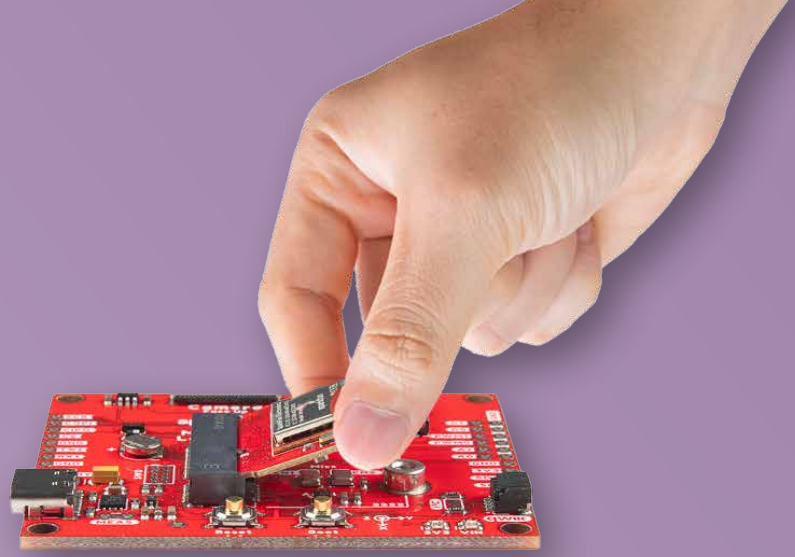
[2] Nate711, "StanfordDoggoProject," GitHub, 2019. : https://github.com/Nate711/StanfordDoggoProject

Listage 1. Place les articulations du robot dans une position neutre et les fait légèrement basculer d'avant en arrière.

```
/*
* Schéma de configuration d'un robot quadrupède
* Rob Reynolds
* SparkFun Electronics
*
* Ce simple croquis fixe les articulations d'un
* robot quadrupède à une position neutre,
* puis les balance légèrement d'avant en arrière
* simplement pour tester le contrôle et le mouvement.
* Actuellement configuré pour 8 degrés de liberté (DoF), mais peut
* facilement être ajusté en modifiant la dimension de
* Servo leg[*] pour correspondre au nombre de servos.
*/

#inclure <Servo.h>
Servo leg [8] ; // créer des objets servo pour commander un servo
int pos = 90 ; // variable pour mémoriser la position du servo

void setup() {
  delay(1000) ;
  for (int l = 0 ; l < 8 ; l++){
    leg[l].attach(l + 2) ; // lier les servos en séquence en commençant par la broche 2
    delay(100) ;
  }
  delay(1000) ;
  for (int l = 0 ; l < 8 ; l++){
    leg[l].write(pos) ; // régler tous les servos à 90
    delay(100) ;
  }
}

void loop() {
  allServoTest() ; // Test initial ou mouvement général des servos
  while(1){
  }
}

void allServoTest(){
  // Test de mouvement initial pour tous les servos. Tous les servos sont réglés à 90°.
  // puis balayage avant et arrière avant de revenir à 90° de nouveau
   for (int i = 0 ; i < 8 ; i++){
    for (pos = 90 ; pos <= 135 ; pos += 1) {
      leg[i].write(pos) ; // dire au servo d'aller à la position de la variable 'pos'.
      delay(10) ; // attendre 10ms que le servo atteigne la position
    }
    for (pos = 135 ; pos >= 45 ; pos -= 1) {
      leg[i].write(pos) ; // dire au servo d'aller à la position dans la variable 'pos'.
      delay(10) ; // attendre 10ms que le servo atteigne la position
    }
    for (pos = 45 ; pos <= 90 ; pos += 1) {
      leg[i].write(pos) ; // dire au servo d'aller à la position dans la variable 'pos'.
      delay(10) ; // attendre 10ms que le servo atteigne la position
    }
  }
}
```

MicroMod

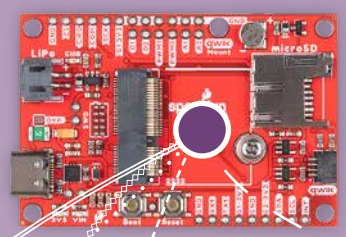
Un **écosystème modulaire** de processeurs et de cartes interchangeables pour accélérer la conception et le prototypage



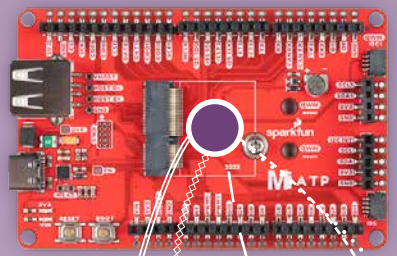
sparkfun
START SOMETHING

Tu choisis tes fonctions ?

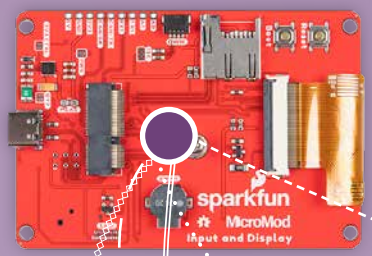
CHOISIS UNE CARTE de SUPPORT



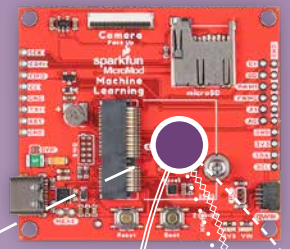
Capture de données



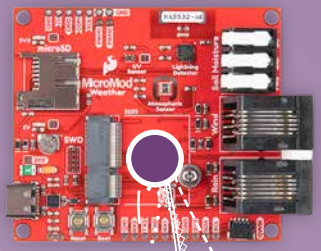
ATP (all the pins)



Visualisation de données et de grandeurs d'entrée

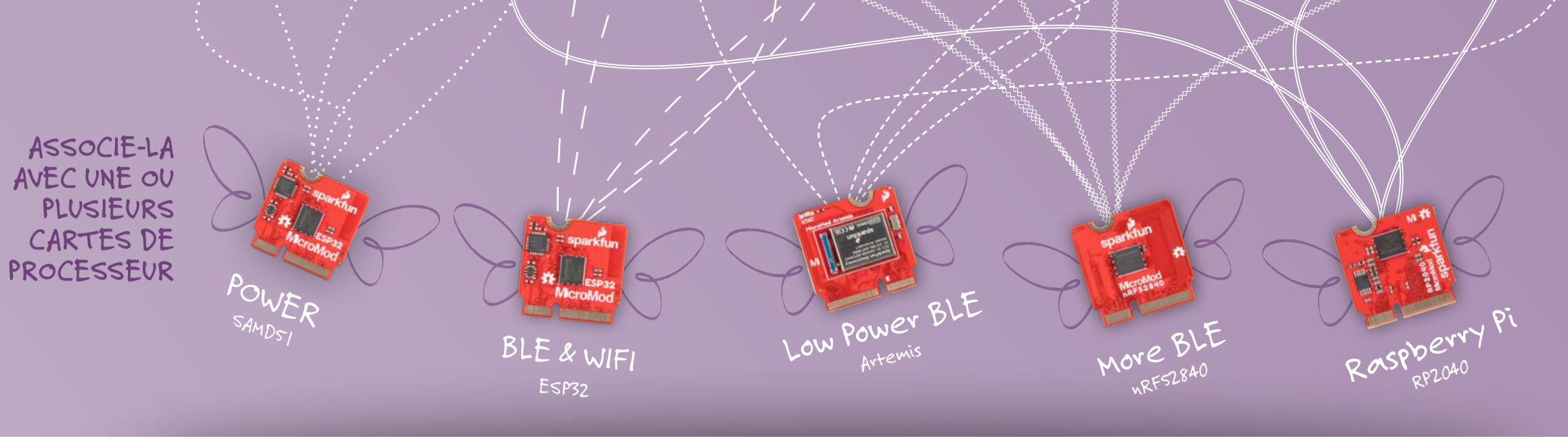


Apprentissage machine



Météo

Carte de support + carte(s) de processeur = prototypage rapide



Y a-t-il des changements dans ton projet ? Pas de problème !

Dans l'écosystème d'interface modulaire MicroMod, une "carte de processeur" de microcontrôleur est reliée à divers périphériques dits "cartes de support". La norme M.2 permet un prototypage et une conception rapides, avec peu ou pas de code en cas de changements.
Tu peux facilement étendre ou modifier les capacités de ton projet en changeant de processeur ou de carte de support, ou en utilisant le système Qwiic Connect. Tu ne trouves pas de combinaison de carte de processeur et carte de support qui te convienne ? Puisque MicroMod est une ligne de produits *open source*, tu peux créer la tienne toi-même ! SparkFun fournit les bibliothèques Eagle et des tutoriels pour te faciliter la conception de ta propre carte.

Informe-toi et commande la tienne !

www.elektormagazine.com/poster-micromod

elektormagazine.com/posters

elektor
design > share > sell

L'internet des objets libéré : RISC-V, AWS et FreeRTOS

Avra Saslow (États-Unis)

Au cours des décennies, la philosophie de l'open source s'est révélée d'une importance considérable dans les communautés logicielles et scientifiques. Le principe est que tout le monde peut contribuer à améliorer un projet/produit, ce qui finit par le rendre plus accessible et plus fiable. En encourageant les experts à travailler ensemble sur un même projet, la technologie open source garantit la qualité et la sécurité et permet in fine d'obtenir un excellent produit sans limitation de propriété. Cet article est un guide pour la mise en œuvre concrète de ces concepts dans une application en temps réel.

Retour sur RISC-V et FreeRTOS

Après des années de succès dans l'industrie du logiciel, l'industrie du matériel informatique dispose enfin d'une technologie *open source*, RISC-V, qui pourrait changer complètement l'avenir des microcontrôleurs. Traditionnellement, l'ISA (architecture du jeu d'instructions) d'une carte décrit la manière dont le logiciel et le matériel communiquent entre eux, et est généralement verrouillée par des licences, des redevances et des NDA (accords de non-divulgation).

Par opposition, l'ISA RISC-V est fournie sous licence open source, même si le SoC (système sur puce) n'améliore pas forcément les performances de calcul (bien qu'il soit encore parmi les plus rapides du marché). L'architecture RISC-V semble ainsi prête à bouleverser complètement le modèle commercial de la technologie. Plutôt que de choisir un ISA et donc de s'enfermer dans la bibliothèque du fournisseur, RISC-V permet aux entreprises de personnaliser, dimensionner et adapter le noyau à leurs besoins spécifiques.

Le projet décrit ici capitalise sur le matériel open source en mettant en œuvre FreeRTOS, qui est un noyau de système d'exploitation en temps réel pour les systèmes embarqués. FreeRTOS permet d'implémenter des solutions flexibles de multithreading ; au lieu d'utiliser des bibliothèques autonomes découplées, FreeRTOS permet à l'utilisateur de maintenir et de porter du code vers différentes applications au fil du temps.

Voyons les choses ainsi… lorsqu'on utilise un microcontrôleur courant comme un Arduino, la structure traditionnelle d'exécution d'un programme est une boucle sans fin qui contient toutes les tâches du système. Lorsque le programme démarre, il effectue une fonction de configuration puis exécute en boucle une série de tâches dans l'attente d'une interruption.

Ces tâches peuvent interroger des capteurs, écrire sur des écrans ou faire des calculs ; peu importe ce qu'elles font, elles le font de manière séquentielle. Cette structure de programme est excellente car elle ne nécessite pas de grosses dépenses d'énergie.

Imaginons que vous deviez effectuer plusieurs tâches simultanément. Même si la structure décrite ci-dessus peut aller vite, elle ne fonctionne pas en parallèle. C'est là qu'un système d'exploitation en temps réel comme FreeRTOS devient utile, car ses temps de latence pour les interruptions et la commutation de fils de traitement (*threads)* sont minimes.

Une latence d'interruption minimale signifie que le système d'exploitation (OS) a optimisé le temps minimum entre les exécutions de tâches ou de sous-programmes. La latence minimale de commutation de fils est le temps minimum nécessaire à l'OS pour commuter le CPU (un seul CPU) afin d'exécuter un autre sous-programme. Ce

Figure 1. Fil unique ou fils multiples, mono-thread vs. multithreading.

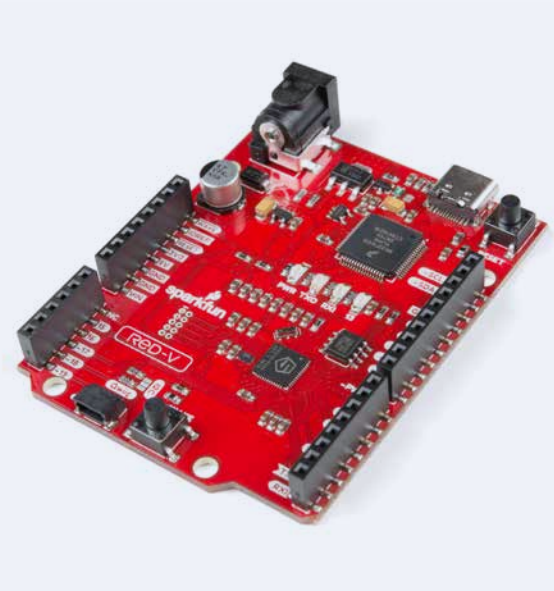


Figure 2. La carte SparkFun RED-V est basée sur une bête de calcul RISC-V.

sont essentiellement ces latences minimales qui font d'un RTOS un OS multitâche, capable d'exécuter des tâches simultanément plutôt que séquentiellement (**fig. 1**).

Bien que vous n'ayez pas besoin d'un RTOS pour les programmes qui interrogent des capteurs, effectuent des calculs sur les données et écrivent sur un écran LCD, le multithreading peut devenir très utile pour des cas comme l'interaction avec les piles logicielles d'un réseau sans fil qui exigent une réponse immédiate aux événements. Spécialement conçu pour les appareils embarqués, FreeRTOS est donc parfaitement adapté pour charger son noyau d'OS sur une carte RISC-V et communiquer dans les AWS (*Amazon Web Services*™) par le biais de bibliothèques sans fil.

Configuration du µC RED-V avec le noyau et les bibliothèques FreeRTOS et déploiement dans le nuage AWS

Pour créer une application utile dans le monde d'aujourd'hui, il faut nécessairement un modèle de base avec un dispositif IdO qui s'appuie sur une plate-forme de services sécurisée du nuage. Cette extension du noyau FreeRTOS tire parti des bibliothèques de protocoles d'application qui fournissent la connectivité nécessaire à la création d'applications IdO à partir de dispositifs à microcontrôleurs, comme la carte SparkFun RED-V (**fig. 2**) construite sur RISC-V. La structure à laquelle nous devons adhérer pour notre application est illustrée sur la **fig. 3**.

Heureusement, FreeRTOS peut être facilement mis en œuvre avec AWS ! Pour configurer une connexion à un serveur HTTP via AWS IoT, vous devez créer un utilisateur dans la section *Identity and Access Management* (IAM) d'AWS. Cet utilisateur aura des autorisations pour accéder à la fois à AmazonFreeRTOSFullAccess et à AWSIoTFullAccess - deux politiques d'accès à AWS. La carte RED-V doit également être enregistrée auprès d'AWS IoT, ce qui implique d'avoir :

- une politique AWS IoT qui accorde à votre appareil des autorisations d'accès aux ressources AWS IoT ;
- un objet AWS IoT qui vous permet de gérer vos appareils dans AWS IoT ;
- une clé privée et un certificat qui permettent à votre appareil de s'authentifier avec AWS IoT.

Tout cela peut sembler lourd, mais on peut se connecter rapidement en utilisant le processus Quick Connect (**fig. 4**) de la console FreeRTOS. Vous pouvez également passer chacune des étapes manuellement via la ligne de commande. La dernière étape pour configurer la carte RED-V en tant que dispositif IoT dans AWS consiste à télécharger FreeRTOS comme OS pour l'architecture RISC-V.

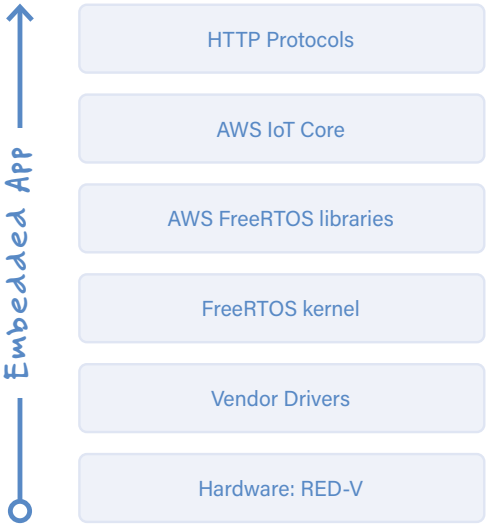


Figure 3. Couches d'application intégrées.

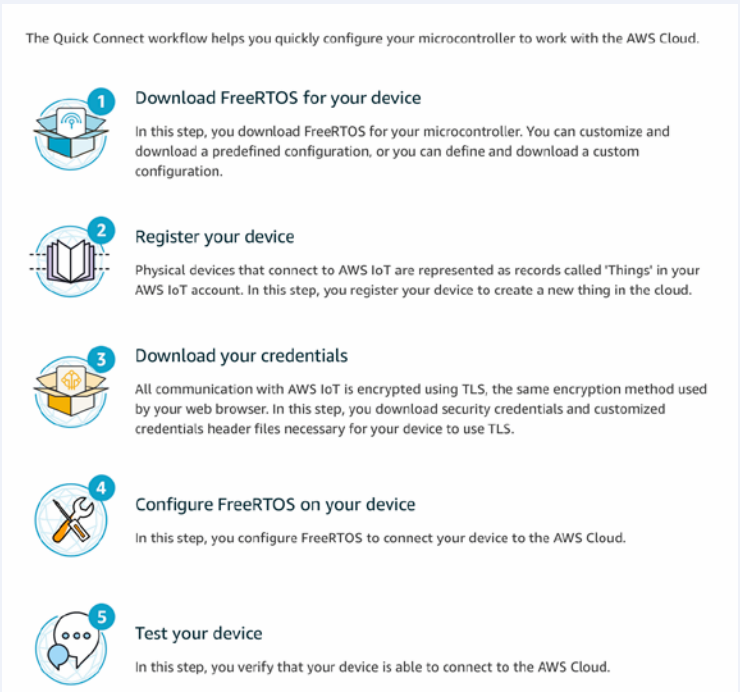


Figure 4. Le processus Quick Connect. (source : AWS)

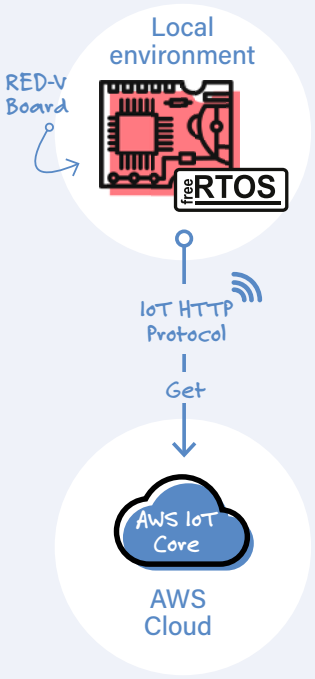


Figure 5. Représentation schématique du processus suivi par notre application IdO-sur-AWS.

Connexion de RED-V comme dispositif IoT au serveur HTTP IoT d'AWS

La configuration est faite, on va s'amuser… et faire une démo HTTP sur la carte RED-V via AWS IoT qui crée une seule tâche d'application. HTTP signifie *HyperText Transfer Protocol*, et HTTPS est crypté avec *Transport Layer Security*, où TLS. HTTP et HTTPS sont des protocoles utilisés pour transférer des données sur le web. Ils utilisent des méthodes de requête spécifiques pour effectuer diverses tâches. Les deux méthodes les plus connues sont GET, qui demande des données à une ressource spécifique, et POST, qui envoie des données à un serveur pour créer ou mettre à jour une ressource. Ce sont les éléments de base de toute application IdO, et cette démo montre qu'on peut tous les déployer via AWS en utilisant du matériel et des logiciels open source.

La démo (**fig. 5**) se connectera au serveur HTTP d'AWS IoT, créera et enverra une requête HTTP, recevra une réponse HTTP, puis se déconnectera du serveur. Grâce à la documentation de FreeRTOS, nous pouvons examiner la structure de l'application à travers le code assez complexe d'AWS [1].

En utilisant la bibliothèque FreeRTOS HTTP Client, le RED-V est maintenant un dispositif IdO capable d'envoyer des requêtes HTTP et de recevoir des réponses HTTP avec la plupart des serveurs HTTP - et pas seulement les serveurs AWS. Par conséquent, il est facilement transposable à d'autres applications dynamiques, car :

- il a des fonctions API à la fois entièrement asynchrones et synchrones ;
- la mémoire est gérée par l'application pour le contexte interne et les en-têtes HTTP ;
- il est *thread-aware* et les connexions sont parallélisées.

Des possibilités infinies

Il est facile de réaliser une application IoT sur l'architecture RISC-V, couplée au noyau et aux bibliothèques FreeRTOS. Elles permettent de personnaliser et dimensionner les dispositifs embarqués pour n'importe quelle application. La configuration de requêtes HTTP via la bibliothèque client est la partie visible de l'iceberg dans la création d'applications IdO. Les possibilités sont infinies pour ces technologies open source !

(200699-02 – VF : Denis Lafourcade)

Espace Elektor en ligne pour cet article
www.elektormagazine.com/esfe-en-redv

Accessoires

Vous trouverez les accessoires mentionnés dans cet article chez SparkFun et chez Elektor !

- **SparkFun RED-V RedBoard - SiFive RISC-V FE310 SoC**
  www.elektormagazine.com/esfe-en-redv1

- **SparkFun RED-V Thing Plus - SiFive RISC-V FE310 SoC**
  www.elektormagazine.com/esfe-en-redv2

LIENS

[1] AWS code: https://www.freertos.org/http/http-demo-with-tls-mutual-authentication.html

L'ÉLECTRONIQUE POUR LE PLAISIR

Conversation entre passionnés

CJ Abate (États-Unis))

La communauté mondiale d'Elektor comprend des électroniciens, des créateurs et des innovateurs aux centres d'intérêt variés, tels les projets pour l'internet des objets à base de µC, la conception de véhicules autonomes ou la programmation embarquée. En interrogeant nos amis de SparkFun sur leurs produits, leur culture d'entreprise et leur communauté, nous avons appris qu'ils travaillent et s'amusent beaucoup !

Comment vos ingénieurs repèrent-ils les nouveautés utilisables dans de nouveaux produits SparkFun ? — Mathias Claussen (Allemagne)

Excellente question ! Nos développeurs consacrent du temps à la veille technologique et au suivi des types de projets sur lesquels travaillent les particuliers, les chercheurs, les ingénieurs et les entreprises. Nous collaborons étroitement avec des fournisseurs et d'autres entreprises pour comprendre comment leurs nouvelles cartes peuvent intéresser nos clients. Nos partenaires distributeurs nous font remonter beaucoup d'info sur la demande de leurs clients. Enfin, beaucoup d'idées naissent des besoins de nos ingénieurs et développeurs de produits pour leurs propres projets personnels. Ils trouvent une solution pour eux-mêmes qui conduit souvent à un nouveau produit SparkFun. Exemple : un ingénieur souhaitait utiliser des relais à semi-conducteurs pour commander un four de refusion fait maison, ça a donné deux nouveaux produits (Qwiic Dual Solid State Relay, www.sparkfun.com/products/16810, et Qwiic Quad Solid State Relay Kit, www.sparkfun.com/products/16833). — **Bryan Hoff (Directeur de l'ingénierie)**

Vos produits sont à la pointe des nouvelles techniques, donc essentiellement numériques, avec des tutoriels, des logiciels et des outils en ligne associés. Comment cela s'équilibre-t-il avec la conception, l'analyse et la construction d'appareils électroniques analogiques ? Y a-t-il des spécialistes chez SparkFun à qui s'adresser pour obtenir des informations et des publications sur l'"analogique", qu'il s'agisse de techniques anciennes ou nouvelles — Jan Buiting (Pays-Bas)

L'accent mis par SparkFun sur le numérique de pointe est lié au fait que la plupart des nouveaux capteurs et circuits intégrés de pointe sont numériques. Beaucoup de ces composants relèvent de tendances telles que l'informatique en nuage, l'apprentissage machine, les appareils mobiles et l'électronique vestimentaire. Nous adhérons à cette tendance générale qui combine faible consommation et faible coût, et abaisse les barrières. Nous donnons à nos clients l'accès à ces techniques. Nous sommes des électroniciens, nous aimons l'analogique, et la plupart d'entre nous ont commencé par l'électronique analogique. Celle-ci se divise en deux : la conception classique de circuits et les mesures industrielles et de construction avancées (que vous mentionnez). L'évolution dans le domaine de l'industrie et de la construction haut de gamme est intéressante à voir, mais son prix est assez élevé ; cela dit, nous restons attentifs à des tendances du marché telles que l'industrie 4.0. Pour le contenu lié à l'analogique, je suggère toujours de consulter Hackaday, qui propose des articles et des projets analogiques intéressants. Et l'IEEE couvre bien des domaines comme les instruments de mesure analogiques de précision. — **Pearce Melcher (Services produits - Chercheur technique)**

Carte de liaison GPS

Des circuits intégrés sont fabriqués en très grandes quantités chaque mois/année. Comment choisissez-vous ceux pour lesquels SparkFun proposera une carte spécifique ? — Clemens Valens (France)

Nous sommes nous-mêmes sidérés de voir apparaître tant de nouvelles techniques. Il y a quelques années à peine, la précision des modules GNSS n'était que de l'ordre du mètre. Aujourd'hui, avec données de correction, nous atteignons le centimètre ! Alors, comment choisir parmi tant d'innovations ? Ce sera toujours un mélange de ce que nous, ingénieurs, trouvons ingénieux, de ce que nos partenaires partagent avec nous, de ce dont nous avons besoin pour nos projets personnels mais peut aussi se transformer en produits vendables, et enfin de certains choix stratégiques. Ceux-ci résultent souvent d'une décision consciente de proposer une bonne, une meilleure voire la meilleure solution technique. Être capable de fournir une bonne solution GNSS (www.sparkfun.com/products/15733) ou un GNSS complet avec navigation à l'estime (www.sparkfun.com/products/16344) offre à nos clients une solution qui répond à leurs besoins et à leur budget. — **Bryan Hoff (Directeur de l'ingénierie)**

EParlez-nous des principaux concurrents de SparkFun (grands et petits). — Don Akkermans (Pays-Bas)

Nous nous concentrons moins sur la concurrence que sur notre communauté et sur les techniques émergentes - ces deux éléments-là guident la plupart de nos actions. Nos ingénieurs sont animés par une intense curiosité pour les nouvelles techniques, ce qui les inspire, comment les utiliser et comment les rendre plus faciles à utiliser pour tout le monde. Nous nous en tenons à cela. Et nous préférons la coopération. Nous avons constaté que, même avec certaines entreprises susceptibles de passer pour concurrentes, nous avons souvent en commun des outils, des compétences, des produits ou une portée complémentaires ... ce qui ne fait qu'augmenter le bénéfice d'une collaboration... bien que nous soyons en situation de concurrence, nous serons, en fin de compte, les uns et les autres, élevés et grandis par cette entente. Depuis le début, nous préconisons la technologie à source ouverte (et nous coopérons étroitement avec l'Open Source Hardware Association) - cela nous pousse non seulement à innover rapidement et ne permet aucune stagnation, mais nous espérons aussi que nos produits seront copiés dans les 12 semaines suivant leur sortie, comme l'a fait remarquer notre fondateur dans l'un des premiers exposés TedEx (http://bit.ly/TEDx-Seidle). Cela se traduit par des défis concurrentiels indéniables. Enfin, nous restons bien sûr attentifs au marché, la concurrence est partout, petite et grande. Qu'il s'agisse de marchés tiers, de grandes entreprises essayant d'atteindre le même marché que nous, ou d'autres équipes d'ingénieurs développant des cartes et des outils de prototypage... nous savons que nos clients ont le choix. Nous faisons de notre mieux pour fournir de la valeur ajoutée (documentation, didacticiels, vidéos, projets, exemples de code...) afin de fidéliser la communauté que nous servons et d'inviter de nouvelles personnes à nous rejoindre. — **Jahnell Pereira (Responsable du développement commercial)**

Votre offre de produits d'IA et d'apprentissage automatique est déjà considérable. Lesquels recommanderiez-vous aux débutants ? — Jens Nickel (Allemagne)

Kit DLI pour Jetson Nano

Pour débuter dans l'apprentissage machine, NVIDIA a mis l'accent sur le développement de points d'entrée et d'outils. Ils ont créé pour cela un cours en ligne gratuit du Deep Learning Institute (DLI) et ont travaillé avec SparkFun pour le kit associé au cours - le DLI Kit for Jetson Nano : www.elektormagazine.com/esfe-en-jetson1. Ce serait un excellent point de départ. — **Derek Runberg (Partenariats stratégiques et éducation)**

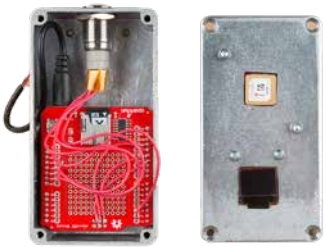
Quels sont vos produits populaires auprès des électroniciens ? Avez-vous une idée des raisons de leur popularité en Europe ? — Muhammed Söküt (Allemagne)

En 2020, environ 30 % de nos activités provenaient de pays autres que les États-Unis. Ce chiffre est en légère baisse, en raison, nous le supposons, de macro-influences comme la COVID-19 et les tarifs. En ce qui concerne l'Europe, certaines de nos familles de produits les plus populaires étaient le GPS/GNSS (en particulier les cartes de développement avec les modules de l'u-blox de Zurich), la RFID, l'IR, le LIDAR et les outils de données comme l'OpenLog – de tels produits sont beaucoup demandés par des passionnés, mais les prototypistes et les chercheurs ont aussi besoin de tels outils qui accélèrent leur progression. En tête des favoris en Europe se trouvent bien sûr ceux de la ligne Raspberry Pi. — **Jahnell Pereira (responsable du développement commercial)**

Avant la COVID-19, avez-vous participé à de nombreux événements de l'industrie électronique aux États-Unis ? Participerez-vous à des événements dans les mois à venir ? — Margriet Debeij (Pays-Bas)

Comment la pandémie affectera-t-elle les événements à long terme ? Avant 2020, notre principal objectif en matière d'événements était de soutenir nos partenaires en organisant des démonstrations sur leur stand, en organisant des présentations/ateliers ou simplement en participant en tant qu'exposant pour rencontrer des partenaires. Voici quelques-uns des endroits où vous pourriez nous trouver : Annual RISC-V Summit, CES, Mobile World Congress, HackaDay's SuperCon, the Arm DevSummit, DEF CON, Open Source Hardware Summit, conférence TensorFlow de Google, conférences sur l'éducation, etc. Depuis la fin de 2020, nous avons assisté à certaines conférences par voie numérique. Nous verrons comment cela évoluera au cours de l'année. — **Hailey Blessing (spécialiste des relations publiques)**

Quels sont les trois principaux projets réalisés avec les produits SparkFun dont vous êtes vraiment fier ? - Udo Bormann (Allemagne)
Pour l'article Rétronique (que je recommande), j'ai demandé à plusieurs personnes leurs projets favoris parmi les plus anciens : le téléphone Port-O-Rotary, le robot perceur de coffres-forts, le scanner de skimmers ont été évoqués (vous pouvez les retrouver dans cette édition). Je suis très fier de notre travail dans le domaine du GPS. Ainsi, Rob Reynolds m'a époustouflé en utilisant le RTK Surveyor pour dessiner sur le sol du parking la flamme du logo de SparkFun à l'aide de SW Maps. J'ai également adoré le tutoriel de Brandon Williams GPS Geo-Mapping at the Push of a Button, qui combine le RedBoard Turbo SAMD21 Development Board, le microSD Shield, le GPS Breakout Chip Antenna SAM-M8Q, et le Micro OLED Breakout pour récolter les coordonnées du monde entier dans Google Earth. Que des trucs marrants ! — **Jahnell Pereira (Directeur du développement commercial)**



Géocartographie par GPS

J'avais participé au concours de véhicules autonomes SparkFun 2013 (AVC) à Boulder. Un grand événement avec des drones étonnants. Quel est l'intérêt pour les drones et les véhicules autonomes ? Prévoyez-vous de nouvelles compétitions une fois que la COVID-19 sera derrière nous ? — C. J. Abate (États-Unis)
Beaucoup de clients construisent des drones et des véhicules autonomes à des fins personnelles et professionnelles. L'AVC a réuni des gens de tous âges, venus du monde entier, pour participer à la compétition. Face à la multiplication et la banalisation de compétitions de ce type, nous avons mis fin à l'événement après une décennie, puis beaucoup réfléchi à l'organisation d'un nouveau concours... la bonne inspiration viendra. Nous renouvellerons l'expérience, mais sous une forme inédite, qui sera un défi pour notre communauté. Si vous ou vos lecteurs avez des idées, faites-le nous savoir. — **Hailey Blessing (spécialiste des relations publiques)**

Véhicule d'un précédent AVC

Il y a quelques années, sur son blog, SparkFun vantait une „culture de travail pas si traditionnelle". L'entreprise accueillait par exemple des chiens au bureau ! Avant la COVID-19, comment vous y preniez-vous pour intégrer vos chiens dans votre espace de travail ? — Denise Bodrone (Pays-Bas)
La culture de SparkFun est extraordinaire à plusieurs égards. Ainsi tout employé est autorisé à amener un chien au travail à condition de respecter notre politique canine. Les règles, établies en accord avec les concernés, leur imposent la responsabilité de veiller à ce qu'elles soient respectées par nos amis à fourrure. Pour avoir son chien sur place, tout propriétaire de chien accepte de participer et d'être responsable devant notre tribunal canin. Celui-ci est composé de cinq propriétaires de chiens sélectionnés chaque mois au hasard, et examine toutes les affaires canines récentes, discute des compliments ou des plaintes reçues, prend une décision sur la responsabilité en fonction de notre règlement et de la jurisprudence, et discute avec les propriétaires de chiens des suites. Cette solution intéressante, affinée au fil des ans, est loin de la perfection, mais fonctionne assez bien pour nous permettre d'avoir quelque chose comme 30 chiens sur place n'importe quel jour de la semaine ! — **Kristen Moorefield (Directrice des opérations)**

Beaucoup d'entre nous s'efforcent de ne pas maltraiter notre planète. Comment s'y prend SparkFun pour concevoir des produits durables et respectueux de l'environnement ? Dans quelle mesure évitez-vous le plastique ? — Mathias Claussen (Allemagne)
Nous nous préoccupons de l'impact écologique de ce que nous produisons et nous nous efforçons de supprimer les déchets. Notre siège, construit en 2012, est le premier bâtiment du comté de Boulder construit selon le Code international de construction écologique. Notre équipe de développement durable se réunit une fois par semaine pour trier notre flux de déchets difficiles à recycler, en s'efforçant de détourner autant de matériaux que possible de la décharge - c'est ainsi que nous traitons les plastiques difficiles à recycler. Nous utilisons des prises de courant intelligentes, nous participons à des programmes de recyclage des déchets électroniques, utilisons des vélos communautaires, et notre installation photovoltaïque sur le toit fournit environ 30 % de notre électricité mensuelle. Dans tous les départements, nous faisons de notre mieux pour réduire notre empreinte écologique. — **Nick Beni (responsable des installations)**

Nick et son prix du recycleur de l'année

Comment décidez-vous si un produit convient à la boutique SparkFun ? Quel est votre produit le plus vendu ? – Luc Lemmens (Pays-Bas)
Si l'on fait le bilan des 18 dernières années, le produit le plus vendu est le SparkFun Inventors Kit ou ses dérivés. L'un de nos points forts est d'être toujours disposés à expérimenter - en suivant nos intérêts et notre inspiration, ou ce que nous trouvons utile. — **Jordan Colby (Responsable de l'analyse des données)**

200688-03

Votre sélection de produits

sparkfun ELECTRONICS

Depuis 2003, SparkFun vous aide à réaliser vos idées, qu'il s'agisse de créer une station météo intelligente, d'explorer les frontières de l'apprentissage automatique, de construire un robot pour l'enseignement ou de prototyper votre premier (ou votre dixième) produit. Quelles que soient votre vision ou vos compétences, les composants, les ressources et didacticiels en ligne de Sparkfun élargissent votre accès aux techniques innovatrices et raccourcissent le chemin qui mène au projet fini.
Voici une sélection de produits recommandés aux lecteurs d'Elektor.

SparkFun Inventor's Kit - V4.1

Le kit de l'inventeur de SparkFun (SIK) est un excellent moyen de se lancer dans la programmation et l'interaction matérielle avec le langage de programmation Arduino. Le SIK comprend tout ce dont vous avez besoin pour réaliser cinq projets globaux consistant en 16 circuits interconnectés qui enseignent tout, du clignotement d'une LED à la lecture de capteurs. Le projet culminant est votre propre robot autonome ! Aucune expérience préalable en programmation ou en électronique n'est requise pour utiliser ce kit.

www.sparkfun.com/products/15267

MicroMod DIY Carrier Kit de SparkFun (paquet de 5)

www.sparkfun.com/products/16549

Qwiic pHAT v2.0 de SparkFun pour Raspberry Pi

www.sparkfun.com/products/15945

Kit de capteurs SparkFun

Si vous avez une bonne connaissance générale des capteurs ou besoin de capteurs variés pour un projet, vous trouverez dans ce kit un capteur pour presque tous vos travaux. Vous détecterez par exemple les gestes, l'humidité, la température, le mouvement, le toucher, le son, l'altitude, l'accélération et bien plus !

www.sparkfun.com/products/16156

SparkFun Servo pHAT pour Raspberry Pi

www.sparkfun.com/products/15316

RED-V RedBoard - SiFive RISC-V FE310 SoC de SparkFun

Le RED-V (prononcer *red-five*) RedBoard est une carte de développement très peu chère, équipée du SoC Freedom E310 et son architecture du jeu d'instructions (ISA) RISC-V. Ce qui distingue la RedBoard RED-V est son approche totalement *open source* du matériel de l'ISA.

www.sparkfun.com/products/15594

RED-V Thing Plus - SiFive RISC-V FE310 SoC de SparkFun

www.sparkfun.com/products/15799

Kit de câbles Qwiic de SparkFun

La popularité du système de connexion Qwiic de SparkFun augmente. Ce kit de câbles Qwiic facilitera encore plus son utilisation pour tous les électroniciens.
www.sparkfun.com/products/15081

Qwiic GPIO de SparkFun

Quand il vous manque juste quelques broches GPIO disponibles, quand peut-être une ou deux broches libres feraient déjà l'affaire, faut-il rajouter tout un microcontrôleur à votre projet ? Ça complique et ça coûte... Et si vous utilisiez le GPIO Qwiic pour ajouter huit broches à votre projet Qwiic ?
www.sparkfun.com/products/17047

Adaptateur Qwiic SparkFun

L'adaptateur Qwiic SparkFun est le moyen idéal pour transformer n'importe quelle ancienne carte I²C en carte Qwiic.
www.sparkfun.com/products/14495

Auto pHAT de SparkFun pour Raspberry Pi

www.sparkfun.com/products/16328

ESP32 Module WROOM MCU – 16 Mo (Chip Antenna)

www.sparkfun.com/products/16281

Carte de distribution d'énergie USB-C de SparkFun (Qwiic)

Maîtrisez la distribution d'énergie de vos réalisations avec ce Power Delivery Board - USB-C de SparkFun ! La plage d'intensité du courant des modules d'alimentation conventionnels est assez étendue, mais leur tension reste fixe à 5 V.

www.sparkfun.com/products/15801

SparkFun GPS Breakout - Chip Antenna, SAM-M8Q (Qwiic)

La carte de liaison GPS SAM-M8Q de SparkFun offre un GPS de qualité, aux impressionnantes options de configuration.

www.sparkfun.com/products/15210

Top pHAT de SparkFun pour Raspberry Pi

www.sparkfun.com/products/16653

Kit de base de SparkFun pour Raspberry Pi 4 – 4 Go

www.sparkfun.com/products/16384

Bouton Qwiic - LED rouge

Ces boutons sont un moyen tactile commode pour interfacer vos réalisations. Grâce au système Qwiic Connect, l'utilisation d'un bouton se résume à brancher un câble et charger un code existant !
www.sparkfun.com/products/15932

Câble Qwiic – 100 mm

Câble à 4 conducteurs de 100 mm avec terminaison JST de 1 mm, conçu pour interconnecter des composants compatibles avec Qwiic, utilisable pour d'autres applications.
www.sparkfun.com/products/14427

Câble Qwiic – 50 mm

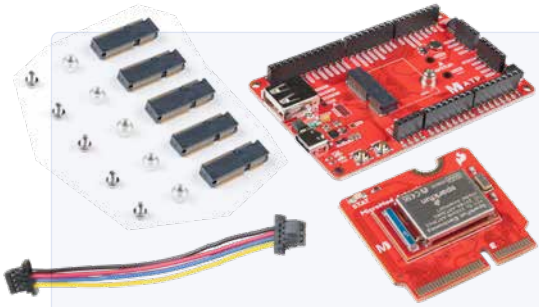
Câble à 4 conducteurs de 50 mm avec terminaison JST de 1 mm, conçu pour interconnecter des composants compatibles avec Qwiic, utilisable pour d'autres applications.
www.sparkfun.com/products/14426

Hexadoku casse-tête pour elektroniciens

Voici une grille de sudoku particulière. Éteignez le fer à souder, oubliez les écrans , rangez les pinces et les fils. Faites travailler votre matière grise d'une autre façon : attrapez un crayon pour remplir la grille d'hexadoku. N'oubliez pas d'envoyer votre réponse pour tenter de gagner l'un des 5 ensembles MicroMod.

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras). Certains chiffres, déjà placés dans la grille, en définissent la situation de départ.

Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.



Résolvez la grille HEXADOKU et gagnez !

Nous tirerons au sort 5 bonnes réponses reçues dans les délais et les heureux gagnants recevront chacun un ensemble SparkFun MicroMod contenant : 1 Artemis processor module, 1 ATP carrier board, 1 Qwiic cable 50mm, 1 Qwiic cable 100mm, 1 SparkFun MicroMod DIY Carrier Kit.

Où envoyer votre réponse ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **10 avril 2021** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de novembre/décembre 2020 est **F1235.**

Les cinq bons Elektor d'une valeur de **50 €** vont à Huub Liebregts (Pays-Bas) – G.D. (Joe) Young (Canada) – Jean-Claude Carré (France) – Ralf Boldt Velbert (Allemagne) – R. Torfs (Belgique).

Bravo à tous les participants et félicitations aux gagnants !

		C			8	0		7	E				B	5	
B		1				2				4	6				F
2						6	7					A	D		0
	5					C	D			0	1	E	2		
D	9	F			E	3				5	8			6	C
		E	5	6				9			A	F			
8				C				B	D	3					
C				8	D			2		E	F			9	
	C			7	3		A			6	D				E
					0	F	8				3				5
			1	2			6				5	8	A		
A	7			D	5				8	1			6	C	4
		9	8	A	4			1	B					3	
7		A	0					D	C						1
F				0	1				7				8		D
	3	B				D	2		5	8			0		

F	1	6	B	4	5	7	C	9	0	8	D	2	3	A	E
E	9	7	2	F	0	A	B	C	1	6	3	5	4	D	8
3	C	0	4	1	6	8	D	A	E	2	5	F	9	7	B
5	D	8	A	E	9	2	3	7	4	B	F	6	0	C	1
A	E	9	6	D	F	0	8	2	5	1	B	4	C	3	7
7	B	D	0	9	A	3	1	4	C	E	6	8	5	F	2
C	2	F	1	6	4	5	E	8	D	3	7	9	A	B	0
8	3	4	5	2	B	C	7	F	9	A	0	D	1	E	6
4	A	B	C	5	1	9	F	3	6	7	8	E	2	0	D
D	5	1	9	7	8	4	0	E	F	C	2	A	B	6	3
0	F	2	E	3	C	B	6	5	A	D	1	7	8	9	4
6	8	3	7	A	D	E	2	0	B	4	9	1	F	5	C
1	4	5	D	0	E	6	9	B	2	F	C	3	7	8	A
9	0	C	3	8	2	1	A	D	7	5	E	B	6	4	F
2	7	A	F	B	3	D	5	6	8	0	4	C	E	1	9
B	6	E	8	C	7	F	4	1	3	9	A	0	D	2	5

Tout recours est exclu, de même que le sont, de ce jeu, les personnels d'Elektor International Media et leur famille. Un seul gagnant par foyer.

Rejoignez les électroniciens de la communauté Elektor

Devenez membre elektor GOLD membership maintenant !

- accès à l'archive numérique depuis 1978 !
- 6x magazine imprimé Elektor
- 9x magazine numérique (PDF) dont Elektor Industry (EN)
- 10 % de remise dans l'e-choppe et des offres exclusives pourles membres
- le DVD annuel d'Elektor
- accès à plus de 1000 fichiers Gerber, collaboration avec les milliers d'électroniciens d'Elektor LAB, et une ligne directe avec nos experts !
- possibilité de voir votre projet publié ou vendu par notre boutique en ligne

---

Également disponible

abonnement elektor GREEN membership sans papier !

- accès à l'archive numérique d'Elektor
- 10 % de remise dans l'e-choppe
- 6x magazine Elektor (PDF)
- offres exclusives
- accès à plus de 1000 fichiers Gerber

www.elektormagazine.fr/membres

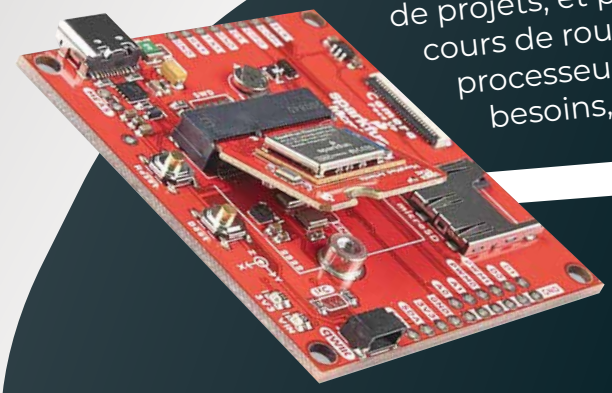
elektor
créer > partager > vendre

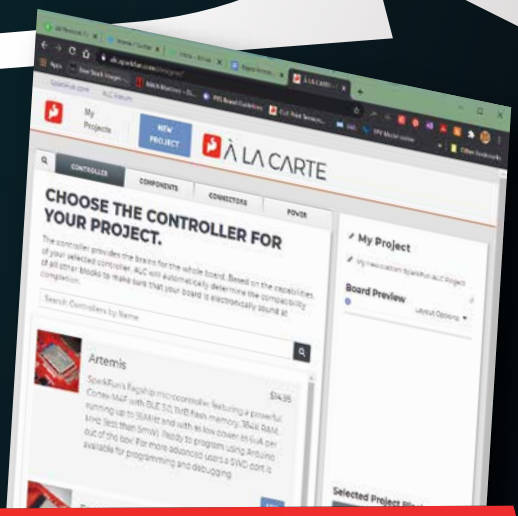
LE PROTOTYPAGE RAPIDE, C'EST FACILE

De la pile de composants en vrac jusqu'au produit fini, SparkFun t'assiste dans ton parcours de prototypage

QWIIC accélère le processus de prototypage grâce à des connecteurs JST à 4 broches pour interfacer les cartes de conception avec des capteurs, des afficheurs LCD, des relais etc. Ajoute de nouveaux composants à ton projet ou change facilement de composant comme bon te semble !

La ligne **MICROMOD** facilite la réalisation et la mise à jour de projets, et permet d'en modifier des fonctions en cours de route. Change de carte de support ou de processeur en fonction de l'évolution de tes besoins, presque sans changer de code.





En comblant le vide entre prototype et production, **À LA CARTE** (ALC) de SparkFun t'aide à créer tes cartes personnalisées ! ALC élabore ta carte en fonction des composants choisis par toi, de sorte que chacun obtienne facilement ce dont il a exactement besoin.

INFORME-TOI !

www.sparkfun.com/qwiic

www.sparkfun.com/micromod

www.alc.sparkfun.com

sparkfun®
START SOMETHING