



LET'S STEAM X MAGNETICS X THEDEXTERLAB

fiches d'activité
Coder avec makecode.lets-steam.eu

Crédits et contributions

Contributeurs

Le L.A.B remercie tous les contributeurs qui ont participé à l'émergence des 16 fiches d'activité et des propositions de protocoles présentées dans ce guide. Ce guide a été créé sur la base du travail des consortia des projets Let's STEAM, TheDexterLab et magnetics. Découvrez nos partenaires et nos ressources complètes :

- Let's STEAM : <https://www.lets-steam.eu/>
- magnetics : <https://www.magnetics.edu-up.fr/>
- TheDexterLab : <http://www.thedexterlab.eu/>

Crédits

Captures d'écran réalisées par les auteurs sur <makecode.lets-steam.eu>. Icônes réalisées par Freepik à partir de <www.flaticon.com>.

Licence et droits

Réalisé avec le soutien du Ministère de l'Education Nationale et du dispositif Edu-up (magnetics), ainsi qu'avec le soutien du programme Européen Erasmus + (Let's STEAM, TheDexterLab). Cette publication n'engage que ses auteurs et ni la Commission Européenne ni le Ministère de l'Education Nationale ne peuvent être tenus responsables de l'usage qui pourrait être fait des informations qu'elle contient. L'ensemble des productions est soumis à une licence Creative Commons Attribution-ShareAlike 4.0 International, qui permet une utilisation, distribution et reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée l'auteur et la source, de fournir un lien vers la licence Creative Commons ainsi que d'indiquer si des modifications ont été apportées et de les partager de la même manière.



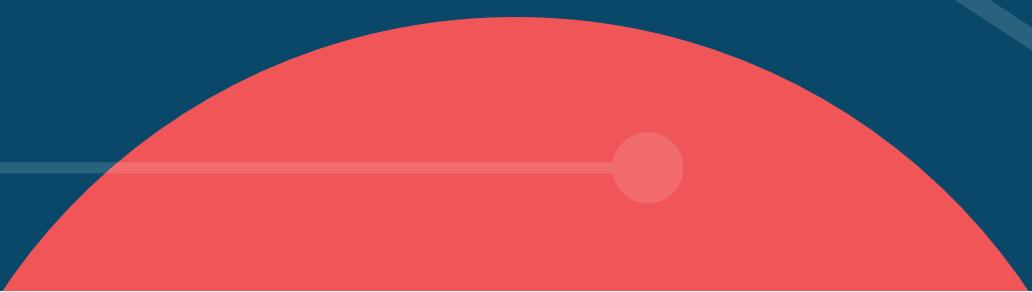
Cofinancé par le programme Erasmus+ de l'Union européenne



partie 1

Les projets

Introduction



Le projet TheDexterLab



Le projet TheDexterLab vise à développer des solutions simples et abordables afin de maintenir des activités d'expérimentation scientifique à l'intérieur et à l'extérieur de la classe en stimulant la créativité des élèves et l'utilisation d'outils numériques. L'accès à l'instrumentation scientifique dans l'enseignement secondaire est souvent entravé par l'inégale disponibilité des ressources dans le monde de l'éducation entre les pays et les écoles d'Europe. Cela conduit à une faible diversité des outils d'expérimentation disponibles, compliquant la possibilité pour les enseignants de développer des activités STEAM (Science, Technologie, Ingénierie, Arts, Mathématiques) à grande échelle, dans une logique interdisciplinaire et axée sur une meilleure compréhension par les élèves des défis sociétaux de notre siècle, illustrant le lien entre les questions scientifiques et techniques et leur impact sur la vie quotidienne des citoyens.

Le projet a été conçu pour répondre à ce défi à court et à long terme en créant plusieurs ressources associées à des protocoles d'expérimentation scientifique, permettant de maintenir et de stimuler l'utilisation de pédagogies actives dans l'enseignement secondaire. Ces ressources comprennent :

- Le développement d'instruments scientifiques de collecte et d'analyse de données, ou dispositifs DIY, accessibles à tous à faible coût, modulaires et réutilisables, conçus à travers une approche Do-It-Yourself pour être facilement reproductibles à l'école, au sein de fablabs ou à la maison.
- Le développement d'outils de simulation, permettant de pallier le manque de ressources financières et de maintenir les activités scientifiques sans dégradation, que le matériel soit disponible pour l'élève/la classe, avec l'objectif sous-jacent de stimuler les compétences en programmation numérique des apprenants.
- Enfin, une documentation de soutien et des lignes directrices permettant la bonne compréhension et la mise en œuvre des protocoles TheDexterLab dans des conditions idéales, au sein d'écosystèmes d'apprentissage formels et informels.

Pour en savoir plus : www.thedexterlab.eu

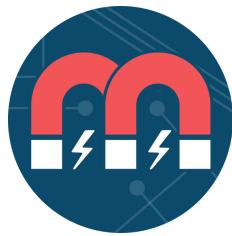
magnetics – un projet inscrit dans un maillage d'initiatives

au service de la pensée computationnelle et de la créativité

Le projet magnetics est une initiative créée grâce au soutien du dispositif edu-up et coordonnée par le Laboratoire d'Aix-périsperimentation et de Bidouille (fablab d'Aix-en-Provence) avec pour objectif de construire une brique logicielle permettant de créer un maillage numérique au service de l'interopérabilité matérielle et logicielle des outils dédiés à la programmation à l'école. Au travers du dispositif edu-up, le Ministère de l'Éducation Nationale et de la Jeunesse soutient le prototypage de solutions numériques innovantes et adaptées. Ces solutions contribuent au maintien de la continuité pédagogique dans le respect de la liberté pédagogique de chacun. Elles prennent en compte les besoins de tous les élèves et répondent aux exigences de l'école inclusive.

Par la mise en œuvre du projet magnetics, l'équipe du L.A.B souhaite participer à la promotion de l'utilisation d'outils numériques et de la programmation au service de la mise en œuvre d'expérimentations stimulantes et créatives dans le domaine de l'enseignement STEAM (Sciences, Technologie, Ingénierie, Arts et Mathématiques).

Il vise à renforcer l'intégration de la programmation au sein des classes, en donnant accès aux enseignants et aux élèves à des outils de collecte de données adaptés aux besoins éducatifs. Grâce à la diffusion des ressources magnetics, adossées à d'autres initiatives promouvant la programmation à l'école, le projet souhaite participer au développement des compétences des enseignants et des élèves dans le domaine de l'informatique et de la production numérique.



suite - magnetics

Le projet magnetics prend principalement la forme d'une brique technique logicielle, implantée dans les plateformes d'apprentissage de la programmation les plus populaires : Scratch, MakeCode et CircuitPython. Elle a pour fonctionnalité de permettre à l'utilisateur de construire des projets multicartes, afin de collecter un ensemble plus large de données sur un terrain d'expérimentation étendu, sans avoir à connecter l'ensemble des cartes entre elles. Par exemple, une carte unique peut jouer le rôle de collecteur de données tandis que les autres cartes ne sont qu'émettrices, et communiquent leurs informations sans connaissance préalable sur la typologie du réseau ni même sur le routage sous-jacent.

Ce développement est basé sur l'utilisation de la technologie de réseau maillé Bluetooth Low Energy Mesh (BLE Mesh) compatible avec toutes les cartes programmables disposant d'un module Bluetooth Low Energy. D'un point de vue pédagogique, l'accès à la brique magnetics permet aux enseignants qui l'utilisent de pouvoir lancer des expérimentations de plus grande envergure tout en gardant un environnement logiciel et matériel connu et maîtrisé.

L'objectif de la solution magnetics est de permettre à tout enseignant sans compétence particulière en développement logiciel ou en électronique de créer une solution IoT pour ses besoins pédagogiques. L'équipement doit pouvoir oublier au maximum les aspects techniques pour permettre à l'utilisateur de se concentrer sur les usages qu'il veut démontrer. Simplicité, minimalisme et automatisation maximale ont été au cœur des étapes de développement de ce premier prototype.

Le prototype magnetics est aujourd'hui disponible gratuitement, couvert par la licence ouverte Creative Commons BY SA. Il a été implémenté dans les plateformes MakeCode (<https://makecode.lets-steam.eu/>) et Micropython (<https://python.lets-steam.eu/>) et l'ensemble des codes sources sont disponibles pour tous sur GitHub.

En savoir plus : <https://www.magnetics.edu-up.fr/>

Dispositif edu-up : <https://eduscol.education.fr/1603/le-dispositif-edu>



Le projet Let's STEAM

En 2016, STMicroelectronics en partenariat avec AMU et le L.A.B ont lancé l'initiative STM32 pour l'éducation, visant à rendre les microcontrôleurs produits par ST utilisables à l'école. Durant ce projet, nous avons rassemblé une trentaine d'enseignants afin de définir leurs besoins et essayer de trouver des solutions adaptées aux enjeux de la classe. En quelques mois, nous nous sommes aperçus que les enseignants non technophiles avaient quitté l'initiative, car ils ne se sentaient pas légitimes face à ces outils malgré leur potentiel. Fort de cette observation, AMU et le L.A.B ont contacté plusieurs de leurs partenaires historiques ainsi que des acteurs de la scène européenne des STEAM, afin de travailler ensemble à des contenus de formation pour les enseignants adaptés à tous dans l'optique de stimuler la créativité et d'aborder les enjeux liés à la pensée computationnelle. La réunion de ces acteurs a permis de créer l'initiative Let's STEAM, rassemblant des visions complémentaires entre le monde de la recherche, de l'électronique, de la programmation et de l'enseignement en milieu scolaire.

A travers la mise en œuvre de Let's STEAM, nous souhaitions apporter de nouvelles compétences ou approfondir les connaissances des enseignants dans le domaine des technologies techno créatives, afin de promouvoir des approches STEAM dynamiques et engageantes pour les professeurs comme pour les élèves.

Grâce à une approche transdisciplinaire de l'utilisation des cartes programmables, le programme de formation Let's STEAM souhaitait promouvoir des pédagogies actives et innovantes, en particulier les approches par enquête et les démarches expérimentales. Le programme de formation s'est articulé autour de 3 thèmes : la programmation au service de la créativité, les démarches par enquêtes et la création d'activités technocréatives inclusives. La totalité du programme a été consolidée sous forme d'un manuel de cours, utilisable pour former les professeurs, ou directement comme ressources documentaires dans la classe.

Pour soutenir ces objectifs, un enjeu complémentaire du projet était de développer des outils au service de la programmation porteuse de sens, adaptés à l'environnement de la classe et aux contraintes des enseignants.

Ces outils sont disponibles gratuitement ici :

- **Manuel :** <https://bit.ly/manuel-lets-steam>
- **Plateforme MakeCode:** <https://makecode.lets-steam.eu/>

En savoir plus : www.lets-steam.eu

partie 2

**makecode, stm32,
STeaMi**

les outils



MakeCode

À propos du codage par blocs et de Microsoft MakeCode

En 1975, Seymour Papert, du MIT Media Lab, a créé un langage de programmation pour débutants appelé LOGO. Il l'a développé sur la base de recherches qui ont montré que jouer avec des blocs de code était un moyen particulièrement efficace d'enseigner les concepts de programmation.

Papert a inventé le terme "constructionnisme" pour décrire la manière dont les apprenants construisent de nouvelles connaissances en s'appuyant sur des connaissances établies.

Les blocs de MakeCode sont eux-mêmes des modèles de la manière dont un nouvel apprentissage se produit par l'application de concepts dans un environnement d'apprentissage ouvert.

Les langages de programmation basés sur des blocs tels que Scratch et MakeCode s'appuient sur les recherches de Papert et constituent un excellent moyen pour les élèves de commencer à apprendre les concepts de codage sans avoir à se soucier des questions de syntaxe et des problèmes techniques.

Recherche

ENTRÉE

BROCHES

BOUCLES

LOGIQUE

VARIABLES

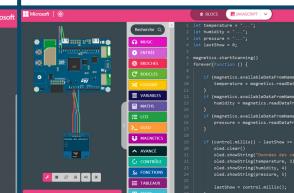
MATHS

AVANCÉ

Présentation de MakeCode

Microsoft MakeCode est une plateforme gratuite et open source permettant de créer des expériences d'apprentissage de l'informatique qui favorisent la progression vers la programmation dans le monde réel. Elle donne vie à l'informatique pour tous les élèves avec des projets amusants, des résultats immédiats et des éditeurs de blocs et de texte pour les apprenants de différents niveaux. Microsoft MakeCode est un projet conjoint entre Microsoft Research et Visual Studio qui vise à simplifier la programmation de dispositifs à base de microcontrôleurs à l'aide d'une application Web moderne. L'outil permet aux utilisateurs de découvrir la programmation en utilisant des "blocs" pour représenter les instructions et les structures de contrôle. Il est également possible de convertir les blocs en langage JavaScript ou Python (vice-versa), ce qui permet de découvrir et d'apprendre progressivement la programmation. Enfin, un simulateur permet de tester le code sans avoir recours à une carte électronique physique (elle-même, ainsi que les capteurs/actionneurs, leur comportement et les connexions, sont simulés).

MakeCode est basé sur les outils suivants :

SIMULATEUR	ÉDITEUR PAR BLOC	ÉDITEUR JAVASCRIPT
		

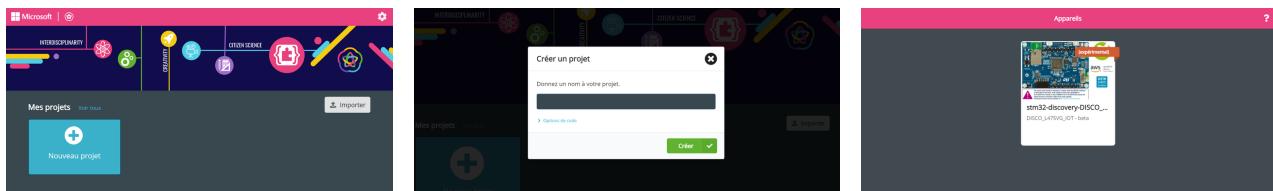
L'éditeur MakeCode offre un écosystème pertinent de par son accessibilité, son interface, la possibilité de programmer en 3 langages différents (donc adaptable à plusieurs niveaux du début du collège au lycée), sa compatibilité multicarte, sa capacité à fournir une simulation visuelle des activités du capteur (intéressant notamment dans le cas d'un enseignement à distance, ou d'un manque de ressources pour équiper les élèves de cartes individuelles), sa compatibilité RGPD, et surtout, la possibilité de développer des projets simples à complexes, pouvant ou non communiquer avec le terrain.

Les développements Magnetics ont été directement intégrés à un éditeur MakeCode spécifique développé lors du projet Let's STEAM et accessible ici : <https://makecode.lets-steam.eu/>. L'utilisation de cet éditeur particulier avait pour ambition d'intégrer le projet magnetics à une ressource conçue pour l'enseignement, comprenant d'autres fonctionnalités pertinentes.



Commencer à utiliser MakeCode

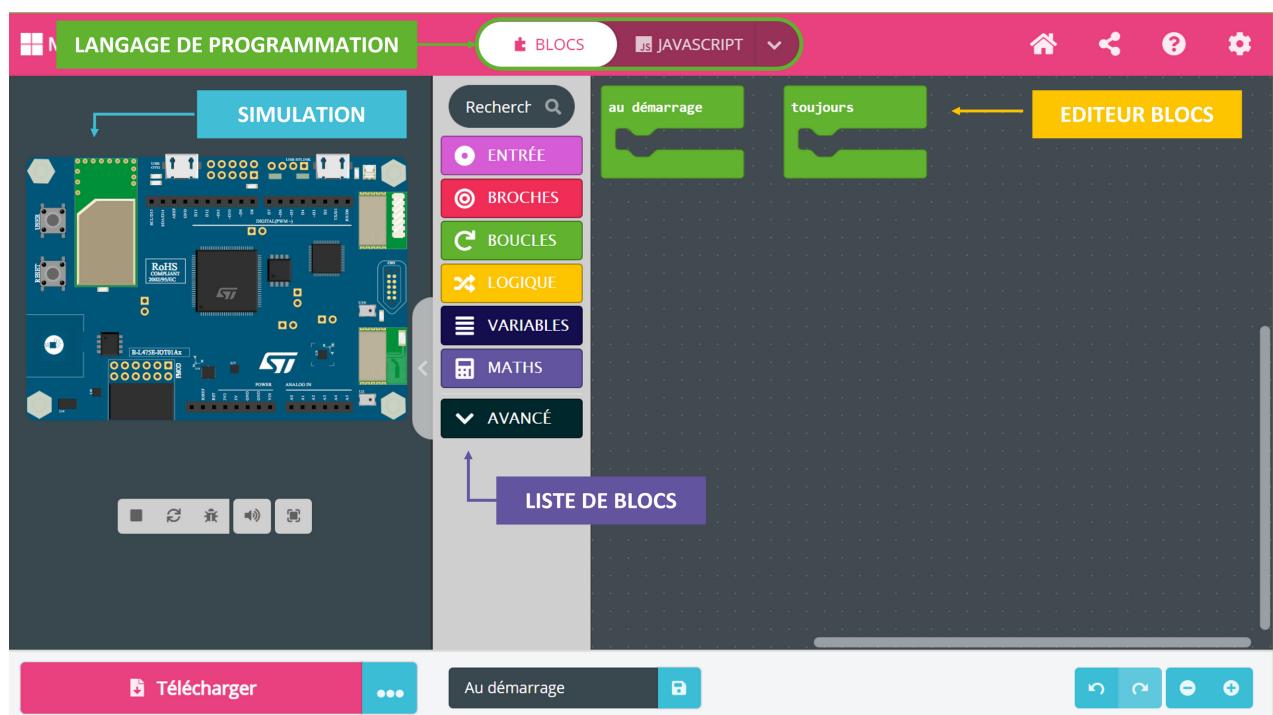
Lorsque vous entrez sur l'interface MakeCode de Let's STEAM, vous arrivez directement sur la page d'accueil. Sur cette page, vous pouvez créer un nouveau projet, ouvrir un projet existant si vous avez déjà travaillé sur l'éditeur, voir les cartes supportées et découvrir des ressources inspirantes. Lorsque vous créez un projet, il est important de le nommer avec un titre clair et compréhensible, vous permettant d'annoncer clairement l'objectif du programme. Sur l'écran suivant, vous devrez choisir la carte sur laquelle vous allez travailler. Sur les fiches d'activités de Let's STEAM et magnetics, tous les exemples ont été développés à l'aide de la carte STM32 IoT Node.



Si l'interface chargée est affichée en anglais au lancement, vous pouvez changer la langue en cliquant sur le bouton "paramètre" afin de voir les versions supportées.



Une fois la carte sélectionnée, vous aurez alors accès à l'éditeur, présenté ci-dessous :



Voici les composants de l'éditeur :

Le SIMULATEUR (à gauche de l'éditeur)

Un simulateur interactif fournit aux élèves un retour immédiat sur comment leur programme fonctionne et leur permet de tester et de déboguer leur code.

La LISTE DES BLOCS au centre

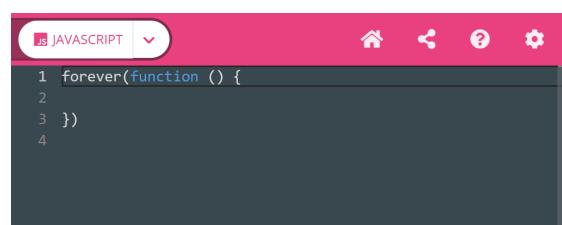
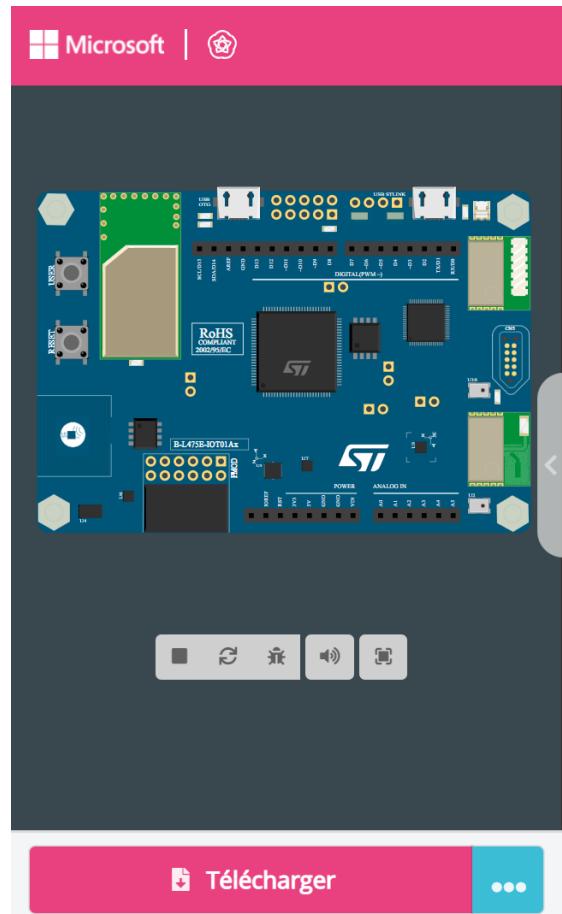
pouvant être utilisés dans votre programme, ainsi qu'un champ de recherche.

L'ÉDITEUR DE BLOCS sur la partie droite

qui comprend déjà deux fonctions communes à toutes les activités : "on start" et "forever loop". Les élèves qui débutent dans le programmation peuvent commencer par des blocs de couleur qu'ils peuvent glisser et déposer sur leur espace de travail pour construire leurs programmes.

Dans l'éditeur, vous pourrez également choisir le mode de programmation :

- Par le biais de blocs (voir la fiche d'activité RIASI - Faire clignoter une LED)
- Par l'intermédiaire d'un éditeur JavaScript
- Par le langage Python pour les élèves plus avancés.



Voici la liste des blocs de base disponibles sur l'éditeur MakeCode de Let's STEAM. Vous aurez un aperçu plus précis de la fonction de chaque bloc dans les diverses fiches d'activités proposées dans ce manuel :

Entrée		ENTRÉE	Utiliser un capteur dans votre programme (comme un bouton, un thermomètre, etc.).
Broches		BROCHES	Interagir directement avec les broches (fréquemment appelées pin) et modifier leur état
Control		CONTROLE	Gérer l'exécution des événements
Boucles		BOUCLES	Mettre en œuvre les répétitions
Logique		LOGIQUE	Effectuer des tests, des comparaisons et des opérations de logique booléenne.
Variables		VARIABLES	Créer des variables et des compteurs
Maths		MATHS	Effectuer divers calculs mathématiques
Fonctions		FONCTIONS	Créer des sous-programmes
Tableaux		TABLEAUX	Créer une valeur ou un texte dans un tableau
Texte		TEXTE	Modifier les textes
Console		CONSOLE	Afficher les données
Extensions		EXTENSIONS	Accéder à la liste des extensions disponibles dans la version de MakeCode
Magnetics		MAGNETICS	Gérer les communications
Datalogger		DATALOGGER	Créez un jeu de données pour enregistrer les données des capteurs
LCD		LCD	Afficher du texte ou des informations sur un écran (LCD)
OLED		OLED	Afficher du texte ou des informations sur un écran (OLED)
Music		MUSIC	Extension pour jouer de la musique

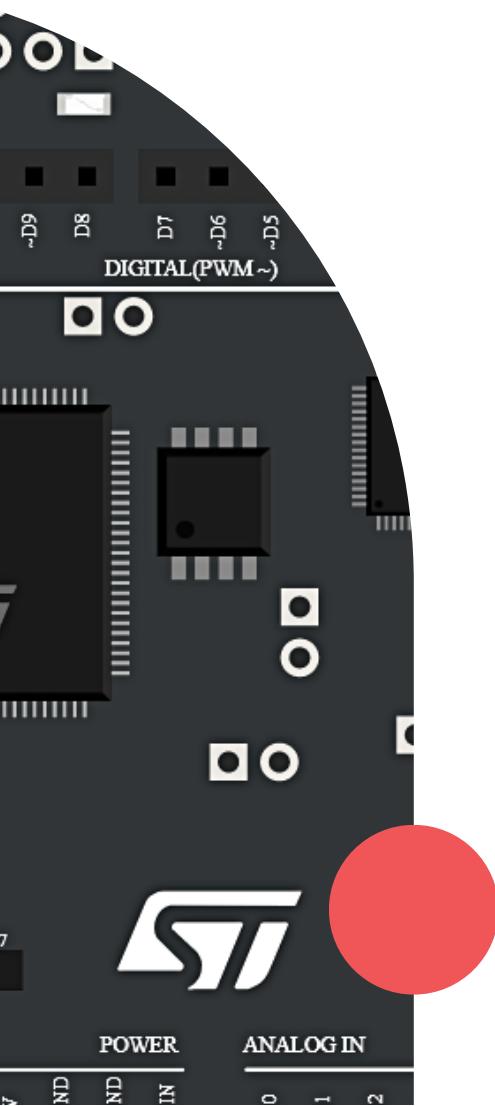
STM32 Education

Collaboration pour l'éducation

Fruit d'une collaboration entre l'académie d'Aix-Marseille et la société STMicroelectronics (Roussset), la carte STM32Education permet de donner du sens à la programmation et à l'algorithmie en proposant une carte complète grâce à laquelle les élèves vont pouvoir concevoir des objets connectés et les programmer simplement.

Cette carte est destinée à de nombreuses disciplines par la richesse de ses capteurs embarqués. La carte STM32Educ est de conception française, son adoption est soutenue par des financements pour l'éducation.

Au-delà de la carte Micro:Bit, elle propose plus de fonctionnalités et un set de capteurs permettant d'intégrer des aspects plus riches aux projets scientifiques et interdisciplinaires qui seront ciblés par Magnetics.



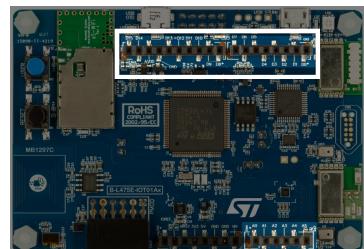
Découvrir la carte STM32 IoT Node et son ensemble de capteurs

La carte "STM32 IoT Node" est une carte programmable, ce qui signifie qu'elle est capable d'exécuter des programmes créés par l'utilisateur.

Pour exécuter ce programme, la carte dispose d'un "microcontrôleur", qui est en quelque sorte son cerveau (voir ci-contre). Par exemple, Le nom du microcontrôleur de notre carte présentée ici est : STM32L475VG.

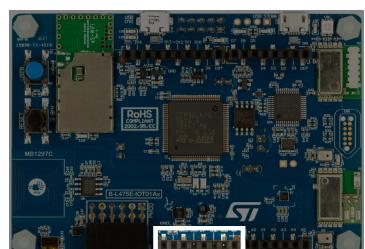
Les GPIO

Comme nous pouvons le constater, il y a beaucoup de "pattes" ou de "broches" sur la carte, appelées "General Purpose Input / Output" (ou GPIO en abrégé). Il est possible de les utiliser pour interagir avec des éléments externes. Même s'il y a beaucoup de GPIO, il n'est pas possible de tous les utiliser. Les GPIO utilisables sont situés en haut et en bas de la carte. Les blocs noirs percés sont appelés "blocs de broches". En regardant attentivement, nous pouvons remarquer les noms des GPIO inscrits autour (par exemple en bas à droite : "D0, D1, D2, D3, ..., A0, A1, A2, ...").



Nous découvrirons les différences entre les broches Ax (A0, A1, ...) et Dx (D0, D1, D2, ...) plus loin dans les activités.

Il reste un autre bloc de broches, celui-ci est spécial, c'est un "power pinout block". Nous pouvons utiliser ces broches pour alimenter des capteurs ou des actionneurs (comme un moteur, une lumière, etc.).



L'inscription sur le dessus du bloc de broches informe sur la manière de l'utiliser. Le "5V" correspond au "+" (pôle positif) d'une batterie et le "GND" (abréviation de "Ground") au "-" (pôle négatif).

Les périphériques

La différence entre le nombre de GPIO disponibles via le bloc de broches et le nombre de pattes du microcontrôleur s'explique par la présence de multiples périphériques déjà connectés au microcontrôleur, disponibles sur la carte "STM32 IoT Node" elle-même. La présence de tous ces périphériques rend cette carte particulièrement accessible, car elle permet de mettre en œuvre un large éventail d'activités, des plus simples aux plus complexes, et des plus basiques aux plus ludiques. C'est un véritable atout pour réaliser des activités entraînantes en classe.

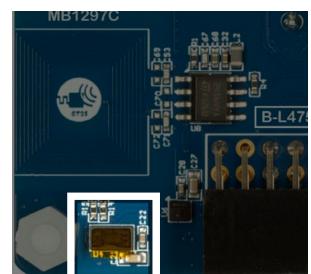
Boutons

Sur le côté gauche de la carte, vous trouverez deux boutons. Le bouton noir est le bouton RESET, permettant au programme de redémarrer si nécessaire. L'autre (bleu) peut être utilisé dans un programme pour détecter quand l'utilisateur appuie sur ce bouton (appui court, appui long, relâchement, etc.). Il peut être utile pour créer des interactions simples avec l'utilisateur, comme un buzzer dans le cadre d'organisation de concours à l'aide de cette carte.



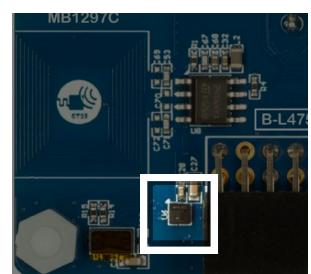
Capteur de distance

Dans le coin inférieur gauche de la carte, juste à droite de la vis en nylon, vous pouvez trouver un capteur pour mesurer la distance. Il est officiellement appelé "temps de vol" (time of flight) parce qu'il mesure le temps que met un rayon laser à faire des allers-retours (voler) entre le capteur et un objet.



Capteur de température et d'humidité

Sur la droite du capteur "temps de vol", on trouve un capteur à la fois thermomètre et hygromètre ("2 en 1"). Cela peut être utile pour mettre en œuvre des activités liées à la surveillance de la chaleur ou pour aborder des notions de météorologie.



Capteur accéléromètre et gyroscope

Juste au-dessus du "power pinout block", se trouve un capteur à la fois accéléromètre et gyroscope ("2 en 1"). L'accéléromètre est utilisé pour mesurer l'accélération. Vous pouvez l'utiliser pour détecter les mouvements de la carte (par exemple, si la carte est secouée). Le gyroscope donne des informations sur l'inclinaison de la carte. Ce capteur fonctionne sur trois axes (X, Y et Z), ce qui implique qu'il est possible de détecter les mouvements dans l'espace 3D.



Capteur de pression atmosphérique

À côté du capteur accéléromètre et gyroscope, vous trouverez un petit capteur appelé baromètre. Ce capteur nous donne la valeur de la pression atmosphérique.



Capteur magnétométrique

À côté du capteur de pression atmosphérique, vous pouvez voir le magnétomètre. Il est utilisé pour récupérer la valeur d'un champ magnétique. Il peut également mesurer des valeurs sur trois axes (X, Y et Z).



Microphone

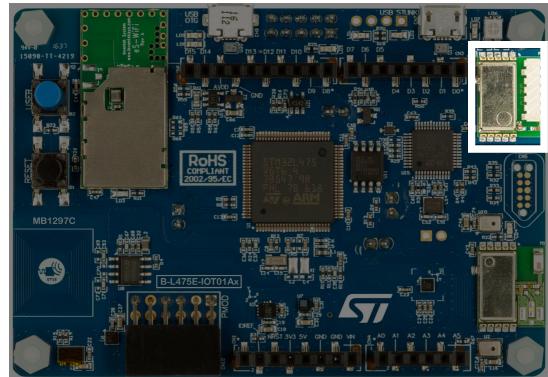
Dans le coin en bas à droite, vous pouvez voir le microphone, utile pour capturer des sons.



Les modules

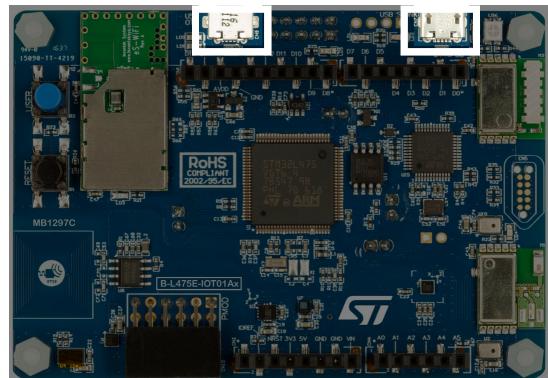
Module Bluetooth

En haut à droite de la carte, vous pouvez trouver le module bluetooth. Il peut être utilisé pour communiquer et échanger des données avec d'autres appareils (comme une autre carte STM32 IoT Node, ou votre téléphone).



Connecteurs Micro-USB

En haut de la carte, vous pouvez voir deux connecteurs micro-USB. Le port USB de droite est celui que vous utiliserez le plus souvent, car il permet de connecter la carte à votre ordinateur et de transférer le programme que vous aurez fait sur MakeCode au microcontrôleur. Le port de gauche, appelé "port USB OTG", permet de programmer la carte pour qu'elle agisse et soit reconnue comme un autre dispositif tel qu'un clavier, une souris ou une manette de jeu.



STeaMi et connecteurs Jacdac

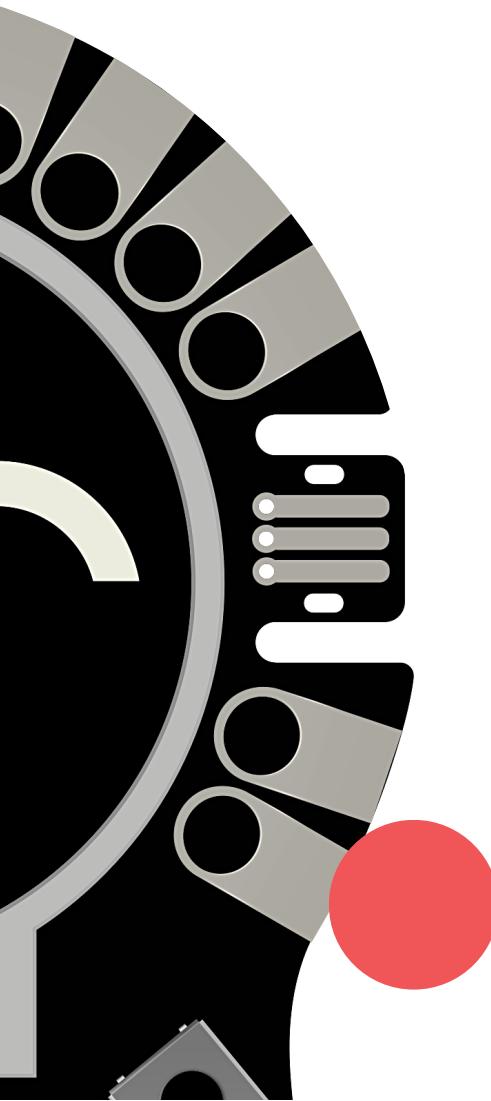
Une nouvelle carte pour mieux répondre aux besoins des enseignants

Fruit d'une coopération d'envergure dans le cadre du projet I-NOVMICRO porté par le Campus d'Excellence Industrie du Futur – GIP FCIP et financé dans le cadre du PIA3, la carte pédagogique STeaM32 ou STeaMi est un support matériel pour les activités d'apprentissage de la programmation dans l'enseignement secondaire.

Outre les aspects purement techniques et technologiques importants pour illustrer les concepts, la carte est adaptée dans sa forme et son utilisation pour être utilisée dans une classe avec des élèves de 10 à 18 ans.

Une grande partie de sa facilité de mise en œuvre provient de son intégration logicielle dans les différentes plateformes éducatives telles que Makecode, Scratch et CircuitPython.

Cependant, la forme et la facilité d'utilisation sont d'une grande importance pour obtenir un produit qui répond aux besoins des enseignants.



Découvrir la carte STeA32 et son ensemble de capteurs

La carte STeA32 est basé sur la famille STM32 de microcontrôleurs 32-bits en circuits intégrés réalisés par la société Franco-Italienne STMicroelectronics. Le processeur d'application STM32WB55 est le lieu d'exécution des programmes utilisateurs. Une seule application complète, comprenant le code utilisateur, le code d'exécution et la pile Bluetooth, est chargée et exécutée directement depuis la mémoire flash de la puce. Toutes les broches GPIO accessibles à l'utilisateur sont fournies par ce processeur. Il y a un moteur radio 2.4GHz intégré utilisé pour fournir des capacités Bluetooth via une antenne hors puce.

Communication sans fil Bluetooth

Le transceiver 2.4GHz embarqué supporte les communications Bluetooth via le MCU M0, qui fournit une pile Bluetooth Low Energy entièrement qualifiée. Cela permet à la carte de communiquer avec une large gamme de périphériques Bluetooth, y compris les smartphones et les tablettes.

Boutons

Les six boutons à l'avant de la carte, et le bouton 1 à l'arrière, sont des boutons tact momentanés de type "push to make". Le bouton arrière est connecté au processeur STM32WB55 pour la réinitialisation du système. Cela signifie que l'application se réinitialisera indépendamment du fait qu'elle soit alimentée par USB ou par batterie. Les boutons avant peuvent être programmés dans l'application utilisateur pour n'importe quel usage. Ils sont débités par le logiciel, ce qui inclut également la détection des pressions courtes et longues. Les boutons fonctionnent dans un mode électrique inversé typique, où une résistance de rappel garantit un '1' logique lorsque le bouton est relâché, et un '0' logique lorsque le bouton est enfoncé. Les boutons A et B sont connectés aux broches GPIO qui sont également accessibles sur le connecteur de bord.

Écran

L'écran est un écran circulaire IPS TFT couleur pour du texte et des graphiques haute résolution. Il est connecté au processeur d'application. Le logiciel d'exécution rafraîchit de façon répétée cet écran à une vitesse élevée, de sorte qu'il se trouve dans la plage de persistance de la vision de l'utilisateur et qu'aucun scintillement ne soit détecté.

Capteur de mouvement

La carte est équipée d'une puce combinant un accéléromètre et un magnétomètre qui fournit une détection sur 3 axes et une détection de l'intensité du champ magnétique. Elle comprend également une détection matérielle de certains gestes (comme la détection de chute) et une détection de gestes supplémentaires (par exemple, logo vers le haut, logo vers le bas, secousse) via des algorithmes logiciels. Un algorithme logiciel dans le runtime standard utilise l'accéléromètre embarqué pour transformer les lectures en une lecture de boussole indépendante de l'orientation de la carte. La boussole doit être étalonnée avant d'être utilisée, et le processus d'étalonnage est automatiquement lancé par le logiciel d'exécution. Ce dispositif est connecté au processeur d'application via le bus I2C.

Broches d'entrée/sortie à usage général

Le connecteur de bord fait ressortir de nombreux circuits GPIO du processeur d'application. Certains de ces circuits sont partagés avec d'autres fonctions de la carte, mais beaucoup de ces circuits supplémentaires peuvent être réaffectés à un usage général si certaines fonctions logicielles sont désactivées.

Alimentation électrique

L'alimentation de la carte peut se faire via la connexion USB, via la puce d'interface (qui possède un régulateur intégré), ou via une batterie branchée sur le connecteur supérieur. Il est également possible (avec précaution) d'alimenter la carte à partir du pad 3V situé en bas. Le pad 3V du bas peut être utilisé pour fournir une petite quantité de puissance aux circuits externes.

Interface

La puce gère la connexion USB, et est utilisée pour flasher un nouveau code sur la carte, envoyer et recevoir des données série vers votre ordinateur principal.

Communications USB

La carte possède une pile de communication USB, qui est intégrée dans le firmware du bootloader. Cette pile permet de glisser et déposer des fichiers sur le disque de stockage de masse afin de charger du code dans le processeur d'application. Elle permet également de transmettre des données en série vers et depuis le processeur d'application via USB vers un ordinateur hôte externe, et supporte le protocole CMSIS-DAP pour le débogage hôte des programmes d'application.

Débogage

Le processeur peut être utilisé avec des outils hôtes spéciaux pour déboguer le code qui s'exécute sur le processeur d'application. Il se connecte au processeur d'application via 4 fils de signal. Le code du processeur peut également être débogué via son interface de débogage logiciel SWD intégrée, par exemple pour charger le code initial du chargeur de démarrage dans ce processeur au moment de la fabrication, ou pour récupérer un chargeur de démarrage perdu.

L'intégration de connecteurs Jacdac

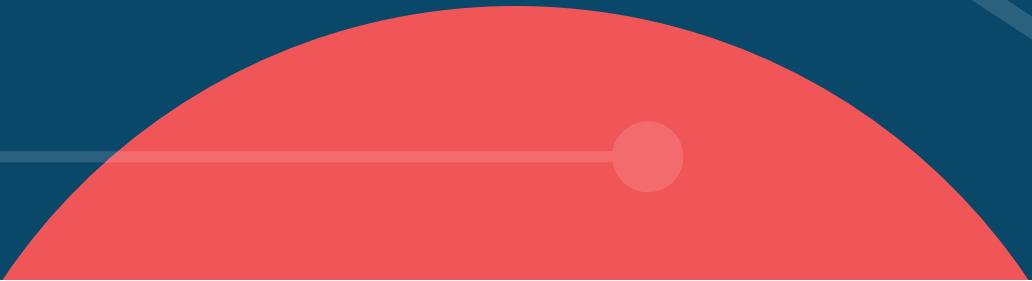
En complément des modules et éléments présentés ci-dessus, la carte STEaM32 intègre deux connecteurs Jacdac ainsi qu'un connecteur Micro:bit permettant de connecter un hub Jacdac afin de câbler un nombre plus important de capteurs. L'environnement Jacdac est un système de connectique plug and play pour ajouter en cascade ses capteurs/actionneurs à la volée. Une fois connectés, les périphériques sont reconnus automatiquement ainsi que les services associés, ce qui permet d'ajouter des capteurs à tout instant.

L'écosystème Jacdac a été conçu pour être convivial et adapté à l'éducation. Les dispositifs Jacdac communiquent à l'aide de paquets sur un bus, où chaque dispositif s'annonce et présente son ensemble de services. Chaque dispositif Jacdac possède un minuscule microcontrôleur qui exécute le protocole Jacdac et communique sur le bus. Les paquets Jacdac sont envoyés en série entre les dispositifs physiques sur le bus Jacdac et peuvent également être envoyés sur WebUSB/WebBLE, fournissant une connectivité à des outils et services basés sur le Web et fonctionnant dans un navigateur Web.

partie 3

Programmer sur makecode.lets-steam.eu

fiches d'activité



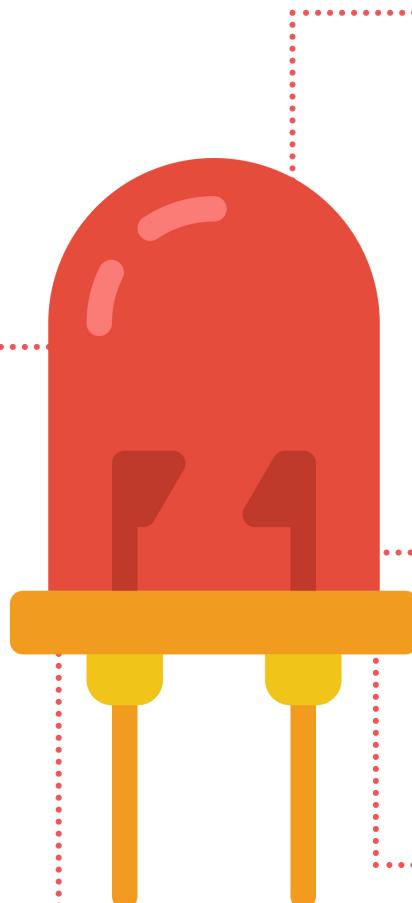
FAISONS CLIGNOTER UNE LED

POUR BIEN DÉBUTER

#R1AS01



Disponible sur



De quoi parle-t-on ?

Une LED est un composant électronique qui produit de la lumière lorsqu'elle est traversée par un courant. Elle peut être utilisée pour éclairer une pièce, ou pour indiquer quelque chose (un réservoir presque vide, une machine allumée, etc.). Les LED existent sous différentes formes et couleurs.

Durée

15 minutes

Matériel

- 1 carte programmable "STM32 IoT Node"
- 1 câble USB Micro-B

Niveau de difficulté

Débutant

OBJECTIFS D'APPRENTISSAGE

- Programmer avec des blocs
- Apprendre les bases de MakeCode
- Utiliser la LED intégrée

POUR BIEN DÉBUTER - FAISONS CLIGNOTER UNE LED



Dans cette activité de prise en main, vous allez aborder le concept de **broche**. Une broche est un fil physique connecté directement au microcontrôleur. L'état d'une broche donne des informations sur le passage ou non du courant dans celle-ci. Plus précisément :

- **LOW** signifie qu'il n'y a pas de courant
- **HIGH** signifie qu'il y a du courant.

Pour rendre le courant visible, nous utilisons un composant appelé LED (light-emitting diode) déjà disponible sur la carte, qui s'allumera lorsqu'il passera par la broche.



ETAPE 1 - CONSTRUIRE



Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (en bas à droite de la carte). Vous devriez voir un nouveau lecteur appelé **DIS_L4IOT** sur votre ordinateur. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

1

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource: makecode.lets-steam.eu

2

Organiser les blocs

Voici les différentes étapes vous permettant de faire clignoter une LED à l'aide de l'éditeur de blocs :

3

Étape 1 - Ajouter une boucle infinie

Comme nous voulons que le programme fasse clignoter la LED indéfiniment, la première étape consiste à ajouter le bloc **"toujours"**. Il sera présent par défaut à l'ouverture de l'éditeur, mais vous pouvez également le trouver dans le sous-menu **BOUCLES**.



Ajouter une boucle infinie en utilisant le bloc FOREVER



Sélectionner et éditer le bloc de broches (ou pin) digital write pour allumer la LED

Étape 2 - Allumer la LED

Contrôler une LED est une tâche simple car elle ne peut être qu'allumée (le courant la traverse) ou éteinte (le courant ne circule pas). Pour y parvenir, nous devons définir l'état de la broche (plus communément appelée **pin**) où la LED est connectée. Dans notre cas, si nous voulons allumer la LED, nous devons mettre l'état de la broche sur **HIGH**. L'état de la broche sur **LOW** l'éteindra alors. Dans MakeCode, pour contrôler l'état d'une broche, sélectionnez le sous-menu **BROCHES**, puis faites glisser le bloc broche **digital write** à l'intérieur de la boucle **toujours**.

POUR BIEN DÉBUTER - FAISONS CLIGNOTER UNE LED



ETAPE 1 - CONSTRUIRE



Étape 3 - Créer le clignotement

Pour créer le clignotement, il est nécessaire que nous puissions voir la LED s'allumer et s'éteindre pendant une durée similaire. Pour créer ce clignotement, nous devons suivre les étapes suivantes :

1) Créer une pause lorsque la LED est allumée pour voir la lumière : avant d'éteindre la LED, nous devons attendre un petit temps, une demi-seconde (500 millisecondes) par exemple, avec la lumière allumée. Pour ce faire, ajoutez le bloc **pause** (à l'intérieur du sous-menu LOOPS), et définissez la valeur à 500 (pour 500 millisecondes).

i Vous pouvez choisir une valeur dans la liste, ou saisir directement une valeur personnalisée.

2) Éteignez la lumière pendant une durée similaire pour créer le clignotement : jusqu'à présent, vous avez fait la moitié du travail ! Ajoutez un autre bloc **digital write** et un autre bloc **pause** pour éteindre la LED et attendre à nouveau 500 ms, permettant de créer cet effet de clignotement. Combiné avec la boucle infinie, on peut voir ce clignotement se répéter à l'infini.

i Au lieu de choisir des blocs dans les sous-menus, vous pouvez cliquer avec le bouton droit de la souris sur un bloc et le "duplicier".

Grâce à cette activité facile, vous avez découvert comment créer un morceau de code en utilisant la programmation par blocs. Vous pouvez jeter un coup d'œil à l'éditeur JavaScript pour voir le code généré tel qu'il est indiqué dans la section "**Programmer**" ci-dessous. Dans les prochaines fiches d'activité, n'hésitez pas à copier/coller directement dans l'éditeur JavaScript de MakeCode le code fourni pour voir le résultat en blocs.

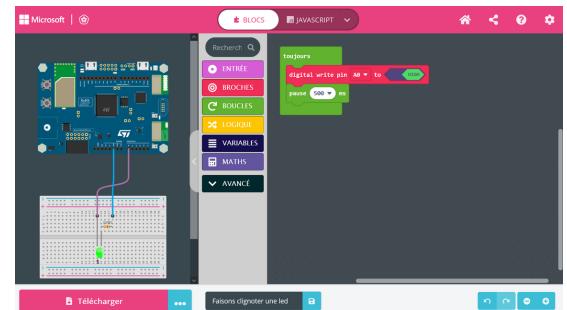
Programmer la carte

Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter. Votre premier programme est maintenant en cours d'exécution et la LED intégrée devrait clignoter !

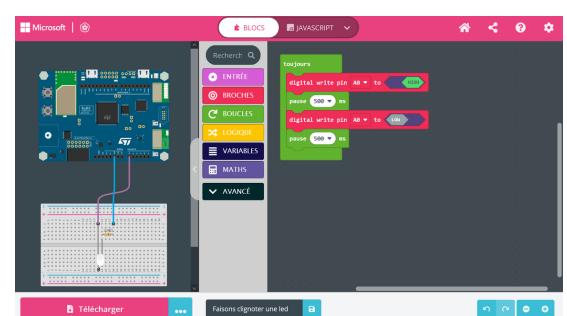
Exécuter, modifier, s'amuser

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (en appuyant sur le bouton RESET). Essayez de comprendre le code et commencez à le modifier en changeant la période entre deux clignotements. N'hésitez pas à essayer de faire clignoter à différents rythmes ou à faire un **SOS** visuel en **morse**.

Ressource: <https://en.wikipedia.org/wiki/SOS>



Créez une pause lorsque la LED est allumée pour voir la lumière



Éteignez la lumière pendant une durée similaire à celle du clignement des yeux



Blocs permettant de programmer l'activité d'une LED clignotante

4

5

BREADBOARD

RÉALISEZ VOTRE PREMIER CIRCUIT ÉLECTRONIQUE !

#R1AS02

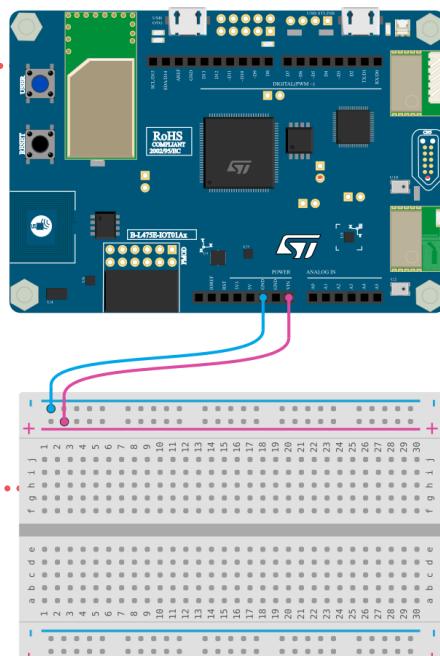


Disponible sur



Prérequis

- R1AS01 - Faire clignoter une LED



Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 breadboard
- 1 set de résistances
- 1 set de LED
- Câbles de connexion

De quoi parle-t-on ?

Une breadboard est une carte plastique rectangulaire percée d'un grand nombre de petits trous permettant d'insérer facilement des composants électroniques afin de créer un prototype de circuit électrique.

Durée

15 minutes

Niveau de difficulté

Débutant

OBJECTIFS D'APPRENTISSAGE

- Découvrir les breadboards
- Réaliser un circuit simple sur une breadboard
- Réaliser un circuit électrique simple avec des LED et des résistances

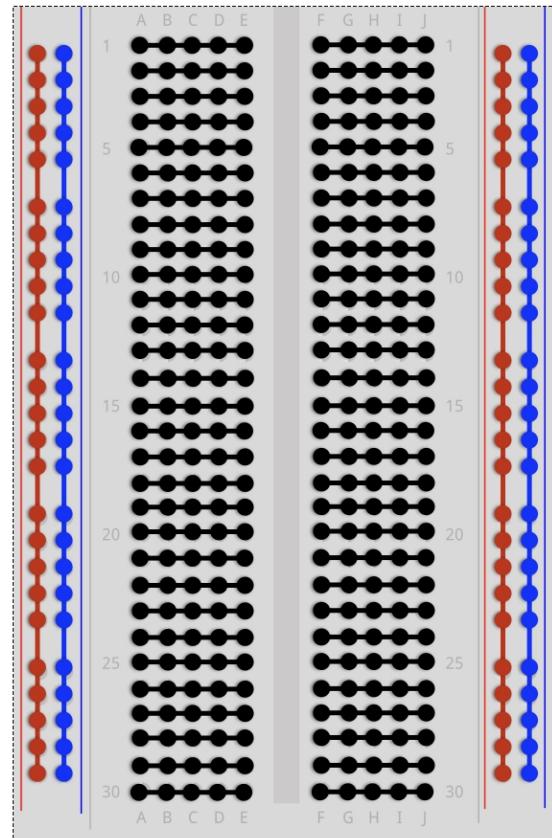


BREADBOARD



Une breadboard est composée de nombreux petits trous et sert à créer un circuit sur ce petit rectangle de plastique. Avant de commencer, vous devez en comprendre les différentes parties.

Les trous d'une breadboard sont faits pour connecter des composants électroniques entre eux. Lorsque nous voulons créer un circuit électrique, nous avons besoin de plusieurs connexions sur le même fil. Pour ce faire, la breadboard est organisée en deux types de rangées ou bandes. Les **rangées de bus** sont principalement utilisées pour les connexions d'alimentation et se trouvent sur les deux colonnes extérieures d'une breadboard. Les **rangées de raccordement** sont principalement utilisées pour les composants électroniques et sont connectées ligne par ligne. Chaque rangée est composée de cinq trous, vous ne pouvez donc connecter que cinq composants au maximum dans une rangée. Tant qu'un composant possède des fils (longues tiges métalliques dépassant du composant) ou des pattes (tiges métalliques plus courtes), il peut être utilisé avec une breadboard. Pour relier certaines rangées entre elles, on utilise des câbles de liaison.



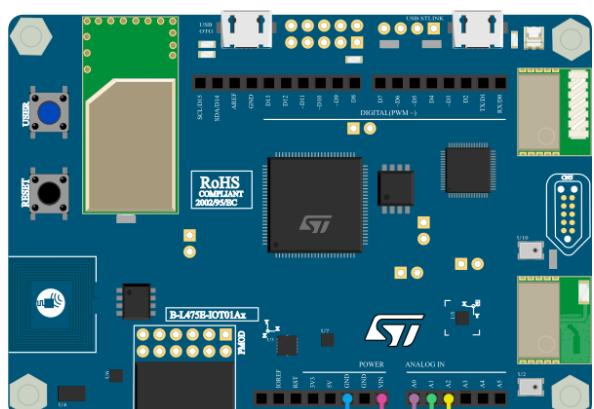
ETAPE 1 - CONSTRUIRE



Câbler l'alimentation électrique

Avant de connecter les composants électroniques, nous ajoutons généralement quelques câbles aux rangées de bus pour distribuer l'alimentation (**+5V** et broche **GND**). Prenez deux câbles et effectuez les connexions ci-contre.

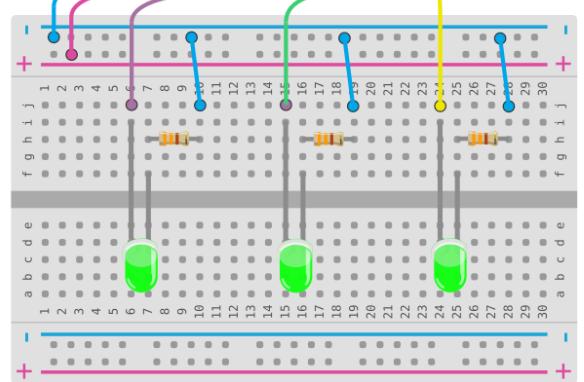
1



Câbler la première LED

Notre circuit se compose d'une simple LED connectée à une broche de la carte. Connectez l'anode de la LED sur la broche appelée **A0** (pour Analog 0). Connectez ensuite la cathode à une résistance (330 ohms) et branchez le fil de la résistance non connectée sur la broche appelée **GND**.

2



Câbler les LEDs



i La LED a une orientation. Pour désigner l'orientation correcte, chaque branche a un nom. Voici comment trouver la différence entre l'anode et la cathode : **Anode** : C'est le '+' de la LED. La branche anodique est plus longue que la branche cathodique. **Cathode** : C'est le '-' de la LED. La branche de la cathode est plus courte que le fil de l'anode.



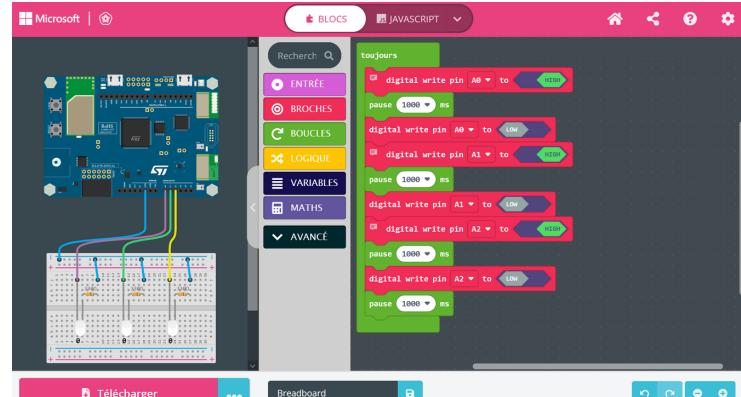
ETAPE 1 - CONSTRUIRE



Câbler les autres LED

Nous allons dupliquer le circuit précédent avec deux LED supplémentaires. L'anode de ces nouvelles LED sera connectée à la broche **A1** et à la broche **A2**.

3



L'éditeur de blocs de MakeCode

Connecter la carte à l'ordinateur

Connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

4

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource: makecode.lets-steam.eu

5

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter. Votre programme est maintenant en cours d'exécution !

6



Blocs permettant l'exécution du programme

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Utilisez les connaissances acquises sur cette fiche d'activité pour réaliser des projets plus ou moins complexes et explorez les fiches d'activité suivantes.

7



ETAPE 2 - PROGRAMMER



```
forever(function () {  
    // Faire clignoter la première LED  
    pins.A0.digitalWrite(true)  
    pause(1000)  
    pins.A0.digitalWrite(false)  
  
    // Faire clignoter la deuxième LED  
    pins.A1.digitalWrite(true)  
    pause(1000)  
    pins.A1.digitalWrite(false)  
  
    // Faire clignoter la troisième LED  
    pins.A2.digitalWrite(true)  
    pause(1000)  
    pins.A2.digitalWrite(false)  
    pause(1000)  
})
```

Comment cela fonctionne-t-il ?

Ce programme est une version étendue du programme “Faire clignoter une LED” adapté avec trois LED. Pour chaque LED :

- le bloc **digitalWrite** éteint ou allume une LED spécifique
- le bloc **pause** attend un petit moment.

BOUTONS ET AFFICHAGE LED

#R1AS03



Disponible sur



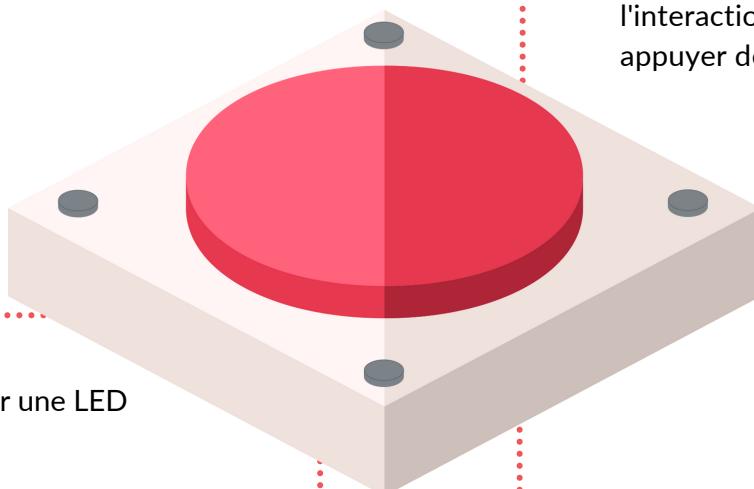
De quoi parle-t-on ?

Nous allons apprendre à interagir avec la carte en utilisant un simple bouton-poussoir. Il en existe de toutes les formes et de toutes les tailles, mais ils nécessitent tous l'interaction la plus simple : appuyer dessus !



Prérequis

- R1AS01 - Faire clignoter une LED
- R1AS02 - Breadboard



Durée

25 minutes

Niveau de difficulté

Intermédiaire

Matériel

- 1 carte programmable "**STM32 IoT Node Board**"
- 1 câble USB Micro-B
- 2 boutons-poussoirs
- 1 set de LED
- 1 set de résistances
- 1 breadboard
- Câbles de connexion

OBJECTIFS D'APPRENTISSAGE

- Ajouter de l'interactivité
- Gérer un événement déclenché par un bouton
- Utiliser une variable pour stocker l'état actuel du programme
- Câbler un circuit simple sur une breadboard avec des boutons et des LED
- Utiliser le simulateur MakeCode



BOUTONS ET AFFICHAGE LED



Pour apprendre à utiliser un bouton, **jouons à un jeu de questions-réponses !**

L'idée est assez simple : **un animateur, deux joueurs, un bouton, et une LED pour chaque joueur.**

Lorsque l'animateur pose une question, les joueurs doivent appuyer en premier sur leur bouton pour donner la bonne réponse. Les LED indiquent quel joueur a appuyé sur son bouton en premier et autorisé à répondre.



ETAPE 1 - CONSTRUIRE

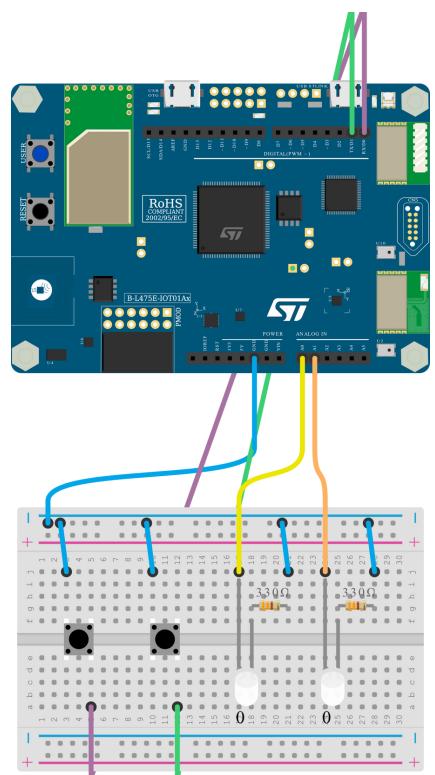


Câbler les boutons et les LED

Connectez une patte de chaque bouton à la broche **GND** de la carte. Connectez ensuite l'autre patte sur la **broche D0** pour le joueur 1, et sur la **broche D1** pour le joueur 2. Connectez l'anode de la LED du joueur 1 sur la **broche A0** et celle du joueur 2 sur la **broche A1**. Connectez la **cathode** de chaque LED à une résistance (330 ohms). Branchez ensuite les pattes non connectées des résistances sur la broche **GND**.



La LED a une orientation. Pour désigner l'orientation correcte, chaque branche a un nom. Voici comment trouver la différence entre l'anode et la cathode : **Anode** : C'est le '+' de la LED. La branche anodique est plus longue que la branche cathodique. **Cathode** : C'est le '-' de la LED. La branche de la cathode est plus courte que le fil de l'anode.



Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le connecteur **micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

Câbler les boutons et les LED

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter, votre buzzer de quiz est prêt !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Utilisez les connaissances acquises sur cette fiche d'activité pour réaliser des projets plus ou moins complexes et explorez les fiches d'activité suivantes.

1

2

3

4

5



ÉTAPE 2 - PROGRAMMER



```
//Initialisation
let weCanPushIt = true
pins.A0.digitalWrite(false)
pins.A1.digitalWrite(false)
```

Initialisation

Dans un premier temps, nous devons déclarer une variable appelée **weCanPushIt**, de type booléen - un type de données avec seulement deux valeurs possibles, généralement "**true**" et "**false**". Cette variable sera utile pour savoir si notre appui sur le bouton sera pris en compte, ou si l'autre joueur est déjà en train de le faire. Les deux dernières lignes donnent l'information que toutes les LED sont éteintes.

i Une **variable** est un moyen de nommer et de stocker une valeur pour une utilisation ultérieure par le programme, comme les données d'un capteur ou une valeur intermédiaire utilisée dans un calcul. Une variable a un nom et un type. Le type permet de spécifier le type de données que la variable peut contenir. Sur les langages dynamique comme python ou javascript, le type n'est pas forcément explicite.

```
input.buttonD0.onEvent(ButtonEvent.Down, function () {
    if (weCanPushIt) {
        weCanPushIt = false
        pins.A0.digitalWrite(true)
        pause(3000)
        pins.A0.digitalWrite(false)
        weCanPushIt = true
    }
})

input.buttonD1.onEvent(ButtonEvent.Down, function () {
    if (weCanPushIt) {
        weCanPushIt = false
        pins.A1.digitalWrite(true)
        pause(3000)
        pins.A1.digitalWrite(false)
        weCanPushIt = true
    }
})
```

Interactions

Le code principal concerne les interactions des boutons réalisées avec les fonctions **input.buttonXX.onEvent**.

i Une **fonction** est un bloc de code qui exécute une tâche spécifique. Elle est vraiment utile pour simplifier le code et rendre un bloc de code plus compréhensible.

La ligne la plus importante ici est la condition **if (weCanPushIt) { ... }** qui teste si les joueurs ont déjà appuyé sur leur bouton ou non. Si cela est le cas (**weCanPushIt** est égal à **true**), nous :

- Affectons **weCanPushIt** à **false**, pour empêcher l'adversaire de répondre,
- Allumons la LED du joueur pour montrer qui est le gagnant,
- Attendons 3 secondes (3 000 millisecondes),
- Éteignons la LED du gagnant,
- Affectons **weCanPushIt** à **true**, pour permettre aux joueurs d'appuyer sur leurs boutons.

CAPTEURS DE LUMIÈRE

DÉCOUVERTE ET PREMIÈRE UTILISATION

#R1AS04

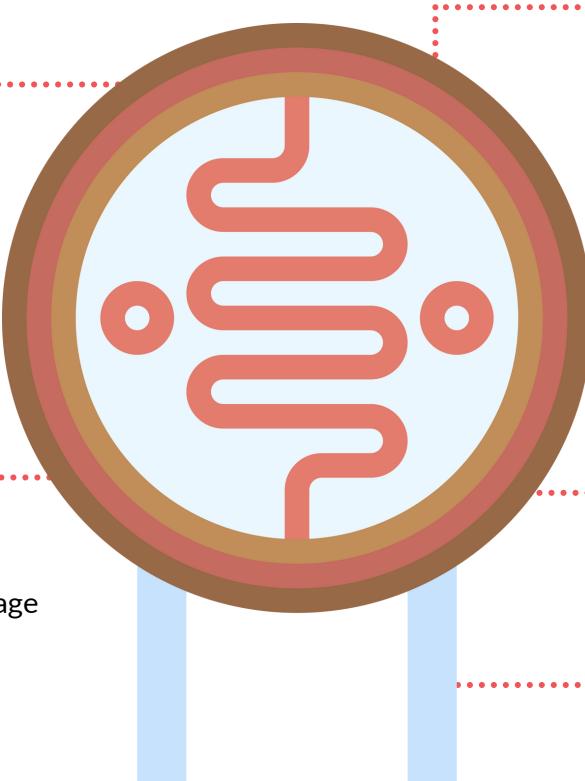


Disponible sur



Prérequis

- R1AS02 - Breadboard
- R1AS03 - Boutons et affichage LED



Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 set de résistances
- 1 photorésistance (LDR)
- 1 breadboard
- Câbles de connexion

De quoi parle-t-on ?

Cette fiche d'activité aborde les résistances. Une photorésistance (light dependent resistor - LDR) est un composant utilisé pour mesurer l'intensité de la lumière.

Durée

25 minutes

Niveau de difficulté

Intermédiaire

LEARNING OBJECTIVES

- Créer un détecteur de lumière basique avec quelques composants électroniques sur une breadboard et connectez-le à la carte
- Créer un programme dans MakeCode qui est capable de mesurer une quantité physique au moyen d'un capteur analogique
- Produire un graphique qui montre comment une valeur mesurée varie dans le temps



DÉCOUVERTE DES CAPTEURS DE LUMIÈRE



Cette activité illustre une caractéristique clef du *Physical Computing* : la possibilité de mesurer une quantité physique à l'aide d'un capteur et représenter graphiquement la façon dont cette quantité varie dans le temps. Nous allons connecter une **photorésistance** (LDR) à la carte pour mesurer l'intensité de lumière. Ce type de capteur est appelé un **capteur analogique** car il présente une caractéristique analogique du circuit (tension) pour représenter la valeur physique réelle.

Ressource: <https://www.watrical.com/what-are-analog-sensors-types-and-their-characteristics/>



ETAPE 1 - CONSTRUIRE



Câbler la cellule photoélectrique

Le circuit que nous devons assembler se compose de deux éléments : une **résistance de 4,7 kΩ** et une **cellule photoélectrique**.

1

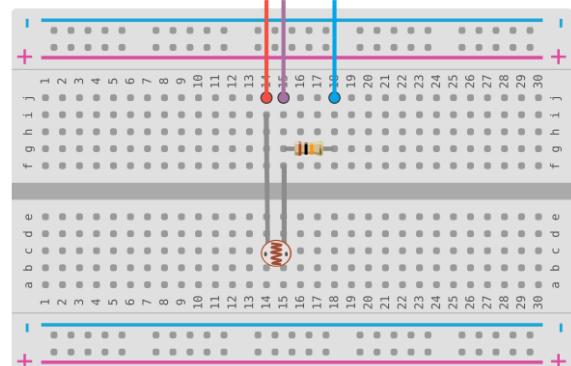
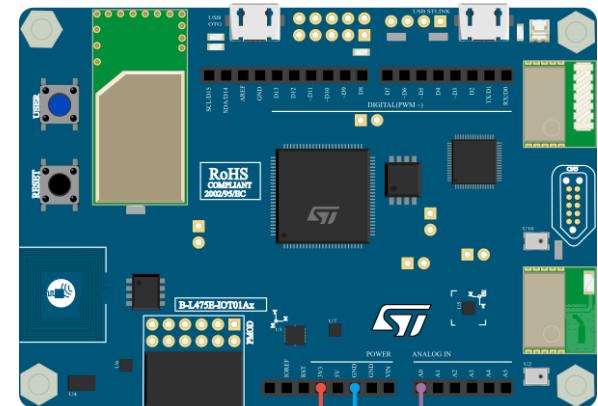
i La couleur des trois premières bandes indique la valeur de résistance du composant, selon un code connu sous le nom de "code couleur des résistances". La quatrième bande indique que la valeur de résistance est sujette à une tolérance (incertitude) qui peut être de 5% (si la bande est dorée) ou de 10% (si la bande est argentée) de la valeur de résistance nominale.

i Les **photorésistances** (alias LDR, cellule photoélectrique, ou cellule photoconductrice) sont des composants dont la résistance électrique varie en fonction de l'intensité de la lumière à laquelle ils sont exposés.

La manière la plus simple de mesurer un capteur résistif est de connecter une extrémité à l'alimentation et l'autre à une résistance de rappel (pull-down) à la masse. Ensuite, le point situé entre la résistance de rappel et la cellule photoélectrique est connecté à l'entrée analogique d'un microcontrôleur. Un tel montage constitue ce que nous appelons un capteur analogique. Ce terme signifie que **ce circuit est capable de capturer une grandeur physique** (à savoir l'intensité lumineuse) et de la transformer en une **grandeur électrique proportionnelle** (à savoir une tension dont la valeur est comprise entre 0 V et 3,3 V). Ces deux composants doivent être assemblés sur une breadboard, comme le montre l'image ci-contre.

Câbler la breadboard à la carte STM32 IoT Node Board

Une fois que la breadboard a été assemblée, il faut la connecter à la carte. L'image montre que la carte possède quatre blocs de broches, nommés respectivement **CN1**, **CN2**, **CN3** et **CN4**. Comme les quatre blocs ont des fonctions différentes, utilisez le bouton bleu situé à l'un des quatre coins de la carte comme point de repère pour identifier correctement les quatre blocs.



Montage de la résistance de 4,7 kΩ et de la cellule photoélectrique sur la breadboard.

DÉCOUVERTE DES CAPTEURS DE LUMIÈRE



ETAPE 1 - CONSTRUIRE

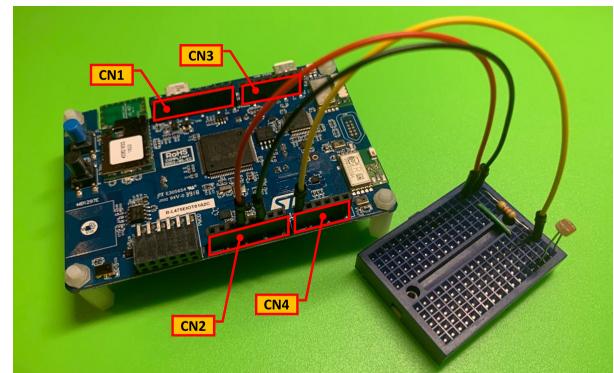


Le fil rouge doit être connecté à la **broche 4** du bloc **CN2**, qui est connecté en interne à un potentiel de 3,3 V. Le fil noir doit être connecté à la **broche 6** du bloc **CN2**, qui est connectée en interne au potentiel de masse (**GND**). Enfin, le fil jaune doit être connecté à la **broche 1** du bloc **CN4**. Cette broche est connectée en interne à la broche d'entrée analogique nommée **A0**.

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le connecteur **micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

3



Câbler la breadboard à la carte STM32 IoT Node Board

Ouvrir MakeCode

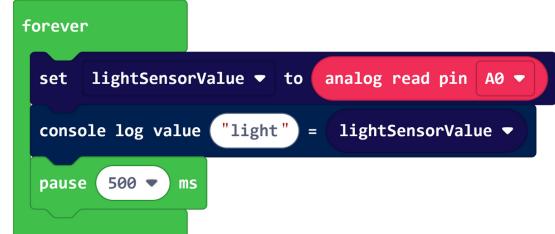
Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource: makecode.lets-steam.eu

4

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre programme est prêt !

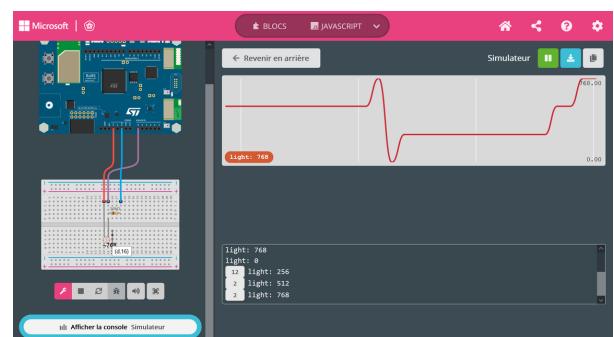


Blocs permettant l'exécution du programme

Connecter la console de la carte

Dans l'éditeur MakeCode, cliquez sur le bouton "Show console Simulator" sur le côté gauche, en dessous de la représentation de la carte. Le terminal montre alors les valeurs de lumière lues périodiquement par le programme. Ces valeurs peuvent être exportées sous forme de fichier CSV en cliquant sur le bouton "Export data" dans le coin supérieur droit de la console.

6



Console sur l'éditeur MakeCode

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Essayez de comprendre l'exemple et commencez à le modifier en changeant la période entre deux sessions de mesure. Vous pouvez cacher la cellule photoélectrique avec votre main pour observer directement le changement de valeur.

7



ÉTAPE 2 - PROGRAMMER



```
let lightSensorValue = 0
forever(function () {
    lightSensorValue = pins.A0.analogRead()
    console.logValue("light", lightSensorValue)
    pause(500)
})
```

Comment cela fonctionne-t-il ?

Le code se compose de :

- un bloc de boucle infinie (**forever**) ;
- un bloc d'enregistrement de la console (**console.log**) ;
- un bloc de **pause**.

Le bloc **forever** implémente "une boucle", qui continue à exécuter trois instructions de base jusqu'à ce que la carte soit éteinte.

Le premier bloc lit la valeur de la broche d'entrée analogique **A0** et la stocke dans une variable appelée **lightSensorValue**. Cette valeur est un nombre entier compris entre **0** et **1023**.



Une broche d'entrée analogique peut être utilisée pour lire une valeur comprise entre 0 et 1023. Cette valeur est proportionnelle à la tension appliquée à la broche, qui DOIT être comprise entre 0 V et 3,3 V (par rapport à GND).

Le deuxième bloc écrit sur la console de la carte ce qui a été obtenu par la lecture de la valeur du capteur. Dès que cette instruction a été exécutée, la carte suspend son activité (**pause**) pendant 500 millisecondes, c'est-à-dire une demi-seconde.

Des questions se posent alors naturellement : qu'est-ce que la console de la carte ? Comment est-il possible de lire ce qui est écrit sur la console ? La console de la carte permet à la carte d'interagir simplement avec le PC qui lui est connecté via le câble USB.

POTENTIOMÈTRE

#R1AS05



Disponible sur

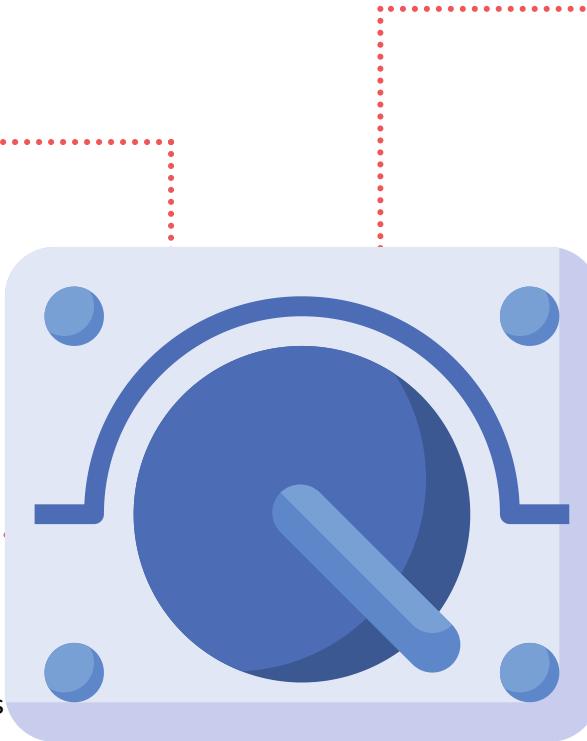


Prérequis

- R1AS01 - Faire clignoter une LED
- R1AS02 - Breadboard
- R1AS04 - Découverte des capteurs de lumière

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 breadboard
- 1 potentiomètre
- 1 set de LED
- 1 set de résistances
- Câbles de connexion



De quoi parle-t-on ?

Dans cette fiche d'activité, nous allons apprendre ce qu'est un potentiomètre en programmant la carte pour régler la luminosité d'une LED en tournant un bouton.

Durée

20 minutes

Niveau de difficulté

Intermédiaire

OBJECTIFS D'APPRENTISSAGE

- Câbler les composants externes à la carte
- Lire une entrée analogique à l'aide d'un potentiomètre
- Utiliser une entrée analogique pour écrire une sortie analogique



POTENTIOMÈTRE



Un potentiomètre est une **résistance à trois bornes** avec un contact glissant ou rotatif qui forme un diviseur de tension réglable. Si l'on utilise seulement deux bornes, une extrémité et le curseur, il agit comme une résistance variable ou un rhéostat. L'instrument de mesure appelé potentiomètre est essentiellement un **diviseur de tension** utilisé pour mesurer le potentiel électrique (tension) ; le composant est une mise en œuvre du même principe, d'où son nom.

Les potentiomètres sont couramment utilisés pour **contrôler des dispositifs électriques** tels que les commandes de volume des équipements audio. Les potentiomètres actionnés par un mécanisme peuvent être utilisés comme transducteurs de position, par exemple dans un joystick. Les potentiomètres sont rarement utilisés pour contrôler directement une puissance importante (plus d'un watt) car la puissance dissipée serait comparable à la puissance de la charge contrôlée.

Ressource: <https://en.wikipedia.org/wiki/Potentiometer>



ÉTAPE 1 - CONSTRUIRE



Câbler le potentiomètre

Connectez la **pin gauche** du potentiomètre à **GND**. La **pin de droite** doit être connectée à **3.3V**. Connectez celle du milieu à **A0**.

Câbler la LED

Connectez l'**anode (+)** de la LED sur **D9**. Connectez la **cathode (-)** de la LED à une **résistance (330 ohms)**. Puis, connectez le côté non connecté de la résistance à **GND**.

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

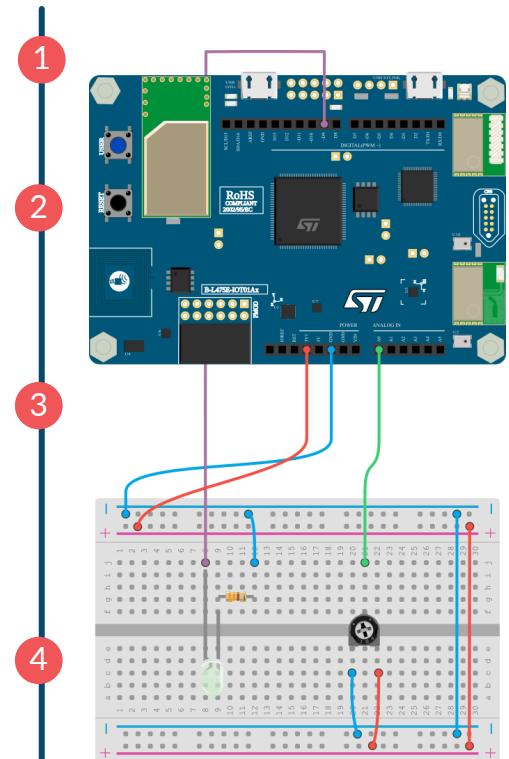
Ressource: makecode.lets-steam.eu

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre carte est prête !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Essayez de comprendre l'exemple et commencez à le modifier.



Câbler le potentiomètre et la LED

5

6



ÉTAPE 2 - PROGRAMMER



```
forever(function () {
    pins.D9.analogWrite(pins.A0.analogRead())
})
```

Comment cela fonctionne-t-il ?

Le code se compose de trois éléments :

- un bloc **forever** ;
- un bloc **analogRead** ;
- un bloc **analogWrite**.

Le bloc **forever** met en œuvre "une boucle" qui continue à exécuter les instructions jusqu'à ce que la carte soit éteinte.

Le bloc **analogRead** est utilisé pour obtenir la valeur du potentiomètre sur la *pin A0*. Cette valeur est un nombre entier compris entre 0 et 1023. En tournant le bouton du potentiomètre, on modifie la valeur.



Le potentiomètre agit comme un diviseur de tension réglable. En changeant la position du bouton du potentiomètre, vous modifiez la tension appliquée sur A0. Plus vous le tournez vers la gauche, plus la tension sera proche de 0V. Plus vous le tournez vers la droite, plus la tension sera proche de 3,3V.



Une pin d'entrée analogique peut être utilisée pour lire une valeur comprise entre 0 et 1023. Cette valeur est proportionnelle à la tension appliquée à la pin, qui doit être comprise entre 0V et 3,3V.

Le bloc **analogWrite** est utilisé pour allumer la LED sur D9. En utilisant **analogWrite**, la carte est capable de limiter la tension à une certaine valeur pour rendre la LED plus ou moins brillante. La luminosité est définie par la valeur d'**analogRead** sur la *pin A0* : plus la valeur est élevée, plus la LED est lumineuse.



En utilisant la pin D9, nous sommes en mesure d'écrire une valeur analogique via une broche numérique sur la carte. La pin D9, comme quelques autres pin de la carte, supporte la modulation de largeur d'impulsion (Pulse Width Modulation ou PWM). Cette technique utilise des fonctionnements tout ou rien pour simuler différentes tensions et donc différents signaux analogiques. La valeur passée à **analogWrite doit être comprise entre 0 et 255. 0 correspond à une tension de 0V et 255 à 3.3V.**

Comme vous le verrez en utilisant ce programme, vous n'utiliserez pas la gamme complète du potentiomètre. Vous pouvez transformer la plage de valeurs du potentiomètre (0...1023) en plage de valeurs du PWM (0...255) avec la fonction **map**.

CODE MORSE

#R1AS06

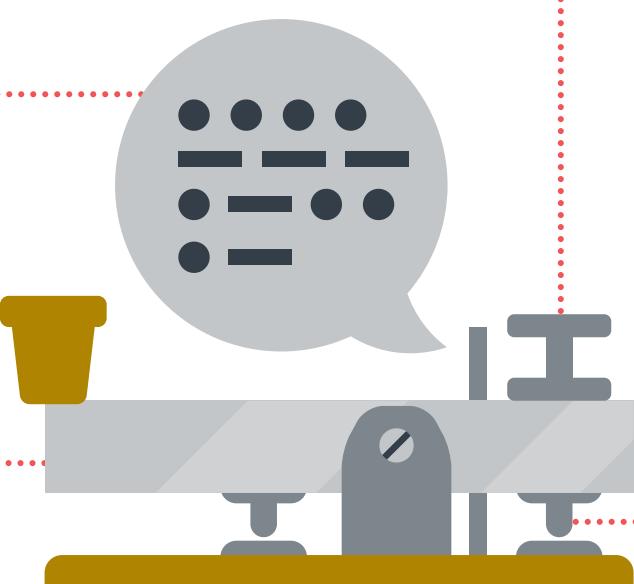


Disponible sur



Prérequis

- R1AS02 - Breadboard
- R1AS03 - Boutons et affichage LED



De quoi parle-t-on ?

Le code Morse est une méthode utilisée dans les télécommunications pour coder les caractères d'un texte sous forme de séquences normalisées de deux durées de signal différentes, appelées points et tirets.

Ressource :
<https://en.wikipedia.org/wiki/Telecommunication>

Durée

30 minutes

Niveau de difficulté

Avancé

Matériel

- 1 carte programmable "**STM32 IoT Node Board**"
- 1 câble USB Micro-B
- 1 breadboard
- 1 buzzer piézoélectrique ou un haut-parleur
- 2 boutons
- Câbles de connexion

OBJECTIFS D'APPRENTISSAGE

- Câbler et utiliser un buzzer passif
- Communiquer avec le code morse

CODE MORSE



Des micro-ondes aux jeux télévisés, les buzzers sont partout autour de nous et permettent de signaler quelque chose par un signal sonore. Pour émettre un son (ou un bruit), le buzzer contient une fine membrane (en quartz), qui vibre à une fréquence donnée (entre 20Hz et 20 000Hz, qui sont les fréquences audibles). Dans cette fiche d'activité, vous allez fixer quelques boutons et un buzzer à la carte et apprendre à communiquer en **Morse** !

Ressources : <https://en.wikipedia.org/wiki/Buzzer>, https://en.wikipedia.org/wiki/Morse_code

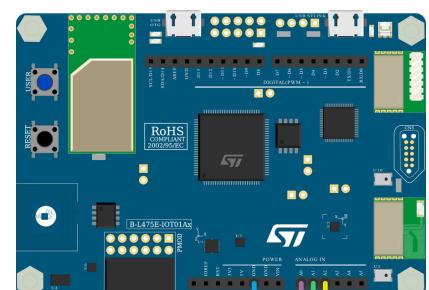


ÉTAPE 1 - CONSTRUIRE



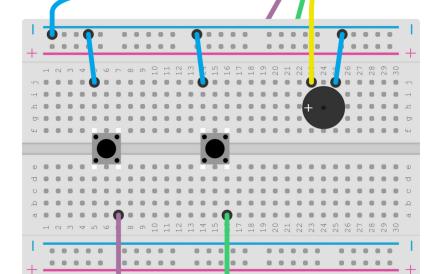
Câbler le buzzer

En théorie, un buzzer ou un haut-parleur n'est pas polarisé (cela signifie qu'il n'y a pas de "+" ni de "-"), mais il y a souvent une paire de fils noir/rouge ou des signes ("+" et/ou "-") sur l'appareil. Si vous êtes dans cette configuration, attachez le fil du côté "+" du buzzer à la **pin D3** et l'autre à la **pin GND**. S'il n'y a pas de couleur ou d'indication, branchez simplement un fil sur la **pin D3** et l'autre sur la broche **GND**.



Câbler les boutons

Connectez un côté de chaque bouton à la **pin GND** de la carte. Puis attachez les autres côtés, sur la **pin A0** (bouton 1), et la **pin A1** (bouton 2).



Connectez la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin droit de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

Câblage du buzzer et des boutons

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre travail est prêt !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Essayez de comprendre l'exemple et commencez à le modifier.



ÉTAPE 2 - PROGRAMMER



```
// Envoyer un signal court
input.buttonA0.onEvent(ButtonEvent.Click, function () {
    music.playTone(440, 100)
})

// Envoyer un signal long
input.buttonA1.onEvent(ButtonEvent.Click, function () {
    music.playTone(440, 300)
})
```

Comment cela fonctionne-t-il ?

Le code est vraiment simple ! Vous pouvez voir les deux fonctions **onEvent** qui permettent de détecter quand un bouton est pressé.

Ensuite, nous avons simplement utilisé la fonction **music.playTone**, avec 2 paramètres :

- **440** : la fréquence que nous voulons jouer
- **100 ou 300** : la durée de la tonalité en millisecondes (1 seconde = 1 000 millisecondes)

Maintenant que vous avez compris les bases, nous allons envoyer un message en Morse !

Signalisation en Morse

Le code Morse est une méthode de communication qui code les caractères sous la forme d'une **séquence de deux signaux de durées différentes**, appelés **points et tirets**.

Un **point** est un **signal court** tandis qu'un **tiret** est un **signal plus long**. En combinant plusieurs séquences, on peut transmettre un message composé de plusieurs mots. Le code Morse peut être émis de différentes manières : à l'aide d'une lampe (flash), d'une radio ou d'une carte comme celle que vous avez !

La figure de droite donne un aperçu de la manière de coder chaque caractère en Morse. Essayez d'envoyer "SOS" à quelqu'un !

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● —	U	● ● —
B	— ● ● ●	V	● ● ● —
C	— ● ● —	W	● — —
D	— — ●	X	— ● ● —
E	●	Y	— ● — —
F	● ● — —	Z	— — — ●
G	— — —		
H	● ● ●		
I	● ●		
J	● — — —		
K	— ● —		
L	— ● ● —		
M	— —		
N	— — ●		
O	— — —		
P	● — —		
Q	— — ● —		
R	— ● — —		
S	● ● ●		
T	—		
1	● — — — —	U	● ● — — — —
2	● ● — — —	V	● ● ● — — —
3	● ● ● ● — —	W	● — — — — —
4	● ● ● ● ● —	X	— ● ● — — — —
5	● ● ● ● ●	Y	— ● — — — — —
6	● ● ● ● ● —	Z	— — — — — — —
7	● ● ● ● ● — —		
8	● ● ● ● ● — — —		
9	● ● ● ● ● — — — —		
0	● ● ● ● ● — — — — —		

MUSIQUE

CRÉONS UNE MÉLODIE

#R1AS07



Disponible sur



Prérequis

- R1AS02 - Breadboard
- R1AS06 - Code Morse

De quoi parle-t-on ?

Créons une mélodie agréable à nos oreilles, inspirée des consoles 8 bits.

Durée

30 minutes

Niveau de difficulté

Avancé

Matériel

- 1 carte programmable "**STM32 IoT Node Board**"
- 1 câble USB Micro-B
- 1 set de LED
- 1 set de résistance
- 1 breadboard
- Câbles de connexion

OBJECTIFS D'APPRENTISSAGE

- Jouer de la musique avec une carte programmable



Alors que nous faisons beaucoup de bruits à l'aide de buzzers et de haut-parleurs dans diverses fiches d'activité, comme lors de la fabrication d'un thétrémine avec un capteur de distance ou le jeu de questions-réponses avec des boutons et des LED, voyons ce qui peut être fait pour créer une mélodie plus agréable pour les oreilles. Nous allons apprendre à jouer quelques notes et tonalités à l'aide d'un programme pour diffuser une mélodie bien connue. Pour rester dans l'ambiance des sons électroniques, nous commencerons par une musique inspirée des consoles 8 bits. Le **chiptune**, également connu sous le nom de musique chip ou musique 8 bits, est un style de musique électronique synthétisée réalisée à l'aide des puces sonores ou synthétiseurs PSG (Programmable sound generator ou générateur de son programmable) des machines d'arcade, ordinateurs et consoles de jeux vidéo vintage.

Resource: <https://en.wikipedia.org/wiki/Chiptune>



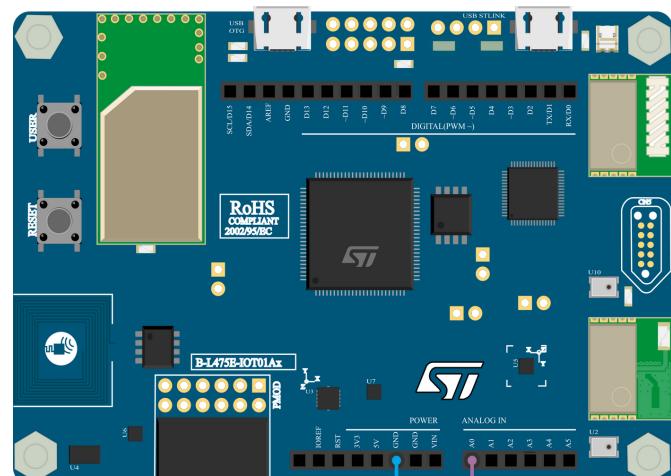
ÉTAPE 1 - CONSTRUIRE



Câbler le buzzer/haut-parleur

En théorie, un buzzer ou un haut-parleur n'est pas polarisé (cela signifie qu'il n'y a pas de "+" ni de "-"), mais il y a souvent une paire de fils noir/rouge ou des signes ("+" et/ou "-") sur l'appareil. Si vous êtes dans cette configuration, attachez le fil du côté "+" du buzzer à la **pin D3** et l'autre à la **pin GND**. S'il n'y a pas de couleur ou d'indication, branchez simplement un fil sur la **pin D3** et l'autre sur la **pin GND**.

1



Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

2

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

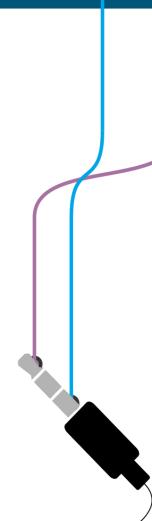
3

Installer l'extension

Après avoir créé votre nouveau projet, vous obtiendrez l'écran par défaut "prêt à l'emploi" illustré ici et vous devrez installer une extension.

4

Simulateur MakeCode



MUSIQUE - CRÉONS UNE MÉLODIE



ÉTAPE 1 - CONSTRUIRE



Les extensions dans MakeCode sont des groupes de blocs de code qui ne sont pas directement inclus dans les blocs de code de base que l'on trouve dans MakeCode. Les extensions, comme leur nom l'indique, ajoutent des blocs pour des fonctionnalités spécifiques. Il existe des extensions pour un large éventail de fonctionnalités très utiles, ajoutant des capacités de manette de jeu, de clavier, de souris, de servomoteurs, de la robotique et bien plus encore.

Vous voyez le bouton noir **AVANCÉ** en bas de la colonne des différents groupes de blocs. Si vous cliquez sur **AVANCÉ**, vous verrez apparaître des groupes de blocs supplémentaires. En bas, il y a une boîte grise appelée **EXTENSIONS**. Cliquez sur ce bouton.

Dans la liste des extensions disponibles, vous pouvez facilement trouver l'extension **Music** qui sera utilisée pour cette activité. Si elle n'est pas directement disponible sur votre écran, vous pouvez la rechercher à l'aide de l'outil de recherche. Cliquez sur l'extension que vous souhaitez utiliser et un nouveau groupe de blocs apparaîtra sur l'écran principal.

Programmer la carte

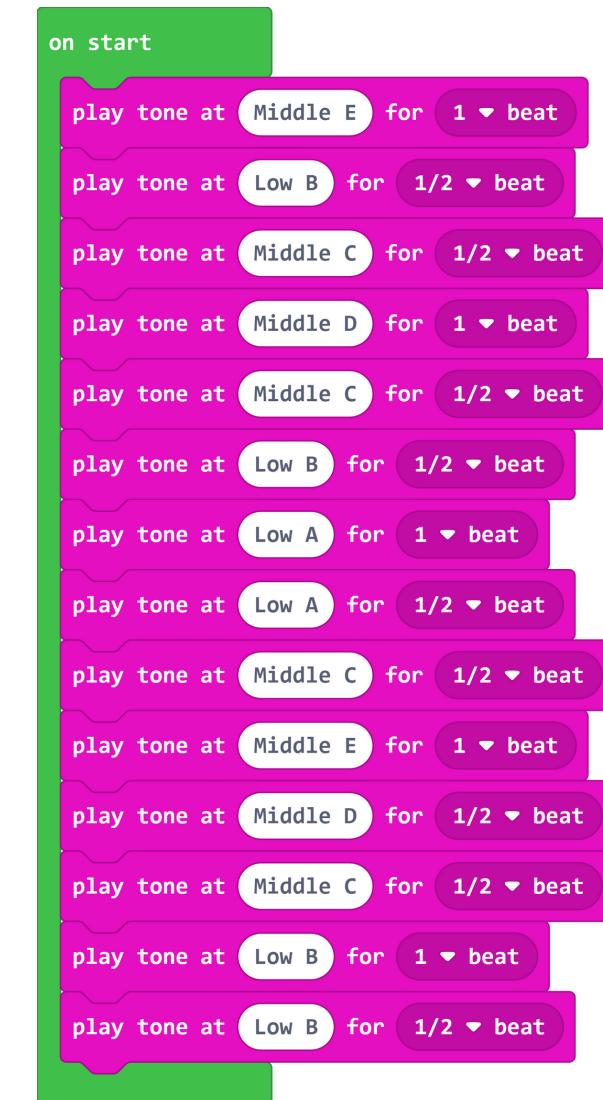
Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre programme est prêt !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Essayez de comprendre l'exemple et commencez à le modifier en changeant la période entre deux notes.



L'éditeur MakeCode avec l'extension Musique



Blocs permettant l'exécution du programme

**ETAPE 2 - PROGRAMMER**

```
music.playTone(330, music.beat(BeatFraction.Whole))
music.playTone(247, music.beat(BeatFraction.Half))
music.playTone(262, music.beat(BeatFraction.Half))
music.playTone(294, music.beat(BeatFraction.Whole))
music.playTone(262, music.beat(BeatFraction.Half))
music.playTone(247, music.beat(BeatFraction.Half))
music.playTone(220, music.beat(BeatFraction.Whole))
music.playTone(220, music.beat(BeatFraction.Half))
music.playTone(262, music.beat(BeatFraction.Half))
music.playTone(330, music.beat(BeatFraction.Whole))
music.playTone(294, music.beat(BeatFraction.Half))
music.playTone(262, music.beat(BeatFraction.Half))
music.playTone(247, music.beat(BeatFraction.Whole))
music.playTone(247, music.beat(BeatFraction.Half))
music.playTone(262, music.beat(BeatFraction.Half))
music.playTone(294, music.beat(BeatFraction.Whole))
music.playTone(330, music.beat(BeatFraction.Whole))
music.playTone(262, music.beat(BeatFraction.Whole))
music.playTone(220, music.beat(BeatFraction.Whole))
music.playTone(220, music.beat(BeatFraction.Whole))
music.playTone(294, music.beat(BeatFraction.Whole))
music.playTone(349, music.beat(BeatFraction.Half))
music.playTone(440, music.beat(BeatFraction.Half))
music.playTone(440, music.beat(BeatFraction.Half))
music.playTone(392, music.beat(BeatFraction.Half))
music.playTone(349, music.beat(BeatFraction.Half))
music.playTone(330, music.beat(BeatFraction.Whole))
music.playTone(262, music.beat(BeatFraction.Whole))
music.playTone(330, music.beat(BeatFraction.Whole))
music.playTone(294, music.beat(BeatFraction.Half))
music.playTone(262, music.beat(BeatFraction.Half))
music.playTone(247, music.beat(BeatFraction.Whole))
music.playTone(247, music.beat(BeatFraction.Half))
```



ETAPE 2 - PROGRAMMER



Comment cela fonctionne-t-il ?

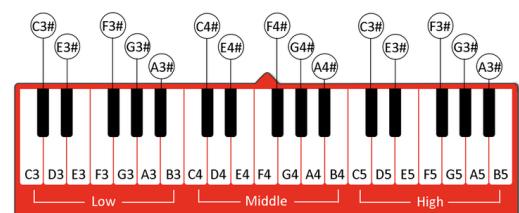
Ce programme représente une séquence de notes avec une durée. La compréhension de cette activité est davantage liée à la musique qu'à la programmation.

La bibliothèque musicale intégrée à MakeCode nous permet de jouer de la musique sur notre carte. Pour jouer une note, nous utilisons la commande suivante :  Où Middle C = note et 1 beat = 1 temps.

Transcription de chansons à partir de partitions

Si nous voulons recréer nos chansons préférées, nous devons d'abord avoir une connaissance de base des partitions. Voici un rappel des notes les plus courantes utilisées dans une partition musicale :

Pour choisir la bonne note sur MakeCode, vous pouvez cliquer sur le nom de la note et faire apparaître le piano virtuel. Chaque touche correspond à une note spécifique :



Durée de la note

Si nous regardons à nouveau les notes d'une partition de musique, vous remarquerez qu'elles ont des formes et des couleurs différentes. Ces différentes formes et couleurs indiquent les différentes durées appelées valeur de note et exprimées en nombre de temps (beat).

Notes	Name	Value	Code
○	Semibreve Whole note	4 beat	4 ▾ beat
—	Minim Half note	2 beat	2 ▾ beat
♩	Crotchet Quarter note	1 beat	1 ▾ beat
♪ ♪	Quaver Eighth note	1/2 beat	1/2 ▾ beat
♪	Semiquaver Sixteenth note	1/4 beat	1/4 ▾ beat

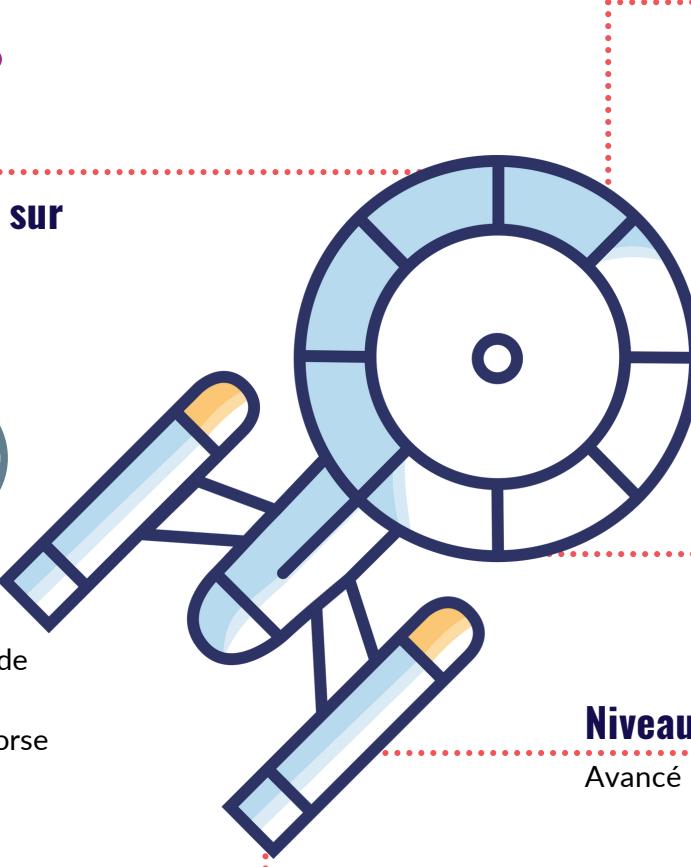
FABRIQUER UN THÉRÉMINE

AVEC LE CAPTEUR DE DISTANCE

#R1AS08

**Disponible sur****Prérequis**

- R1AS04 - DéTECTEUR de lumière basique
- R1AS06 - Le code Morse

**De quoi parle-t-on ?**

Un thérémne est un instrument de musique électronique dont on peut jouer sans le toucher. Le concept original est basé sur l'utilisation de deux antennes pour détecter la position des mains. Une antenne est utilisée pour le volume, et l'autre pour la hauteur du son.

Durée

30 minutes

Niveau de difficulté

Avancé

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 buzzer piézoélectrique ou un haut-parleur
- 1 breadboard
- Câbles de connexion

OBJECTIFS D'APPRENTISSAGE

- Utiliser un capteur de distance et comprendre son fonctionnement
- Faire de la musique avec un instrument vraiment étrange
- Utiliser la fonction map pour transformer un nombre d'une plage à une autre



FABRIQUER UN THÉRÉMINE AVEC LE CAPTEUR DE DISTANCE



Le thérémine est un instrument de musique électronique contrôlé sans contact physique par le théréministe (interprète). Il doit son nom à son inventeur, Léon Thérémine, qui a fait breveter l'appareil en 1928. La section de contrôle de l'instrument se compose généralement de deux antennes métalliques qui détectent la position relative des mains du théréministe et contrôlent les oscillateurs pour la fréquence avec une main et l'amplitude (volume) avec l'autre. Les signaux électriques émis par le thérémine sont amplifiés et envoyés à un haut-parleur.

Notre version sera plus simple, nous ne contrôlerons que la hauteur du son, avec le capteur de distance, le volume sera prédéterminé. **Faisons de la musique !**

Ressources : <https://en.wikipedia.org/wiki/Theremin>, <https://youtu.be/x0NVb25p1oU>



ÉTAPE 1 - CONSTRUIRE



Câbler le buzzer/haut-parleur

En théorie, un buzzer ou un haut-parleur n'est pas polarisé (cela signifie qu'il n'y a pas de "+" ni de "-"), mais il y a souvent une paire de fils noir/rouge ou des signes ("+" et/ou "-") sur l'appareil. Si vous êtes dans cette configuration, attachez le fil du côté "+" du buzzer à la **pin D3** et l'autre à la **pin GND**. S'il n'y a pas de couleur ou d'indication, branchez simplement un fil sur la **pin D3** et l'autre sur la **pin GND**.

1

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

2

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

3

Programmer la carte

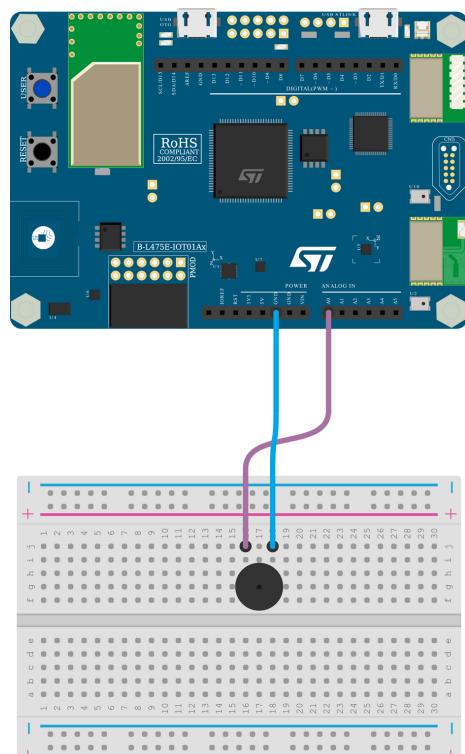
Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre programme est prêt !

4

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Essayez de comprendre l'exemple et commencez à le modifier.

5



Câbler le buzzer/haut-parleur

FABRIQUER UN THÉRÉMINE AVEC LE CAPTEUR DE DISTANCE



ÉTAPE 2 - PROGRAMMER



```

let distance = 0
forever(function () {
    // Mesurer la distance
    distance = input.distance(DistanceUnit.Millimeter)

    if (distance > 500) {
        // Convertir la distance en fréquence
        let note = Math.map(distance, 0, 500, 440, 830)
        music.ringTone(note)
    } else {
        music.stopAllSounds()
    }
})

```

Dans ce programme, il y a 2 variables. La première, **distance**, est utilisée pour mémoriser la distance et déterminer la note à jouer. Ensuite, il y a **note**, qui n'est pas techniquement nécessaire/obligatoire mais permet d'introduire une plus grande compréhension de chaque étape du programme. Elle contient la transformation de la distance en fréquence du ton.

Mesurer la distance

Utiliser une variable pour mémoriser la distance, c'est bien, mais savoir comment mesurer la distance, c'est mieux ! Cette opération ne présente aucune difficulté. Nous devons appeler la fonction **input.distance(DistanceUnit.Millimeter)**. Le paramètre **DistanceUnit.Millimeter** indique à la fonction que nous voulons le résultat en millimètres (1 mètre = 1 000 millimètres).

Condition

La condition **if (distance > 500) { ... }** donne l'information que nous ne jouons un son que si la distance mesurée est inférieure ou égale à 500 millimètres.

Convertir la distance en fréquence

La partie la plus importante est la conversion. Pour la réaliser, nous utilisons une fonction mathématique appelée **map**. Cette fonction transforme une valeur d'une plage à une autre. Dans ce cas, la valeur est transférée de la plage de distances à la plage de fréquences. Comme vous pouvez le voir dans le code ci-dessus, cette fonction prend cinq paramètres, à savoir : **value, in_min, in_max, out_min, out_max**. Examinons de plus près chacun d'entre eux :

- **value** : la valeur à transformer
- **in_min** : la valeur minimale de la plage d'entrée (distance)
- **in_max** : la valeur maximale de la plage d'entrée (distance)
- **out_min** : la valeur minimale de la plage de sortie (fréquence)
- **out_max** : la valeur maximale de la gamme de sortie (fréquence)

Nous pouvons donc comprendre ce que fait cette ligne, c'est-à-dire transformer la distance (avec une plage de 0mm à 500mm) en fréquence (avec une plage de 440Hz à 830Hz).



Les fréquences choisies ne sont pas aléatoires, la gamme de fréquence de 440Hz à 830Hz représente une octave. Cela signifie que vous pouvez trouver toutes les notes : LA, SI, DO, RE, MI, FA, SOL

Nous avons maintenant une fréquence. Il est temps de la jouer, en utilisant simplement la fonction **music.ringTone(note)**.

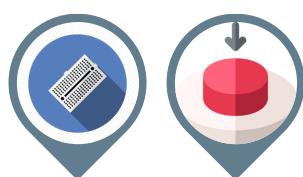
CAPTEUR D'INCLINAISON

AVEC L'ACCÉLÉROMÈTRE

#R1AS09

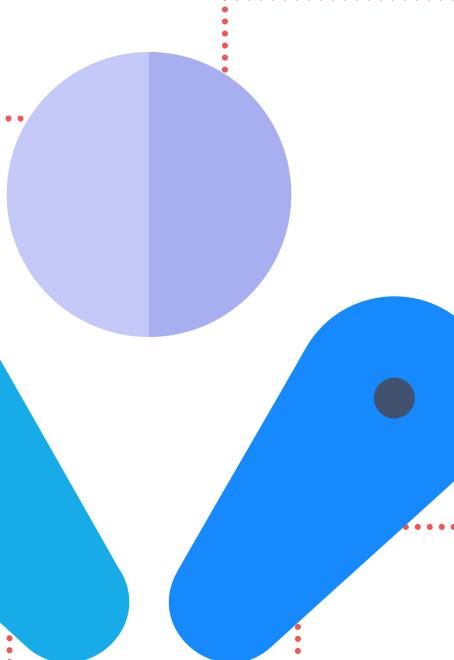


Disponible sur



Prérequis

- R1AS02 - Breadboard
- R1AS03 - Boutons et affichage LED



De quoi parle-t-on ?

Les accéléromètres sont de petits capteurs qui peuvent détecter la force de l'accélération et sont parfaits pour détecter le mouvement et l'orientation.

Durée

30 minutes

Niveau de difficulté

Avancé

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 set de LED
- 1 set de résistances
- 1 breadboard
- Câbles de connexion

OBJECTIFS D'APPRENTISSAGE

- Utiliser un accéléromètre en lisant la valeur de l'accélération sur chaque axe
- Réagir aux secousses
- Déetecter une situation de chute libre

CAPTEUR D'INCLINAISON AVEC L'ACCÉLÉROMÈTRE



L'accélération fait tourner le monde - littéralement ! C'est la force qui provoque le mouvement, comme une voiture qui accélère ou un objet qui tombe sur le sol sous l'effet de la gravité lorsqu'on le lâche.

Pour découvrir le potentiel de ce capteur de mouvement, nous allons écrire un capteur d'inclinaison qui allume une LED lorsque l'accélération est trop forte. Ce genre de dispositif est utile si vous voulez éviter la tricherie sur un **vieux flipper classique**.

Resource: <https://en.wikipedia.org/wiki/Pinball>

L'accéléromètre 3 axes est déjà intégré à la carte, vous n'avez donc pas besoin de connecter quoi que ce soit pour l'utiliser !



ÉTAPE 1 - CONSTRUIRE



Câbler trois LED sur la carte

À l'aide d'une breadboard, connectez trois LED simples aux pin de la carte :

- **LED verte** vers la **pin A0**
- **LED bleu** à la **pin A1**
- **LED rouge** vers la **pin A2**

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

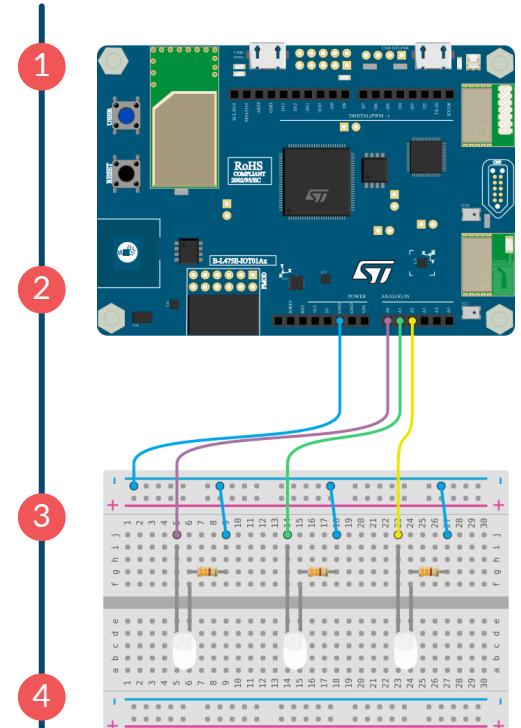
Ressource : makecode.lets-steam.eu

Programmer votre carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Essayez de comprendre l'exemple et commencez à le modifier en changeant les seuils pour calibrer votre capteur d'inclinaison. Pour tester votre capteur d'inclinaison, posez la carte sur une table et donnez un petit coup de pied à la table. Si votre voyant s'allume, l'accélération de votre coup de pied est assez forte !



Câbler trois LED sur la carte

CAPTEUR D'INCLINAISON AVEC L'ACCÉLÉROMÈTRE



ÉTAPE 2 - PROGRAMMER



```

function turnOffLEDs() {
    pins.A0.digitalWrite(false) // Green
    pins.A1.digitalWrite(false) // Blue
    pins.A2.digitalWrite(false) // Red
}

forever(function () {
    turnOffLEDs()
    // Axe X : vert L
    if (Math.abs(input.acceleration(Dimension.X)) > 700)
        pins.A0.digitalWrite(true)
    // Axe Y : LED bleue
    if (Math.abs(input.acceleration(Dimension.Y)) > 700)
        pins.A1.digitalWrite(true)
    // Axe Z : LED rouge
        if (Math.abs(input.acceleration(Dimension.Z)) > 700)
            pins.A2.digitalWrite(true)
    pause(500)
})

```

Comment cela fonctionne-t-il ?

Le programme consiste à allumer une LED sur l'axe sur lequel est détectée l'accélération (-1g) due à la gravité.

i La force G d'un objet est son accélération par rapport à une chute libre. Sur terre, elle est de 1G, soit 9,8 mètres par seconde au carré (m/s^2). Les astronautes subissent des forces G exceptionnellement élevées et exceptionnellement faibles. La force G peut également être observée sur les montagnes russes. Lorsque les wagonnets de montagnes russes dévalent la pente, vous êtes repoussé dans votre siège en raison de la force G.

Voici la configuration des axes d'accélération / couleurs des LED :

- Axe X : LED verte
- Axe Y : LED bleue
- Axe Z : LED rouge

Lire la valeur de l'accélération

Pour lire la valeur de l'accélération, MakeCode fournit la fonction **acceleration()**. La valeur est par défaut en mG. Nous utilisons la fonction valeur absolue **abs()** pour ignorer la direction de l'accélération. Pour détecter la condition "tilt", nous utilisons un seuil de 700mG. Pour éteindre les trois LED en même temps et améliorer l'expressivité de notre code, nous définissons une fonction **turnOffLED()**.

i Une fonction est un bloc de code qui exécute une tâche spécifique. Comme une variable, elle a un nom qui permettra de l'utiliser à plusieurs endroits dans votre programme. Il est très utile de simplifier le code et de rendre un bloc de code plus lisible en lui donnant un nom qui explique votre intention.

AFFICHAGE DE TEXTE

AVEC UN ÉCRAN OLED

#R1AS10



Available on

**Prérequis**

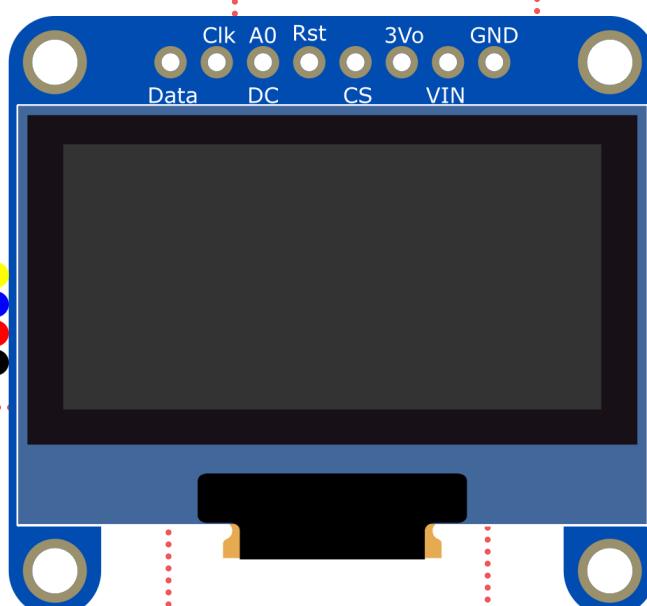
- R1AS03 - Boutons et affichage LED

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 écran OLED Monochrome 1.3" 128x64 OLED de Adafruit
- 1 câble QT pour connecter l'écran à la carte

De quoi parle-t-on ?

Un écran qui vous permet d'afficher certaines informations nichées dans vos composants électroniques.

**Durée**

30 minutes

Niveau de difficulté

Avancé

OBJECTIFS D'APPRENTISSAGE

- Connecter un écran LCD à votre carte
- Afficher du texte sur votre écran LCD
- Placer du texte sur un écran
- Afficher l'état actuel de votre programme



AFFICHAGE DE TEXTE AVEC UN ÉCRAN OLED



La programmation d'une carte électronique est parfois une activité très déroutante. Un microcontrôleur est une boîte noire (nous ne pouvons pas voir ce qui se passe à l'intérieur). Pour éclaircir votre code, vous pouvez utiliser un écran qui vous aide à afficher certaines informations nichées dans vos composants électroniques. Cette fiche d'activité explore comment utiliser les **écrans OLED monochromes à base de SSD1306 avec MakeCode**.

Ressource : <https://www.electronicwings.com/sensors-modules/ssd1306-oled-display>



ÉTAPE 1 - CONSTRUIRE



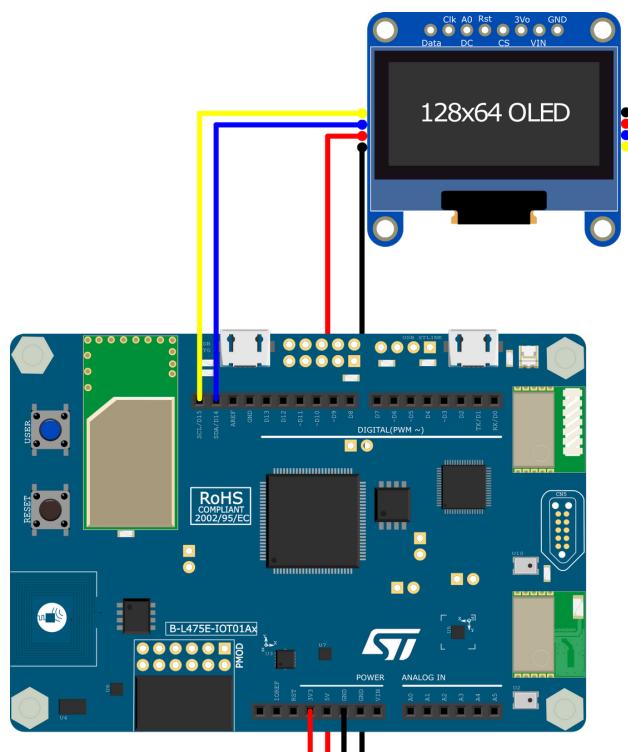
Connecter la carte à l'écran

Il y a deux façons de câbler l'écran **OLED SSD1306** à une carte, soit avec une connexion **I2C** ou **SPI**. Pour notre écran, nous utilisons la connexion **I2C** via le câble **QWIIC/STEMMA** avec la convention suivante :

- **Noir** pour **GND**
- **Rouge** pour **V+ (3V3)**
- **Bleu** pour **SDA (D14)**
- **Jaune** pour **SCL (D15)**

Ressources : <https://en.wikipedia.org/wiki/I2C>,
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface,
<https://www.sparkfun.com/qwiic>,
<https://learn.adafruit.com/introducing-adafruit-stemma-qt/what-is-stemma-qt>

1



Connecter la carte à l'écran

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

2

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

3

Installer l'extension

Après avoir créé votre nouveau projet, vous obtiendrez l'écran par défaut "prêt à l'emploi" illustré ici et vous devrez installer une extension.

4

AFFICHAGE DE TEXTE AVEC UN ÉCRAN OLED



ÉTAPE 1 - CONSTRUIRE



Les extensions dans MakeCode sont des groupes de blocs de code qui ne sont pas directement inclus dans les blocs de code de base que l'on trouve dans MakeCode. Les extensions, comme leur nom l'indique, ajoutent des blocs pour des fonctionnalités spécifiques. Il existe des extensions pour un large éventail de fonctionnalités très utiles, ajoutant des capacités de manette de jeu, de clavier, de souris, de servomoteurs, de la robotique et bien plus encore.

Vous voyez le bouton noir **AVANCÉ** en bas de la colonne des différents groupes de blocs. Si vous cliquez sur **AVANCÉ**, vous verrez apparaître des groupes de blocs supplémentaires. En bas, il y a une boîte grise appelée **EXTENSIONS**. Cliquez sur ce bouton.

Choisissez l'extension "**oled**".

5

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et que votre programme affiche du texte !

6

Exécuter, modifier, jouer

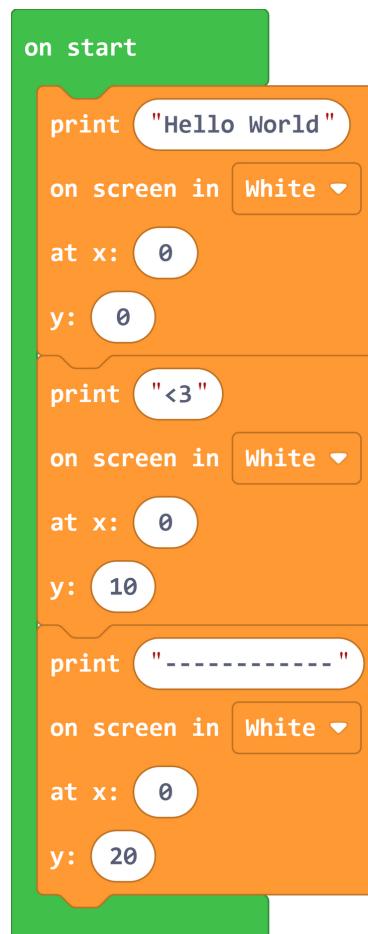
Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**).

Si tout va bien, votre carte vous adressera des salutations amicales. Essayez de comprendre l'exemple et commencez à le modifier en modifiant le texte, en ajoutant autant de symboles que possible ou en remplaçant l'écran lentement, lettre par lettre.

N'hésitez pas à essayer d'afficher n'importe quel élément d'information dans votre programme pour voir l'état actuel de votre carte.



Menu avancé comprenant l'outil
"Extentions"



Blocs permettant l'exécution du programme



ÉTAPE 2 - PROGRAMMER



```
oled.printString("Hello World", PixelColor.White, 0, 0)
oled.printString("<3", PixelColor.White, 0, 10)
oled.printString("-----", PixelColor.White, 0, 20)
```

Comment cela fonctionne-t-il ?

Vous pouvez écrire une ligne de texte avec la fonction **printString()**. Cette fonction prend les paramètres suivants :

- Chaîne de texte
- Couleur du texte (PixelColor.Black ou PixelColor.White)
- Position du texte sur l'axe des X
- Position du texte sur l'axe des Y

i Sur l'écran du SSD1306, l'origine (la position x=0 et Y=0) se trouve dans le coin supérieur gauche.

RESSOURCES - PROGRAMMATION - FICHE D'ACTIVITÉ 11

FABRIQUER UN THERMOMÈTRE

... TRÈS, TRÈS LISIBLE

#R1AS11



Disponible sur

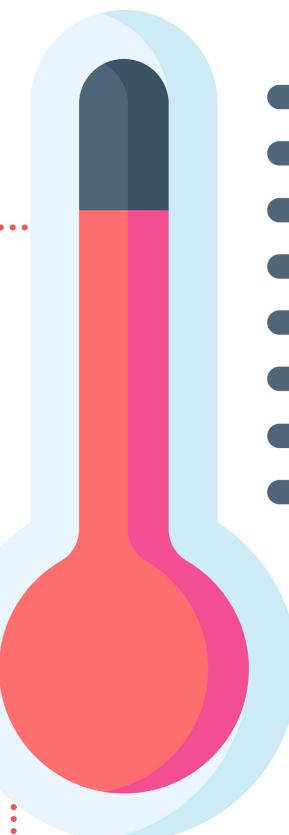


Prérequis

- R1AS04 - Découverte des capteurs de lumière

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 écran texte LCD Grove I2C
- 1 câble de liaison Grove



De quoi parle-t-on ?

Dans cette activité, nous allons apprendre à quel point il est facile de lire le capteur de température de la carte et d'afficher sa valeur.

Durée

20 minutes

Niveau de difficulté

Intermédiaire

OBJECTIFS D'APPRENTISSAGE

- Lire le capteur de température
- Utiliser un écran d'affichage LCD



FABRIQUER UN THERMOMÈTRE ... TRÈS, TRÈS LISIBLE



La température est une grandeur physique qui exprime le chaud et le froid. C'est la manifestation de l'énergie thermique, présente dans toute matière, qui est à l'origine de l'apparition de la chaleur. En les mettant en contact, les chocs entre particules font que cette énergie cinétique microscopique se transmet d'un corps à l'autre. C'est ce transfert d'énergie qui, en sciences physiques, est appelé chaleur. Dans cette activité, vous pourrez découvrir l'utilisation du capteur de température, intégré dans la carte. Un capteur de température est un dispositif électronique qui mesure la température de son environnement et convertit les données d'entrée en données électroniques pour enregistrer, surveiller ou signaler les changements de température.



ÉTAPE 1 - CONSTRUIRE



Connecter l'écran à la carte

Pour connecter l'écran LCD Grove, nous allons utiliser le bus **I2C**. Pour notre écran, nous utilisons la connexion **I2C** via le câble de liaison Grove avec la convention suivante :

- **Rouge** pour **V+ (3V3)**
- **Violet** pour **SDA (D14)**
- **Vert** pour **SCL (D15)**

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

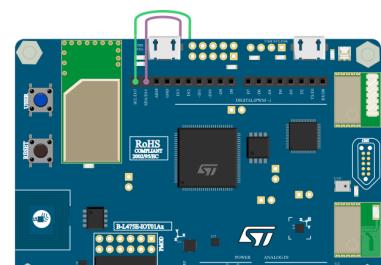
Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre datalogger est prêt !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Si tout fonctionne bien, votre carte changera l'état des LED pour montrer que la collecte de données est en cours. Essayez de comprendre l'exemple et commencez à le modifier en changeant la période entre deux mesures, en ajoutant d'autres données provenant d'autres capteurs de la carte. Essayez d'afficher autant de données que vous le souhaitez à plusieurs endroits pour comprendre comment la température évolue

1



2



3

Connecter l'écran à la carte

4

5

FABRIQUER UN THERMOMÈTRE ... TRÈS, TRÈS LISIBLE



ÉTAPE 2 - PROGRAMMER



```
lcd.clear()
forever(function () {
  lcd.setCursor(0, 0)
  lcd.ShowValue("T", input.temperature(TemperatureUnit.Celsius))
  pause(500)
})
```

Comment cela fonctionne-t-il ?

Le code se compose de :

- un bloc **clear screen**
- un bloc **forever**
- un bloc **set cursor position**
- un bloc **show value**

i L'écran LCD mémorise la position du prochain emplacement d'insertion. Lorsque nous voulons afficher quelque part à l'écran, nous devons toujours commencer par définir la position du curseur.

Avant d'écrire à l'écran, nous effaçons l'écran en appelant la fonction **LCD.clear()**.

À chaque itération de la boucle, avant d'écrire quelque chose, nous plaçons le curseur au début de l'écran (au premier caractère de la première ligne).

input.temperature(TemperatureUnit.Celsius) renvoie la valeur entière de la température en degrés Celsius. La valeur est affichée à l'écran avec la fonction **LCD.ShowValue()**. Le premier paramètre de cette fonction donne le nom de la valeur et le second, la valeur à afficher.

Simulation du capteur de température

Vous pouvez jouer avec le capteur simulé en appuyant sur la **petite icône de thermomètre** affichée sur le simulateur de la carte. Vous pouvez modifier la valeur détectée (par exemple, en touchant avec le doigt le capteur physique de la carte), ce qui modifie en conséquence la valeur affichée sur l'écran LCD.

ALARME DE DÉTECTION DE MOUVEMENT

#R1AS12



Disponible sur

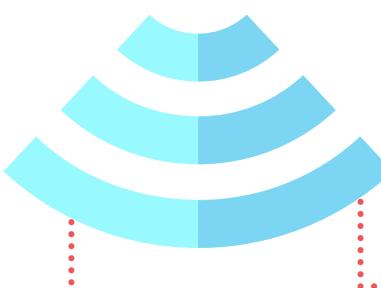
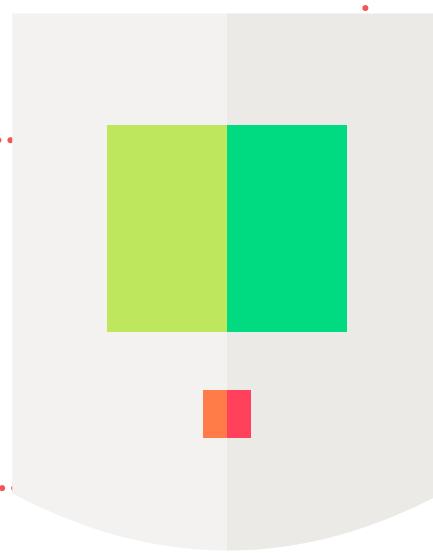


Prérequis

- R1AS09 - Capteur d'inclinaison avec l'accéléromètre
- R1AS07 - Fabriquer un thévémine avec le capteur de distance

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 breadboard
- Câbles de connexion
- 1 buzzer piézoélectrique ou un haut-parleur
- 1 petite boîte en carton de bricolage (environ 15x5 cm)



De quoi parle-t-on ?

Dans cette fiche d'activité, nous allons travailler sur un détecteur de mouvement comprenant une alarme avec 2 types de protections : protection contre l'ouverture par la force et détection d'ouverture.

Durée

30 minutes

Niveau de difficulté

Advanced

OBJECTIFS D'APPRENTISSAGE

- Utiliser le bloc événementiel pour le capteur de distance
- Utiliser le bloc événementiel pour le capteur de mouvement

ALARME DE DÉTECTION DE MOUVEMENT



Dans cette fiche d'activité, nous allons travailler sur un détecteur de mouvement, vous permettant de garder en sécurité tous vos objets précieux et importants. Pour les besoins de cette fiche d'activité, votre objet le plus précieux sera contenu dans une boîte. Nous allons créer une alarme avec 2 fonctions : 1) un déclencheur d'alarme lorsque la boîte est secouée, et 2) un déclencheur d'alarme lorsque quelqu'un ou quelque chose entre dans la boîte. Cela permettra de découvrir le détecteur de mouvement intégré et ses usages. Un détecteur de mouvement est un dispositif électrique qui utilise un capteur pour détecter un mouvement à proximité. Un tel dispositif est souvent intégré en tant que composant d'un système qui exécute automatiquement une tâche ou alerte un utilisateur en cas de mouvement dans une zone. Ils constituent un élément essentiel de la sécurité, du contrôle automatisé de l'éclairage, de la domotique et d'autres systèmes utiles.

Ressource : https://en.wikipedia.org/wiki/Motion_detector



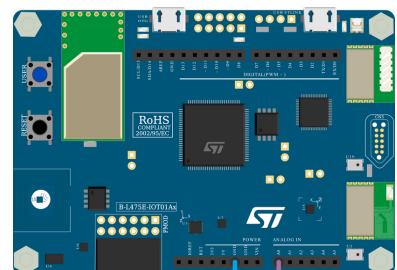
ÉTAPE 1 - CONSTRUIRE



Câbler le buzzer/haut-parleur

En théorie, un buzzer ou un haut-parleur n'est pas polarisé (cela signifie qu'il n'y a pas de "+" ni de "-"), mais il y a souvent une paire de fils noir/rouge ou des signes ("+" et/ou "-") sur l'appareil. Si vous êtes dans cette configuration, attachez le fil du côté "+" du buzzer à la **pin D3** et l'autre à la **pin GND**. S'il n'y a pas de couleur ou d'indication, branchez simplement un fil sur la **pin D3** et l'autre sur la **pin GND**.

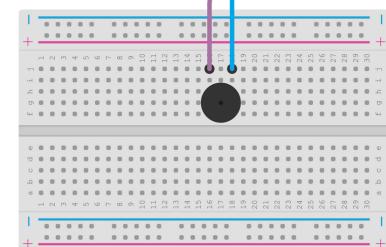
1



Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

2



Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

3

Programmer votre carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre alarme est prête.

4

Câbler le buzzer/haut-parleur

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Mettez votre carte programmable dans votre boîte et voyez la réaction lorsque vous la secouez ou l'ouvrez. Essayez de comprendre l'exemple et commencez à le modifier en changeant la distance de détection d'ouverture.

5



ÉTAPE 2 - PROGRAMMER



```

let isAlarmEnable = false

// Activation/désactivation de l'alarme en cas de pression sur le bouton "Utilisateur"
input.buttonUser.onEvent(ButtonEvent.Click, function () {
    isAlarmEnable = !(isAlarmEnable)
    pins.LED.digitalWrite(isAlarmEnable)
})

// Quand la carte tremble
input.onGesture(Gesture.Shake, function () {
    if (isAlarmEnable) {
        music.playTone(880, 3000)
    }
})

// Lorsque la distance est supérieure à 1 000 millimètres (1 mètre)
input.onDistanceConditionChanged(DistanceCondition.Far, 1000, DistanceUnit.Millimeter,
    function () {
        if (isAlarmEnable) {
            music.playTone(880, 3000)
        }
})

```

Comment cela fonctionne-t-il ?

Ce programme est une simple réutilisation de ce qui a déjà été appris dans les fiches d'activités précédentes. Comme vous pouvez le constater, il y a 3 parties en plus d'une variable permettant de connaître l'état de l'alarme. Nous allons les détailler ci-dessous :

Activer/désactiver l'alarme

Le premier bloc vise à détecter quand le bouton intégré est pressé. Lorsque cet événement se produit, nous inversons l'état de l'alarme : **isAlarmEnable = !(isAlarmEnable)**.

Détection des secousses

Lorsque la carte est secouée, si l'alarme est activée (**if (isAlarmEnable) {...}**), cela signifie que quelqu'un essaie de forcer notre boîte, nous devons donc faire sonner l'alarme (**startAlarm**) !

Détection d'ouverture

Considérons que votre boîte est fermée. La distance entre l'objet à l'intérieur de la boîte et le couvercle est presque nulle. Lorsque quelqu'un ouvre votre boîte, votre objet n'est plus en contact direct avec le couvercle. Dans ce cas, la distance entre votre précieux trésor et l'objet le plus proche sera plus élevée que précédemment. Vous pouvez alors détecter l'ouverture de votre boîte en évaluant le changement de la variable de distance (**onDistanceConditionChanged**). Cela permettra, lorsqu'on détectera une distance supérieure à 1000 millimètres (cette distance peut être modifiée) avec votre alarme activée, d'identifier que quelqu'un a ouvert le contenant et que l'alarme doit sonner (**startAlarm**) !

LES SERVOS FONT BOUGER LES CHOSES !

#R1AS13

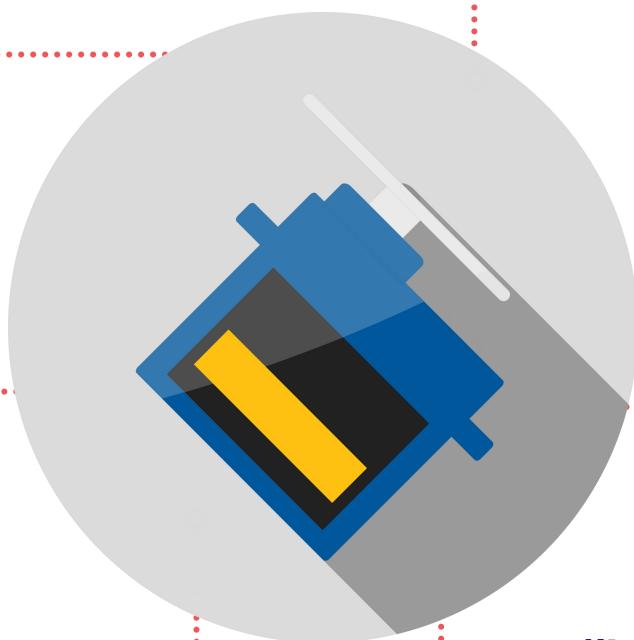


Disponible sur



Prérequis

- R1AS03 - Boutons et affichage LED



De quoi parle-t-on ?

Le servomoteur est un moteur pilotable en position. Il permet de conserver un angle précisément. Il convient pour contrôler un système dont l'angle peut changer au cours du fonctionnement et doit maintenir cette position.

Durée

25 minutes

Niveau de difficulté

Intermédiaire

Matériel

- 1 carte programmable "**STM32 IoT Node Board**"
- 1 câble USB Micro-B
- 1 mini-servo SG-90 (1,6 kg)
- Câbles de connexion

OBJECTIFS D'APPRENTISSAGE

- Mettre un objet en mouvement

LES SERVOS FONT BOUGER LES CHOSES !



Un servomoteur est un moteur doté d'un ensemble de systèmes de contrôle automatique, qui se compose d'un **moteur ordinaire à courant continu** (moteur électrique rotatif qui convertit l'énergie électrique (courant continu) en énergie mécanique), d'un réducteur, d'un **potentiomètre** (diviseur de tension utilisé pour mesurer le potentiel ou la tension électrique) et d'un circuit de contrôle. Il peut définir l'angle de rotation de l'arbre de sortie en envoyant des signaux. En général, un servomoteur a un angle de rotation maximal (par exemple, 180 degrés). Les servomoteurs sont commandés par l'intermédiaire d'un **câble électrique à trois fils** qui permet d'alimenter le moteur et de lui transmettre des consignes de position sous forme d'un signal codé en largeur d'impulsion plus communément appelé PWM. Cela signifie que c'est la durée des impulsions qui détermine l'angle absolu de l'axe de sortie et donc la position du bras de commande du servomoteur. Le cycle d'un signal de référence servomoteur est de 20 ms. Sur le servomoteur proposé, la position du bras varie de 0 à 180°, pour des largeurs d'impulsion allant de 1 à 2 ms (respectivement).

Ressources : https://en.wikipedia.org/wiki/DC_motor, <https://en.wikipedia.org/wiki/Potentiometer>



ÉTAPE 1 - CONSTRUIRE



Connecter le servomoteur à la carte

Il y a plusieurs façons de connecter un servomoteur à votre carte. Vous pouvez utiliser n'importe quelle pin de sortie analogique (broches PWM) pour connecter la pin de contrôle. Dans notre exemple, nous allons utiliser la pin **D4**. Le servomoteur sera connecté comme suit :

- **Noir** pour **GND**
- **Rouge** pour **V+ (3V3)**
- **Orange** pour **SIG (D4)**

Connecter la carte à l'ordinateur

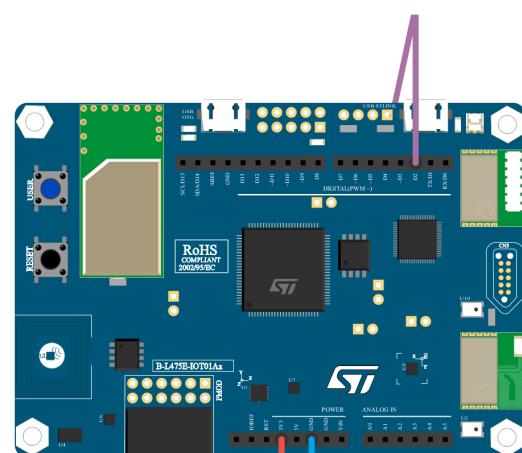
Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

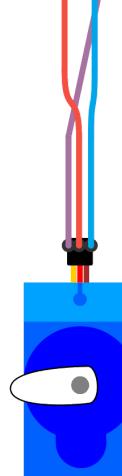
Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

1



2



3

Connecter le servomoteur à la carte

LES SERVOS FONT BOUGER LES CHOSES !



ÉTAPE 1 - CONSTRUIRE



Après avoir créé votre nouveau projet, vous accédez à l'éditeur par défaut ci-dessous.

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "Programmer" ci-dessous.

Avant d'essayer ce programme sur la carte, vous pouvez l'essayer directement dans le simulateur. Si vous changez les valeurs 0 et 180, vous verrez directement le résultat.

Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre servomoteur commencera à bouger !

Exécuter, modifier, jouer

Votre programme s'exécute automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**).

Si tout fonctionne bien, votre servomoteur va commencer à bouger.

Essayez de comprendre l'exemple et commencez à le modifier en changeant la période entre les deux mouvements.

4

```
au démarrage toujours
    au démarrage
        pause(1000)
    toujours
        pins.D2.servoWrite(180)
        pause(1000)
        pins.D2.servoWrite(0)
        pause(1000)
```

Editeur par défaut

5

```
// aller de 0 à 180 degrés
forever(function () {
    // dire au servomoteur d'aller à la position 180 degrés
    pins.D2.servoWrite(180)
    // attendre que le servomoteur atteigne la position
    pause(1000)
    // dire au servomoteur d'aller à la position 0 degré
    pins.D2.servoWrite(0)
    // attendre que le servomoteur atteigne la position
    pause(1000)
})
```

Editeur Javascript Makecode

```
toujours
    servo write pin D2 to 180
    pause(1000 ms)
    servo write pin D2 to 0
    pause(1000 ms)
```

Votre servomoteur se met à bouger

LES SERVOS FONT BOUGER LES CHOSES !



ÉTAPE 2 - PROGRAMMER



```
// aller de 0 à 180 degrés
forever(function () {
    // dire au servomoteur d'aller à la position 180 degrés
    pins.D4.servoWrite(180)
    // attendre que le servomoteur atteigne la position
    pause(1000)
    // dire au servomoteur d'aller à la position 0 degré
    pins.D4.servoWrite(0)
    // attendre que le servomoteur atteigne la position
    pause(1000)
})
```

Comment cela fonctionne-t-il ?

Cet échantillon est assez simple puisqu'il s'agit du classique clignotement adapté à un servomoteur.

L'instruction principale est **pins.D2.servoWrite(XXX)**. Cette instruction demande au servomoteur de tourner jusqu'à un angle de XXX degrés (comme défini par vos besoins spécifiques en fonction du projet que vous développez).

Pour passer d'une position à l'autre, le servomoteur prend un certain temps. Il faut donc toujours ajouter un délai avant de commencer un autre mouvement.

Ce programme ne fait que balayer de gauche à droite sans s'arrêter !

i Par rapport à un moteur à courant continu ordinaire, un servomoteur ne tourne que dans une certaine plage d'angles, alors qu'un moteur à courant continu ordinaire (CC) tourne sur un cercle.

Un servomoteur ne peut pas tourner sur un cercle. Un moteur CC ordinaire ne peut pas nous donner un retour sur l'angle de rotation, mais un servomoteur peut le faire. Leurs utilisations sont donc différentes.

Les moteurs CC ordinaires s'utilisent comme source d'énergie rotative, tandis que les servomoteurs s'utilisent pour donner un certain angle à un objet qu'ils contrôlent, comme faire bouger une articulation de robot.

CRÉER UN MINUTEUR POUR LES ŒUFS

#R1AS14



Disponible sur



Prérequis

- R1AS13 - Les servos font bouger les choses !



Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 mini-servo SG-90 (1,6 kg)
- Câbles de connexion
- 1 petite feuille de carton (20cm*10cm)
- Des bâtons de bois robustes (moins de 10 cm)

De quoi parle-t-on ?

Créons un objet simple mais utile, un minuteur à œufs ! Cette activité permettra d'appliquer les connaissances acquises sur les servomoteurs, sur un problème du quotidien.

Durée

35 minutes

Niveau de difficulté

Avancé

Activité étendue



OBJECTIFS D'APPRENTISSAGE

- Créer une minuterie
- Utiliser un servomoteur pour afficher des données
- Effectuer un processus d'étalonnage pour améliorer la précision du chronomètre



CRÉER UN MINUTEUR POUR LES ŒUFS



Dans cette activité, nous allons créer un objet simple mais utile, une minuterie, en utilisant la programmation et la pratique du bricolage ! Après l'avoir réalisé, vous serez un vrai grand chef ! Pour faire bouillir correctement un œuf, nous utilisons la **règle appelée 3,6,9** ! Cette règle donne la durée correcte de cuisson exacte en minutes d'un œuf en fonction de vos objectifs de cuisson :

- 3 minutes pour les **œufs à la coque**
- 6 minutes pour les **œufs mollets**
- 9 minutes pour les **œufs durs**



ETAPE 1 - CONSTRUIRE



Préparer le matériel électronique

Câblez correctement votre carte et votre servomoteur en utilisant la fiche d'activité #R1AS13 - Les servos font bouger les choses !

Créer l'aiguille de l'horloge et la fixer

Prenez les bâtons de bois robustes et fixez-les au bras du servomoteur.

i Les bras de servomoteurs sont des accessoires qui s'adaptent sur l'arbre de sortie et vous permettent de relier mécaniquement la sortie du servomoteur au reste de votre mécanisme. Les servomoteurs sont généralement fournis avec un assortiment de bras de servomoteur.

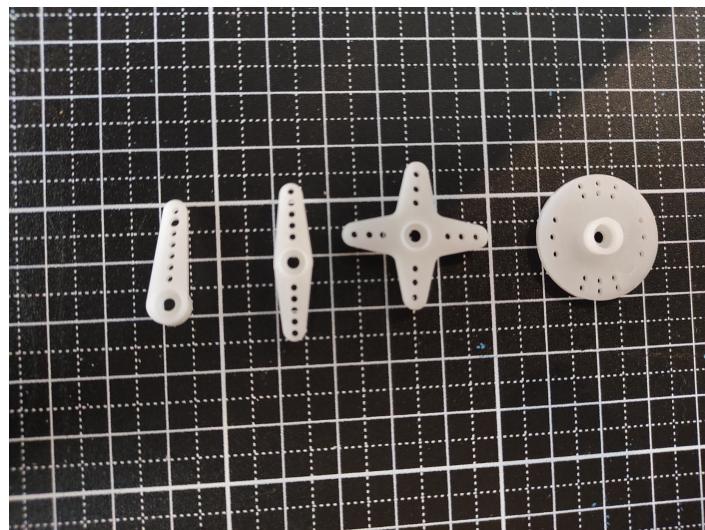
Malheureusement, les bras exacts inclus ne sont généralement pas spécifiés et peuvent varier.

Et, comme les arbres de sortie des servomoteurs et leurs cannelures varient, les bras sont souvent incompatibles entre les marques et les modèles de servomoteurs.

La façon la plus simple de fixer l'aiguille de votre horloge est d'utiliser un élastique, mais vous pouvez aussi utiliser de la colle chaude ou du scotch.

1

2



Créez l'aiguille de l'horloge et fixez-la au bras du servomoteur.

CRÉER UN MINUTEUR POUR LES ŒUFS



ETAPE 1 - CONSTRUIRE



Créer le panneau avant du minuteur

Sur le carton, faites un petit trou de la taille de l'arbre de votre servomoteur. Le trou doit se trouver au milieu du côté le plus long de votre carton.

Mettez le servomoteur derrière et fixez l'aiguille d'horloge sur l'arbre du servomoteur.

Tourner le bras en position minimale (angle 0°) et fixer le servomoteur de façon à ce que l'aiguille de l'horloge soit horizontale. Avec un stylo, faites une petite marque pour indiquer 0 seconde. Tournez le bras en position maximum (angle 180°) et faites une petite marque pour indiquer 180 secondes.

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

Programmer votre carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre servomoteur commencera à bouger ! Avant d'essayer ce programme sur la carte, vous pouvez l'essayer directement dans le simulateur. Si vous cliquez sur le bouton USER, vous verrez votre minuterie démarrer.

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé **RESET**). Si tout fonctionne bien, votre servomoteur commencera à bouger.

3



Créer le panneau avant du minuteur

4

6

7



ETAPE 2 - PROGRAMMER



```
input.buttonUser.onEvent(ButtonEvent.Click, function () {
    for (let pos = 0; pos <= 179; pos++) {
        pins.D3.servoWrite(pos)
        pause(1000)
    }
    for (let i = 0; i < 5; i++) {
        pins.D4.servoWrite(0)
        pause(1000)
        pins.D4.servoWrite(180)
        pause(1000)
    }
})
```

Comment cela fonctionne-t-il ?

La partie principale du code concerne les interactions avec les boutons. Ces interactions sont faites avec la fonction **input.buttonUSER.onEvent**.

Lorsque vous cliquez sur le bouton **USER**, vous démarrez la minuterie en modifiant la position du servomoteur d'un degré par seconde.

Lorsque vous avez fini de compter de 179 à 0, vous commencez à déplacer rapidement votre servomoteur pour signaler la fin de la minuterie.

COLLECTER DES DONNÉES

#R1AS15



Disponible sur

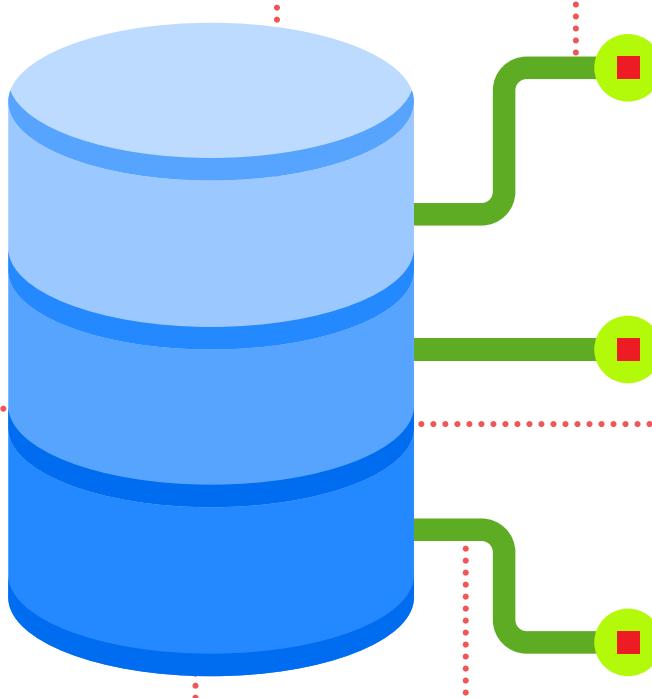


Prérequis

- R1AS04 - Découverte des capteurs de lumière

Matériel

- 1 carte programmable "STM32 IoT Node Board"
- 1 câble USB Micro-B



De quoi parle-t-on ?

Cette fiche d'activité explique comment collecter les données d'un capteur environnemental et les exporter vers un ordinateur afin d'effectuer une analyse simple à l'aide d'un tableur.

Durée

50 minutes

Niveau de difficulté

Avancé

OBJECTIFS D'APPRENTISSAGE

- Objectifs d'apprentissage
- Lire la valeur d'un capteur
- Stocker la valeur du capteur sur la mémoire flash de la carte
- Exporter toutes les valeurs collectées dans un fichier CSV (Comma Separated Values).
- Ajouter une extension à MakeCode



COLLECTER DES DONNÉES



Un capteur mesure une quantité physique et la convertit en un signal qui peut être transformé en une valeur numérique par un microcontrôleur. Dans votre programme, vous pouvez utiliser cette valeur pour adapter le comportement de votre algorithme (par exemple fermer la porte de la maison lorsque la valeur du capteur de lumière devient faible).

Lorsque vous souhaitez mener une expérience scientifique, une seule valeur ne vous donne pas suffisamment d'informations pour émettre des hypothèses. Vous devez observer comment la valeur de votre capteur va évoluer sur une longue période de temps.

Cette fiche d'activité explique comment collecter les données d'un capteur environnemental et comment les exporter vers un ordinateur pour effectuer une analyse simple à l'aide d'un tableau.



ÉTAPE 1 - CONSTRUIRE



Connecter la carte à l'ordinateur

1

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le **connecteur micro-USB ST-LINK** (sur le coin en haut à droite de la carte). Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur appelé **DIS_L4IOT**. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "New Project". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

2

Installer l'extension

Après avoir créé votre nouveau projet, vous accédez à l'éditeur MakeCode par défaut ci-contre et vous devrez installer une extension.

3



Editeur MakeCode par défaut



Les extensions dans MakeCode sont des groupes de blocs de code qui ne sont pas directement inclus dans les blocs de code de base que l'on trouve dans MakeCode. Les extensions, comme leur nom l'indique, ajoutent des blocs pour des fonctionnalités spécifiques. Il existe des extensions pour un large éventail de fonctionnalités très utiles, ajoutant des capacités de manette de jeu, de clavier, de souris, de servomoteurs, de la robotique et bien plus encore.

COLLECTER DES DONNÉES



ÉTAPE 1 - CONSTRUIRE



Vous voyez le bouton noir **AVANCÉ** en bas de la colonne des différents groupes de blocs. Si vous cliquez sur **AVANCÉ**, vous verrez apparaître des groupes de blocs supplémentaires. En bas, il y a une boîte grise appelée **EXTENSIONS**. Cliquez sur ce bouton. Dans la liste des extensions disponibles, vous pouvez facilement trouver l'extension du **Datalogger** qui sera utilisée pour cette activité. Si elle n'est pas directement disponible sur votre écran, vous pouvez la rechercher en utilisant l'outil de recherche. Cliquez sur l'extension que vous souhaitez utiliser et un nouveau groupe de blocs apparaîtra sur l'écran principal.



Fonctionnalités avancées

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**". Copiez le fichier binaire sur le lecteur **DIS_L4IOT** et attendez que la carte finisse de clignoter et votre datalogger est prêt !

4

Utiliser votre datalogger

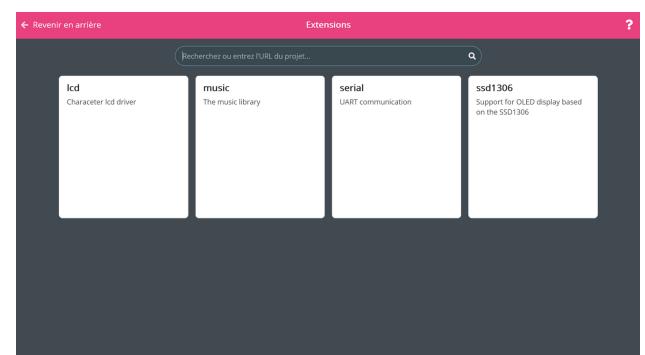
Le programme enregistre les données dans la mémoire flash (la LED 1 est allumée) jusqu'à ce que vous appuyiez sur le bouton USER, ce qui fait s'allumer la LED2. C'est l'indication que l'enregistrement des données est arrêté et que vous pouvez copier les données sur votre ordinateur.

5

Obtenir les données

Avec votre câble USB, connectez la carte à votre ordinateur en utilisant le connecteur **USB OTG** (celui de gauche lorsque vous regardez la carte de haut en bas). Lorsque votre projet est enregistré, une nouvelle clef USB devrait apparaître sous le nom de **MAKECODE**. Le répertoire **SPIFLASH** contient les données du programme. Les données de journalisation sont écrites dans un fichier nommé **log.csv**.

Ressource : [wikipedia.org/wiki/Serial_Peripheral_Interface](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface)



Liste d'extensions et outil de recherche

6

i Assurez-vous d'avoir arrêté l'enregistrement de vos données avant d'accéder au fichier **log.csv** avec un programme quelconque. Appuyer sur Reset ou débrancher la carte sans mettre en pause l'enregistrement des données avec le bouton USER corrompra le fichier **log.csv** ! Appuyez sur le bouton USER pour arrêter l'enregistrement, ce qui fermera correctement le fichier et permettra de copier les données.



Datalogger and associated blocks



ETAPE 1 - CONSTRUIRE



Copiez le fichier **log.csv** sur votre disque pour le sauvegarder et le consulter ultérieurement.

Visualiser les données

Ouvrez un **programme de feuille de calcul** tel que Google Sheets, Microsoft Excel, macOS Numbers, etc. Ouvrez le fichier **log.csv**. Le tableur devrait reconnaître le **CSV** (si votre programme ne le reconnaît pas, vous devrez peut-être préciser que vous essayez d'ouvrir un fichier **CSV** ou utiliser une fonction d'importation). Dans Google Sheets, le fichier s'ouvre correctement.

Ressource : https://en.wikipedia.org/wiki/Comma-separated_values

Les lignes sep= et NAN peuvent être ignorées si elles apparaissent. La ligne 2 contient les titres des données que vous lisez. D'abord l'heure, puis, pour l'exemple, les relevés de température, de lumière et d'humidité du sol dans chaque colonne. Les données peuvent aller très loin puisque l'exemple enregistre toutes les 10 secondes. Vous pouvez enregistrer plus lentement, par exemple toutes les 60 secondes (1 minute) ou les 300 secondes (5 minutes), etc. Les données peuvent être utilisées à des fins d'analyse ou pour établir un graphique des valeurs sur une période donnée. En utilisant la fonction graphique de Google Sheets, appuyez sur le bouton graphique de la barre d'outils et sans aucun formatage, vous obtenez un superbe graphique !

Exécuter, modifier, jouer

Votre programme s'exécutera automatiquement chaque fois que vous le sauvegarderez ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé RESET). Si tout fonctionne bien, votre carte mettra à jour les LED d'état pour montrer que la collecte de données est en cours. Essayez de comprendre l'exemple et commencez à le modifier en changeant la période entre deux mesures, en ajoutant d'autres données provenant d'autres capteurs de la carte. Essayez d'enregistrer autant de données que vous le souhaitez dans autant d'endroits pour comprendre comment la température, l'humidité et la pression évoluent.

7

```

on start
  data logger set separator comma ▾
  set data logger sampling interval to 100 ▾ (ms)
  data logger to console ON
  data logger ON
  set running to 1
  digital write pin LED to HIGH
  digital write pin LED2 to LOW

```

```

on button USER click
  set running to 0
  data logger OFF
  digital write pin LED to LOW
  digital write pin LED2 to HIGH

```

```

forever
  if running = 1 then
    set temperature to temperature in °C
    set pressure to pressure in hPa
    set humidity to relative humidity in percent
    data logger add "Temp" = temperature
    data logger add "Pressure" = pressure
    data logger add "Humidity" = humidity
    data logger add row
    pause 10000 ms

```

Blocs permettant l'exécution du programme



ETAPE 2 - PROGRAMMER



```
//Initier la collecte de données
let running = 0
datalogger.setSampleInterval(100)
datalogger.sendToConsole(true)
datalogger.setEnabled(true)
running = 1
pins.LED.digitalWrite(true)
pins.LED2.digitalWrite(false)

//Stopper la collecte de données après l'appui sur le bouton
input.buttonUser.onEvent(ButtonEvent.Click, function () {
    running = 0
    datalogger.setEnabled(false)
    pins.LED.digitalWrite(false)
    pins.LED2.digitalWrite(true)
})

//Collecter des données des capteurs toutes les 10 secondes
forever(function () {
    if (running == 1) {
        let temperature = input.temperature(TemperatureUnit.Celsius)
        let pressure = input.pressure(PressureUnit.HectoPascal)
        let humidity = input.humidity()

        datalogger.addValue("Temp", temperature)
        datalogger.addValue("Pressure", pressure)
        datalogger.addValue("Humidity", humidity)
        datalogger.addRow()
    }
    pause(10000)
})
```



ETAPE 2 - PROGRAMMER



Comment cela fonctionne-t-il ? Initialiser la collecte de données :

Pour télécharger le fichier sur un ordinateur, nous devons arrêter la collecte de données. La variable **running** permet de connaître l'état actuel du processus de collecte de données. Lorsque la valeur est 0, la collecte de données est arrêtée et lorsqu'elle est 1, la collecte de données est en cours.

Les 3 instructions suivantes configurent le datalogger avec les paramètres suivants :

- Une virgule est utilisée comme séparateur de champs dans le fichier CSV.
- L'intervalle minimal entre deux lignes est fixé à 100 ms.
- Toutes les données sont envoyées à la console MakeCode afin de les afficher directement dans l'éditeur.

Après la configuration, le processus de collecte de données est activé et le voyant d'état est utilisé pour montrer l'état actuel du processus.

Arrêtez la collecte de données après avoir cliqué sur le bouton USER

Pour arrêter le processus de collecte de données, nous utilisons le bouton USER. Lorsque l'on clique sur ce bouton, le datalogger est désactivé, les LED d'état sont mises à jour et la valeur 0 est attribuée à l'exécution.

Pour gérer l'asynchronisme du clic du bouton (un clic de bouton peut se produire à n'importe quelle étape de notre programme), nous utilisons le mécanisme Event de MakeCode. Ce mécanisme permet d'exécuter un ensemble spécifique d'instructions lorsqu'une condition donnée apparaît. Dans notre cas, l'événement est "le bouton USER est cliqué".

Lorsque le datalogger est désactivé, il n'y a plus d'écriture sur le fichier journal, nous n'avons donc aucun risque de le corrompre.

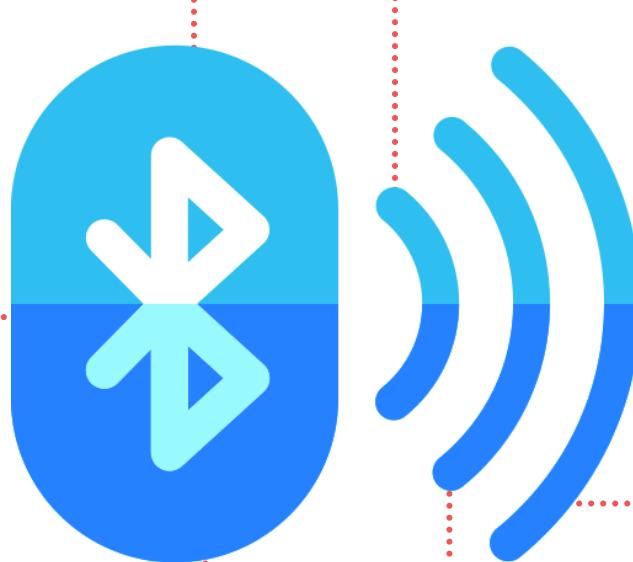
Collecter les données des capteurs toutes les 10s

Dans la boucle principale, il suffit de lire les données et de les envoyer au datalogger si la variable **running** est fixée à 1. La **pause** à la fin de la boucle permet de fixer la période entre deux mesures. Si nous voulons observer une expérience plus longue, nous augmenterons probablement cette valeur.

MAGNETICS

Créer des projets multicartes

#R1AS16

**Disponible sur****Durée**

50 minutes

Matériel

- 4 cartes programmables "STM32 IoT Node Board"
- 1 câble USB Micro-B
- 1 écran OLED Monochrome 1.3" 128x64 OLED de Adafruit
- 1 câble QT pour connecter l'écran à la carte

De quoi parle-t-on ?

Dans cette activité, nous allons programmer plusieurs cartes électroniques et échanger des données de capteurs à l'aide de l'extension Magnetics, qui permet une communication maillée sans fil.

Coopération**Niveau de difficulté**

Avancé

OBJECTIFS D'APPRENTISSAGE

- Utiliser l'extension magnetics pour constituer des projets multicartes
- Appréhender le fonctionnement des modules bluetooth BLE Mesh

MAGNETICS - CRÉER DES PROJETS MULTICARTES



Cette fiche d'activité propose de créer des projets plus complexes en utilisant plusieurs cartes électroniques non connectées entre elles. Une fois les capteurs maîtrisés, nous pouvons en effet mettre en place des expériences nécessitant l'utilisation de plusieurs cartes. Afin de réaliser la collecte des données, il faut pouvoir faire communiquer les cartes entre elles par les airs. Dans cette activité nous allons programmer plusieurs cartes électroniques et échanger des données de capteurs à l'aide de l'extension **Magnetics** que permet de mettre en œuvre une communication sans fil maillée. Le projet magnetics prend la forme d'une brique technique logicielle implantée directement dans MakeCode. Ce développement est basé sur l'utilisation de la technologie de réseau maillé **Bluetooth Low Energy Mesh** (BLE Mesh) compatible avec toutes les cartes programmables disposant d'un module **Bluetooth Low Energy**.

Ressources : <https://www.magnetics.edu-up.fr/>

<https://blog.rtone.fr/bluetooth-mesh>

https://fr.wikipedia.org/wiki/Bluetooth_%C3%A0_basse_consommation



ÉTAPE 1 - CONSTRUIRE



Pour réaliser cette activité nous avons besoin de **quatre cartes STM32 IoT Nodes**. Trois d'entre elles seront **émettrices** de données de capteurs (température, humidité, pression), et la dernière sera **collectrice** des données qu'elle affichera sur un écran OLED. Mis à part l'écran de la dernière, il n'y a pas de câblage car nous utiliserons uniquement les capteurs internes. Nous allons donc vous donner la marche à suivre pour câbler et programmer en premier lieu la carte collectrice puis dans un second temps, programmer individuellement chaque carte émettrice afin de pouvoir construire votre projet.

Activité 1 - Préparer, câbler et programmer la carte collectrice

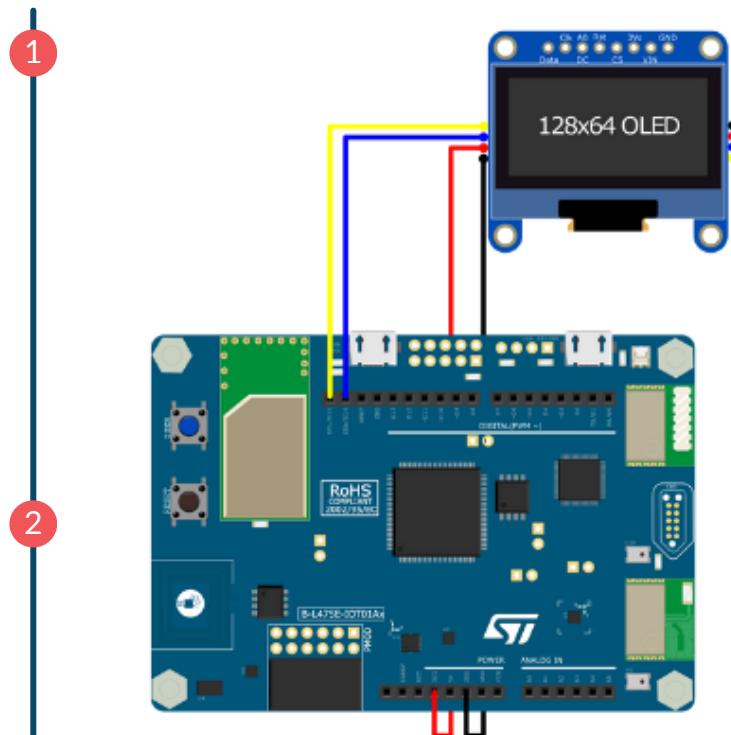
Câbler l'écran OLED

Nous devons en premier lieu câbler l'écran OLED directement à la carte collectrice. Il y a deux façons de câbler l'écran **OLED SSD1306** à une carte, soit avec une connexion **I2C** ou **SPI**. Pour notre écran, nous utilisons la connexion **I2C** via le câble **QWIIC/STEMMA** avec la convention suivante :

- **Noir** pour **GND**
- **Rouge** pour **V+ (3V3)**
- **Bleu** pour **SDA (D14)**
- **Jaune** pour **SCL (D15)**

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte collectrice à votre ordinateur en utilisant le connecteur micro-USB. Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur.



Câblage de l'écran OLED sur la carte collectrice

MAGNETICS - CRÉER DES PROJETS MULTICARTES



ÉTAPE 1 - CONSTRUIRE



Ouvrir MakeCode

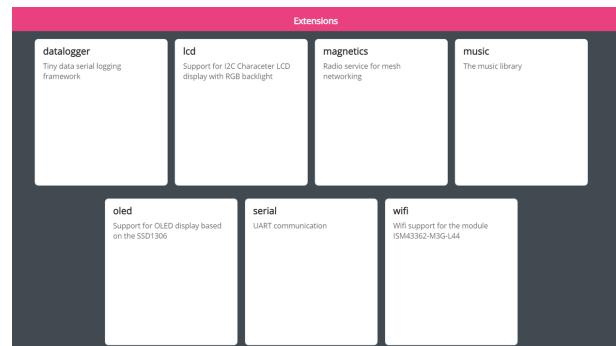
Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "Nouveau projet". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

Installer les extensions

Après avoir créé votre nouveau projet, vous obtiendrez l'écran par défaut et vous devrez installer deux extensions pour ce projet spécifique : "magnetics" et "oled".

3



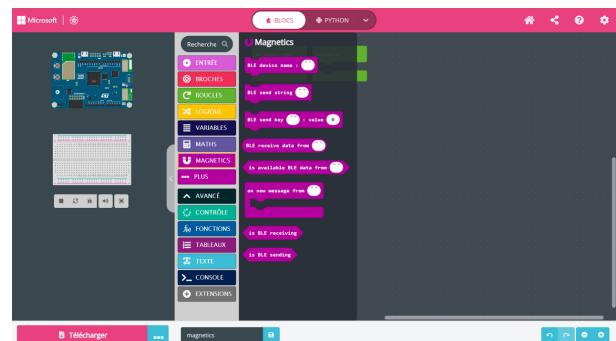
Liste d'extensions et outil de recherche

i Les extensions dans MakeCode sont des groupes de blocs de code qui ne sont pas directement inclus dans les blocs de code de base que l'on trouve dans MakeCode. Les extensions, comme leur nom l'indique, ajoutent des blocs pour des fonctionnalités spécifiques. Il existe des extensions pour un large éventail de fonctionnalités très utiles, ajoutant des capacités de manette de jeu, de clavier, de souris, de servomoteurs, de la robotique et bien plus encore.

Vous voyez le bouton noir **AVANCÉ** en bas de la colonne des différents groupes de blocs. En cliquant sur ce bouton, vous verrez apparaître des groupes de blocs supplémentaires. En bas, il y a une boîte grise appelée **EXTENSIONS**. Cliquez sur ce bouton.

Dans la liste des extensions disponibles, vous pouvez trouver l'extension **Magnetics** qui sera utilisée pour cette activité. Si elle n'est pas directement disponible sur votre écran, vous pouvez la rechercher en utilisant la barre de recherche. Cliquez sur l'extension et un nouveau groupe de blocs apparaîtra sur l'écran principal. Répétez cette action pour installer l'extension **OLEO** également disponible. Vous pouvez commencer à programmer.

4

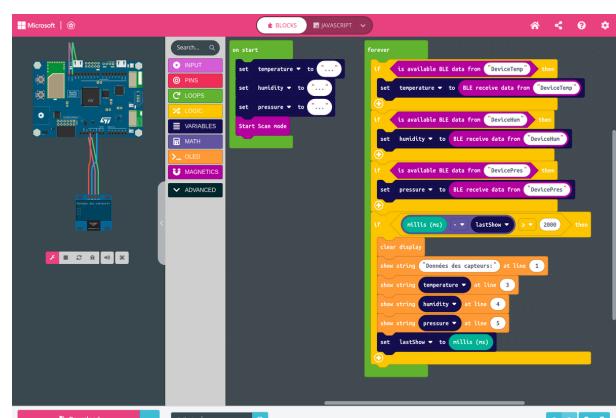


Blocs associés à l'extension magnetics

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Cliquez sur le bouton "**Télécharger**" et attendez que la carte finisse de clignoter.

5



Capture d'écran de MakeCode avec les blocs du programme du collecteur



ETAPE 1 - CONSTRUIRE



Activité 2 - Programmer chaque carte émettrice individuellement

Une fois la carte collectrice câblée avec l'écran OLED et programmée, nous pouvons préparer les trois cartes émettrices en suivant les mêmes étapes de programmation que lors de l'étape 1. Pour chaque carte, il faudra donc effectuer les tâches suivantes :

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte émettrice que vous souhaitez programmer à votre ordinateur en utilisant le **connecteur micro-USB**. Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

6

Ouvrir MakeCode

Allez dans [l'éditeur MakeCode de Let's STEAM](#). Sur la page d'accueil, créez un nouveau projet en cliquant sur le bouton "**Nouveau projet**". Donnez à votre projet un nom plus expressif que "Sans titre" et lancez votre éditeur.

Ressource : makecode.lets-steam.eu

7

Installer l'extension

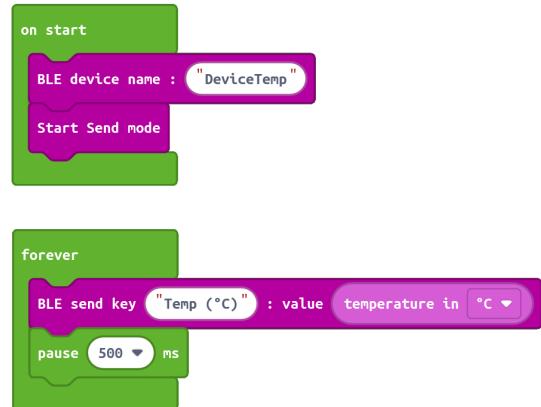
De la même manière qu'à l'étape 1, ajoutez l'extension magnetics à la liste de bloc via le menu "**AVANCÉ**" puis "**EXTENSIONS**".

8

Programmer la carte

Dans l'éditeur JavaScript de MakeCode, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous relatif à la carte émettrice que vous êtes en train de programmer. Si ce n'est pas déjà fait, pensez à donner un nom à votre projet et cliquez sur le bouton "**Télécharger**" et attendez que la carte finisse de clignoter.

9



Exemple de la programmation par bloc relative à la carte émettrice collectant les données de température

Activité 3 - Exécuter, modifier, jouer

Une fois votre carte collectrice reliée à votre écran, et vos quatre cartes programmées, votre programme est prêt à être utilisé. Votre programme s'exécutera automatiquement chaque fois que vous les mettrez sous tension ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé RESET). Essayez de comprendre le fonctionnement de votre code et commencez à le modifier en créant vos propres projets.

MAGNETICS - CRÉER DES PROJETS MULTICARTES



ETAPE 2 - PROGRAMMER



Code de la carte collectrice

```

let temperature = "...";
let humidity = "...";
let pressure = "...";
let lastShow = 0;

magnetics.startScanning()
forever(function () {

    if (magnetics.availableDataFromName("DeviceTemp")) {
        temperature = magnetics.readDataFromName("DeviceTemp")
    }
    if (magnetics.availableDataFromName("DeviceHum")) {
        humidity = magnetics.readDataFromName("DeviceHum")
    }
    if (magnetics.availableDataFromName("DevicePres")) {
        pressure = magnetics.readDataFromName("DevicePres")
    }

    if( control.millis() - lastShow >= 2000 ){
        oled.clear()
        oled.showString("Données des capteurs:", 1)
        oled.showString(temperature, 3)
        oled.showString(humidity, 4)
        oled.showString(pressure, 5)

        lastShow = control.millis();
    }
})

```

Code de la carte émettrice de la température

```

magnetics.setLocalName("DeviceTemp")
magnetics.startEmitting()
forever(function () {
    magnetics.setAdvertisingKeyValueData("Temp ( °C)", input.temperature(TemperatureUnit.Celsius))
    pause(500)
})

```



ETAPE 2 - PROGRAMMER



Code de la carte émettrice de l'humidité

```
magneticss.setLocalName( "DeviceHum" )
magneticss.startEmitting()
forever(function () {
  magneticss.setAdvertisingKeyValueData( "Hum (%)", input.humidity())
  pause(500)
})
```

Code de la carte émettrice de la pression

```
magneticss.setLocalName( "DevicePres" )
magneticss.startEmitting()
forever(function () {
  magneticss.setAdvertisingKeyValueData( "Pres (hPa)", input.pressure(PressureUnit.HectoPascal))
  pause(500)
})
```

Comment cela fonctionne-t-il ? Initialiser la collecte de données :

Cela fait beaucoup de code, mais rien de très complexe, puisque le code des **cartes émettrices** est quasiment identique (il n'y a que le capteur qui change) et reste simple à comprendre.

Pour commencer, on donne un nom à notre carte à l'aide de la fonction `setLocalName`, ce qui permettra au collecteur de reconnaître la donnée envoyée. Pour que la carte puisse émettre des données, il faut lui préciser son rôle, c'est ce que fait la fonction `startEmitting`. Enfin, via la fonction `setAdvertisingKeyValueData`, nous envoyons la donnée du capteur toutes les 500 millisecondes (c'est-à-dire deux fois par seconde).

Enfin, intéressons-nous au code du collecteur. Au début, nous déclarons 4 variables:

- `temperature`, `humidity` et `pressure` qui contiendront les données émises par les émetteurs (respectivement pour la température, l'humidité et la pression atmosphérique)
- `lastShow` qui permet de savoir à quand remonte le dernier affichage des données (en millisecondes).

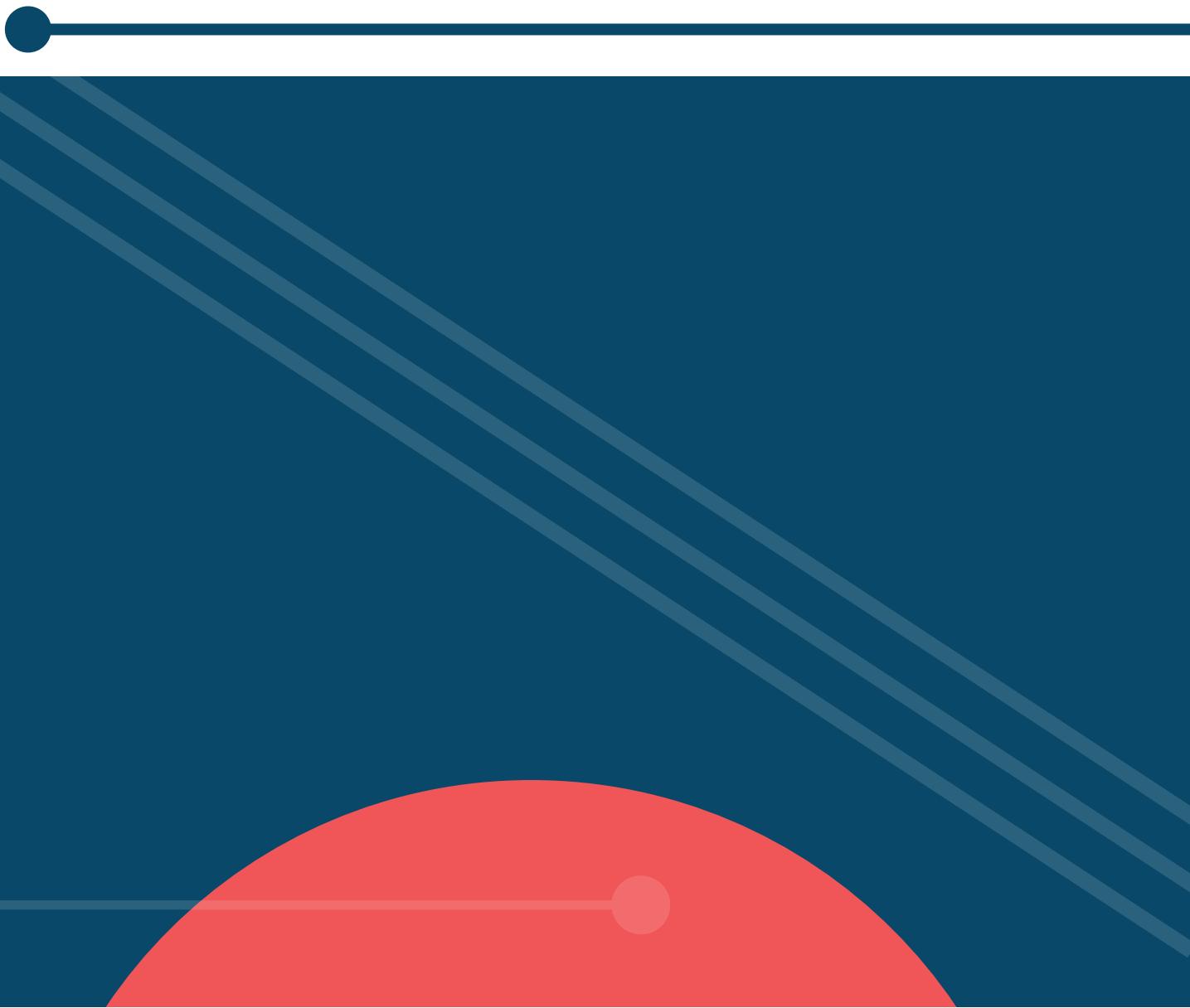
Une fois les variables initialisées, nous faisons appel à `startScanning`, pour rechercher les cartes émettrices qui sont à proximité. Le reste du code se trouve dans la fonction `forever` ce qui implique, que nous allons répéter indéfiniment le code suivant. Les trois premiers `if(...)` permettent de savoir si nous avons reçu des données de la part des émetteurs (grâce à la fonction `availableDataFromName`), si tel est le cas, alors on enregistre cette donnée dans la variable associée (en utilisant `readDataFromName`).

Maintenant que nous avons les données, il faut les afficher sur notre écran OLED, c'est le rôle de la dernière condition `if`, qui nous permet de mettre à jour notre écran avec les nouvelles données, seulement si cela fait plus de deux seconds qu'il n'a pas été rafraîchi.

partie 4

Idées de projets STEAM

inspirations et ressources



Les inspirations et ressources

Les tableaux ci-dessous vous donnent accès à différentes idées de projet que vous pourriez mettre en œuvre grâce à nos différentes ressources ! N'hésitez pas à vous emparer de chaque proposition et à utiliser magnetics pour les transformer en expérimentation porteuse de sens et motivante !

Idées de projet Let's STEAM

Chaque projet Let's STEAM a été décrit plus en détails au sein d'un wiki dédié. Vous pouvez accéder à chaque projet en visitant la page suivante : <https://bit.ly/3Oc62eF>

NOM DU PROJET	OBJECTIFS
Comment rendre visible l'invisible ?	Reproduire l'environnement naturel des grenouilles afin d'assurer leur survie
Préserver la biodiversité	Surveillez le nombre d'espèces végétales dans votre quartier. Explorez les rues et les parcs de votre quartier pour en savoir plus sur l'écosystème et utilisez la technologie pour faciliter ce processus !
Contrôle de la température dans la classe	Il fait trop chaud dans la salle de classe. Lorsque les élèves entrent, ils savent qu'ils doivent fermer les stores, mais pendant la pause, la classe devient vraiment chaude. Comment créer un système plus autonome grâce à la programmation ?
Construire une salle de classe accueillante	Identifiez les besoins en intensité lumineuse particulière dans votre classe pour réaliser une activité spécifique.
Votre maison idéale (et durable)	Rêvez à l'endroit où vous aimeriez vivre, à ce que serait votre maison idéale et à la façon dont cette maison idéale pourrait être plus durable.
Gestes barrières	Bien que de nouvelles routines aient été mises en place pour s'assurer que tous les enfants se lavent les mains, comment la programmation peut nous aider à respecter les gestes barrières ?
Consommation raisonnable de chauffage	Identifiez la position optimale d'utilisation des appareils de chauffage, à des moments donnés, pour économiser l'électricité.

Les protocoles d'expérimentation TheDexterLab

Chaque protocole TheDexterLab fait l'objet d'une fiche d'activité spécifique accessible au sein d'un wiki dédié. Vous pouvez accéder à chaque protocole en visitant la page suivante : <https://bit.ly/3tAkrb9>

PROTOCOLE	DÉFI ABORDÉ
Pourquoi l'océan est-il salé ?	Comment séparer le sel et l'eau et contrôler le résultat ?
Le son se déplace-t-il de plus de 100 mètres en 1 seconde ?	A quelle vitesse le son voyage-t-il ?
Quand faut-il arroser une plante ?	Comment irriguer automatiquement une plante ?
Votre temps de réaction est-il inférieur à une demi-seconde ?	Comment évaluer le temps de réaction d'un cycliste ?
Est-ce que notre corps ou ses parties (par exemple les mains, les jambes, etc.) sont accélérés de plus de 1 g même si nous dansons comme des fous ?	Quelle force notre corps ou ses parties subissent-ils lorsque nous bougeons, courons ou dansons ?
La distraction peut-elle modifier votre temps de réaction ?	Comment évaluer le temps de réaction d'un cycliste ?
Le vent et les chutes d'eau ont-ils de l'énergie ?	Comment pouvons-nous récolter de l'énergie pour produire de l'électricité à partir de sources renouvelables ?
La lumière a-t-elle de l'énergie ?	Comment pouvons-nous récolter de l'énergie pour produire de l'électricité à partir de sources renouvelables ?
La qualité de l'air diminue-t-elle lorsque le nombre de personnes présentes dans une pièce augmente ?	Comment améliorer la qualité de l'air en milieu fermé ?
Peut-on mesurer l'atténuation du son par un matériau ?	Quels sont les matériaux habituels qui peuvent atténuer le son ?
Peut-on construire une machine qui utilise le vent pour soulever un poids de 50 g ?	Peut-on utiliser le vent comme source d'énergie mécanique ?
Comment maximiser l'apport en énergie solaire et créer des panneaux solaires auto-orientables ?	Comment construire des systèmes solaires efficaces ?

Les protocoles TheDexterLab - suite

PROTOCOLE	DÉFI ABORDÉ
Comment pouvons-nous réduire la quantité d'énergie que nous utilisons ?	Comment faire des économies d'énergie ?
Une plante consomme-t-elle plus de CO2 qu'elle n'en rejette ?	Comment améliorer la qualité de l'air ?
L'IA peut-elle être un outil de sécurité sans faille ?	Comment détecter les visages ?
La température modifie-t-elle la vitesse du son ?	Est-il possible de mesurer la variation de la vitesse du son en fonction de la température ambiante ?
Commander les équipements en fonction de l'énergie produite par le panneau solaire	Identifier différentes ressources en énergie et connaître quelques conversions d'énergie



Contactez-nous et découvrez l'ensemble de nos ressources



Posez-nous des questions

Contactez-nous : contact@labaixbidouille.com

Partagez vos commentaires et corrections

Ce guide a été réalisé avec la meilleure qualité possible et une réelle volonté de participer à l'émergence de contenus stimulants dans le domaine de la programmation. Cependant, nous ne sommes que des humains ! Si vous découvrez des erreurs ou des corrections à apporter, n'hésitez pas à nous contacter ! Nous ferons en sorte que vous soyez crédité·e pour votre aide !

Coopérer avec nous au sein de nouveaux projets

Le L.A.B lance régulièrement de nouveaux projet et est ouvert à de nouvelles coopérations, que ce soit avec des écoles mais aussi avec des les acteurs du monde de la programmation. N'hésitez pas à nous contacter pour construire ces initiatives ensemble !

Explorer nos ressources

- magnetics : <https://www.magnetics.edu-up.fr/>
- Let's STEAM : <https://lets-steam.eu/>
- TheDexterLab :
<http://www.thedexterlab.eu/resources>

Découvrir le dispositif edu-up

<https://eduscol.education.fr/1603/le-dispositif-edu-up>

Découvrir les actions et financement Erasmus +

<https://monprojet.erasmusplus.fr/>