UNE HORLOGE DES PHASES DE LA LUNE

Hervé Faivre, enseignant de sciences physiques, Semur-en-Auxois

Comme le montre cet article, il est tout à fait possible de concilier, ici dans le cadre d'un club, astronomie (avec les phases de la Lune), programmation (avec un Arduino), bricolage (branchements, soudure...) et utilisation d'une imprimante 3D.

Le projet qui nous a occupé une bonne partie de l'année passée au club astronomie du collège était de construire une horloge qui indique les positions relatives du Soleil, de la Terre et de la Lune avec une aiguille imprimée en 3D et motorisée avec un rotor pas à pas, la phase de la Lune étant affichée avec des LED et un écran LCD.

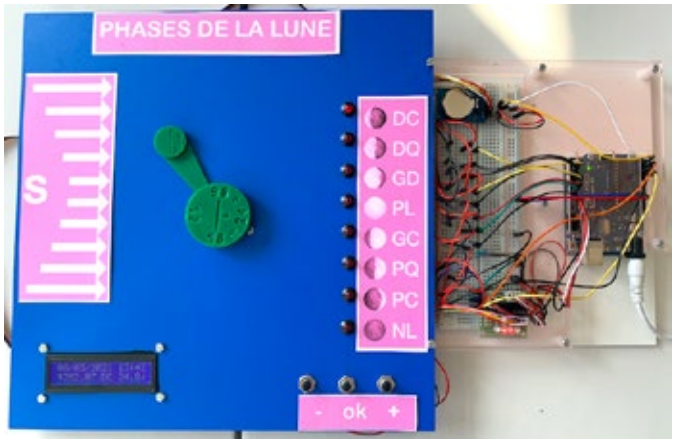

Fig.1. **L'ensemble de la maquette.**

Vue d'ensemble de la maquette

En voici un petit résumé avec quelques détails techniques.

L'écran LCD 16x02 à 16 caractères sur deux lignes (figure 2) indique la date (08/03/2021), l'heure (13:44), le jour julien décimal (9282,07) « raccourci » (en retirant

Fig.2. **L'écran LCD.**

2 450 000)[1], la phase de lune (DC) et l'âge de la Lune (24,8 j).

Sur le **disque central** qui représente la Terre, le nombre 12 pointe vers le Soleil (figure 3). C'est le midi. Le petit

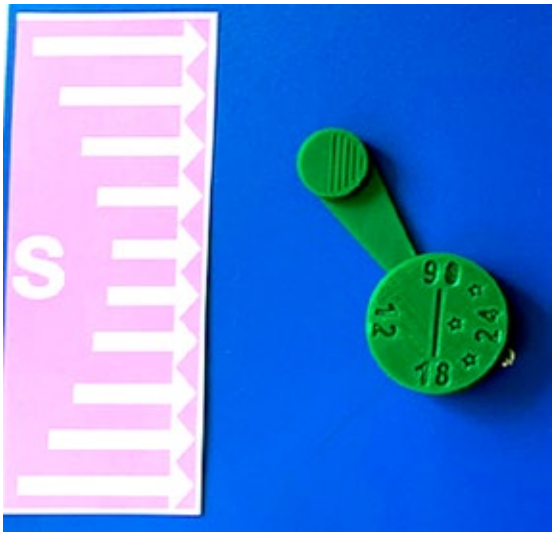

Fig.3. **À gauche, le Soleil S, au centre, la Terre avec une graduation en heure solaire. Au-dessus, la Lune avec sa partie éclairée à gauche et sa partie dans l'ombre à droite, hachurée.**

1 Le but est de limiter le nombre de chiffres significatifs dans les calculs et à l'affichage.

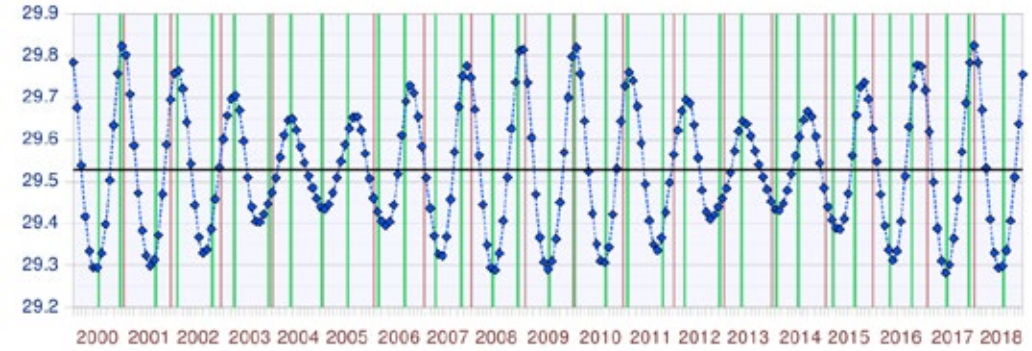

Fig.4. **La durée de la lunaison peut varier de +/- 6 h comme le montre ce graphique (crédit W!B/Wikipédia).**

disque demi-hachuré représente la Lune et sa nuit. Il pivote pour orienter la partie éclairée de la Lune dans la direction du Soleil.

Sur cette image, on voit que pour la date indiquée la Lune est en dernier croissant. L'horloge montre aussi que la Lune est bien visible au sud dans le ciel vers 7 h le matin. Ainsi, avec cet exemple, on peut se représenter ce que voit un observateur terrestre au sud vers 7 h du matin : un croissant éclairé par la gauche *i.e.* un dernier croissant.

Dans cette version simplifiée de l'horloge[2], la position de la Lune est calculée à partir de la fraction de l'âge de la Lune sur la période synodique moyenne de 29,53 jours [3]. L'horloge n'est donc jamais vraiment juste mais jamais très fausse (+/- 1 j environ pour la phase).

Fig.5. **Les 8 phases.**

Cette période moyenne de 29,53 j est découpée en 8 phases principales : NL, PC, PQ, GC, PL, GD, DQ, DC. **Une LED rouge** est allumée à côté de l'abréviation DC (dernier croissant) et d'un dessin de la phase correspondante (figure 5).

Chaque phase durant 29,53 j / 8 soit environ 3,7 j. Ainsi la LED pour la pleine Lune doit s'allumer environ 2 j avant et rester allumée presque 4 j.

Trois boutons permettent d'interagir avec la machine. Deux d'entre eux (+ et -) permettent de modifier la date par pas de 1 jour afin de chercher la prochaine pleine lune par exemple. Le bouton Ok permet de revenir à l'heure et à la date actuelles ou de réinitialiser l'horloge avec un appui long (il n'est pas nécessaire de connaître la phase de la Lune pour démarrer, on doit juste placer l'aiguille en position de nouvelle lune avec les boutons + et - puis valider avec Ok pour enregistrer la position de référence pour le moteur lors du démarrage de l'horloge.

Le matériel électronique pour la maquette coûte au total moins de 25 €. Il est composé d'un Arduino Uno[4], d'une horloge RTC I2C DS1307, d'un écran LCD I2C 16x02, d'un moteur pas à pas 28YBJ-48 et son *driver*, de 8 LED et résistors 470 Ω, de 3 boutons poussoirs, des fils, une alimentation et une platine d'expérimentation ou « breadboard » (plaque pour connexions rapides).

Les connexions

Les leds sont connectées aux broches 2 à 9 de sorties numériques et protégées par les résistances avec la masse (GND) en commun.
Les entrées analogiques A1, A2 et A3 sont utilisées pour les boutons reliés au GND. On définira ces entrées en INPUT_PULLUP. Ainsi lorsqu'on presse un bouton, on force un état bas, l'Arduino réagira donc à un front descendant de potentiel sur les entrées analogiques A1, A2 et A3.
Le moteur via son contrôleur utilise les broches 10, 11, 12 et 13.
L'horloge temps réel RTC et l'écran LCD utilisent le même bus I2C : SDA (Serial Data) sur A4 et SCL (Serial Clock) sur A5 et sont alimentés entre le 5V et le GND.

Pour la **décoration du plan de l'horloge** nous avions le projet de réaliser les indications en gravant des plaques de PVC bicolores avec le robot d'usinage de la salle de technologie, mais le confinement nous a fait prendre du retard sur cette partie du projet alors nous avons étudié une suggestion de décoration presque comestible et biodégradable venue d'outre-Atlantique.

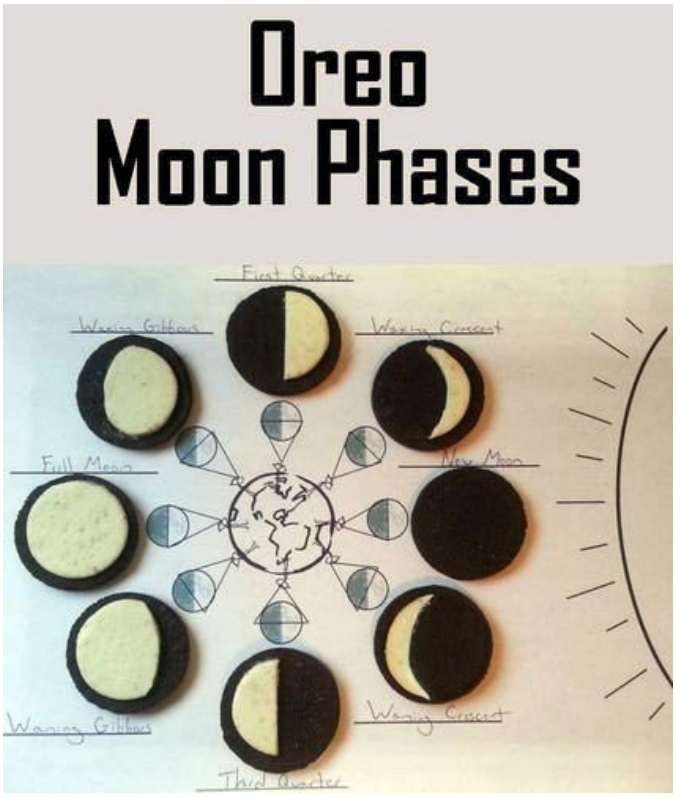

Fig.6. **Ici les phases de la Lune sont représentées en gâteaux « OREO ».**

2 Une version plus complète et plus précise de l'horloge calcule la phase à partir des longitudes écliptiques géocentriques du Soleil et de la Lune à l'aide des équations données dans les *Calculs astronomiques à l'usage des amateurs* de Jean Meeus.
3 29,53 j pour une lunaison est une moyenne. Entre 2005 et 2012 elle a oscillé entre 29,82 et 29,28 (figure 4).
4 Il s'agit d'un microcontrôleur qui peut être programmé. Il dispose d'entrées numériques et analogiques et de sorties numériques.

Finalement, nous nous sommes contenté temporairement d'une impression couleur [5] sur bristol.

Les pièces à imprimer en 3D

Les modèles ont été créés avec *www.tinkercad.com* en ligne. Pour un premier prototype, une simple flèche en carton indiquait la phase. C'était déjà satisfaisant mais nous avons réquisitionné l'imprimante 3D des salles de technologie.

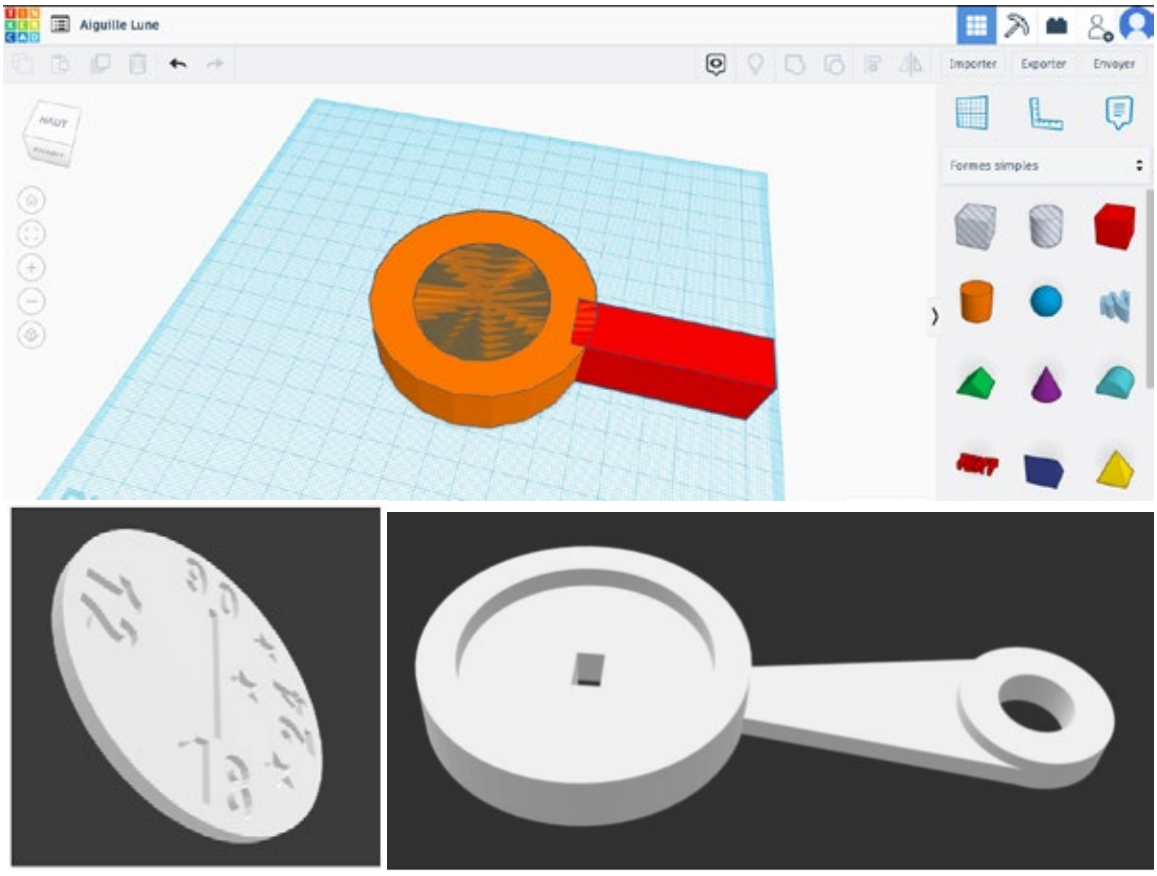

Fig.7. Les pièces à imprimer en 3D.

Principe de fonctionnement et programmation en C++

La phase de la Lune est obtenue en comptant le nombre de jours depuis une date de référence (notée REFLUNE dans le programme) : la nouvelle lune du 13 janvier 2021 à 05 h 01 min UT (julien 2 459 227,709 03).
On commence par calculer le jour julien (variable de type float notée JD) en fonction de la date donnée par l'horloge RTC en temps universel. Le calcul est donné dans le code en fin d'article. L'algorithme est issu du livre de Jean Meeus *Calculs astronomiques à l'usage des amateurs*. On peut aussi négliger le décalage heure d'été / heure d'hiver). On calcule ensuite le nombre (décimal) **k** de lunaisons moyennes (29,53 j) écoulé depuis la date de référence.

5 Hélas, l'imprimante était en panne de couleur. Le bleu nuit est sorti rose. On s'y fait.

k = (JD - REFLUNE) / 29.53
On ramène ce nombre dans l'intervalle [0,1[(avec des boucles de type while par exemple) .
Le reste de la division euclidienne par 8 de l'arrondi à l'entier de 8´k correspond à un numéro de phase entre 0 et 7.
numero_phase = round(k * 8) % 8

Ainsi 0 est la nouvelle lune, 1 est le premier croissant, 2 est le premier quartier… 7 est le dernier croissant.
On allume la LED qui correspond à ce numéro de phase. Pour chaque LED dans une boucle **for (i= 0; i<8; i++)** on regarde si la LED numéro i doit être éteinte ou allumée. Cette procédure évite d'écrire les 8 cas à la suite.
On affiche sur le LCD la date, l'heure, le jour julien, la phase selon le numéro et le nombre k × 29,53 qui correspond à l'âge de la Lune. Pour la phase on pourrait utiliser une suite de conditions comme :
if (numero_phase = = 0) nom_phase= « NL » ;
if (numero_phase = = 1) nom_phase= « PC » ; …
ce qui est facile à comprendre pour les collégiens, ou même créer un (switch / case) sur la variable numero_phase mais, après quelques explications, il s'est avéré plus rapide d'aller chercher dans une liste déclarée en constante :
nom_phase[numero_phase]
Il reste à gérer **la rotation du moteur** pour placer correctement la Lune. On est aidé par le driver du stepper (moteur pas à pas) en lui indiquant simplement le nombre

de pas à réaliser dans un sens ou l'autre. Ici un tour complet de 360° correspond à 32 × 64 = 2 048 pas.
Pour se positionner à la bonne phase, on doit indiquer au moteur un nombre de pas égal à 2 048 ´ k / 29,53 par rapport à la position de référence nouvelle lune. Cela ne paraît pas trop compliqué a priori mais il existe une contrainte de taille : le moteur n'a pas de « mémoire » de sa position de référence. On devra donc gérer cela en créant une variable qui sera la mémoire de la position actuelle, la mettre à jour à chaque mouvement et on devra donc passer par une étape d'initialisation pour donner la référence au démarrage de l'horloge.
Quelques petites améliorations optionnelles :

- afin d'éviter au moteur (assez lent) de réaliser plus d'un demi-tour dans un mouvement on peut calculer l'angle entre la position initiale et finale et si cette valeur dépasse 180° on indique alors un ordre de rotation dans l'autre sens égal à (360° − l'angle) ;
- afin de faire un petit effet, on peut précéder chaque changement par un petit mouvement de recul qui sera rattrapé ensuite dans l'ordre de rotation.

L'interaction avec les boutons

Les boutons + et - permettent de changer la date par pas de 1 j. La date est alors déconnectée de l'horloge temps réel RTC (suspension du mode automatique) et l'heure est fixée à midi.
Le bouton Ok permet de revenir au mode automatique piloté par l'horloge RTC. Un appui long sur Ok démarre la séquence d'initialisation qui se fait normalement au démarrage de l'horloge.
Pour initialiser l'horloge on utilise les boutons (+) et/ou (-) pour placer la Lune en position de nouvelle lune puis on valide avec **Ok**.

On utilise ici les PIN A1, A2 et A3 qui sont des entrées analogiques mais dans ce montage elles sont utilisées comme des entrées numériques avec un digitalRead (ButtonPin) qui retourne un état LOW lorsqu'on presse le bouton. En effet, en appuyant sur le bouton, on force la mise à la masse GND de cette entrée qui est maintenue en temps normal à l'état HIGH dans sa déclaration comme INPUT_PULLUP.

```
philune §
1
2 //librairies
3 #include <LiquidCrystal_I2C.h>        // Ecran LCD sur le bus I2C
4 #include <DS3231.h>                   // Horloge RTC
5 #include <math.h>
6 #include <Stepper.h>                  // Driver du moteur pas à pas
```

Fig.8. **Pour le code C++ dans l'IDE ARDUINO, des bibliothèques sont nécessaires (écran LCD, horloge RTC, math, driver du moteur pas à pas).**

Quelques pistes d'améliorations possibles, pour un prochain projet

Avec un Arduino MEGA on obtient plus de ports d'entrée/sortie, de puissance et de mémoire. Il est possible d'améliorer le dispositif en tous points :

- utilisation d'une bibliothèque ephemeris.h (trop lourde pour la version UNO de l'Arduino) basée sur les calculs de Jean Meeus, la VSOP87 et ELP2000[6] ; on peut réaliser des calculs plus rapides et précis des positions Terre, Lune et Soleil ;
- affichage TFT de la phase de la Lune avec l'inclinaison de la Lune par rapport à l'horizon;
- utilisation d'un GPS pour la position et le temps universel ;
- affichage des phases sur une matrice de LED ;

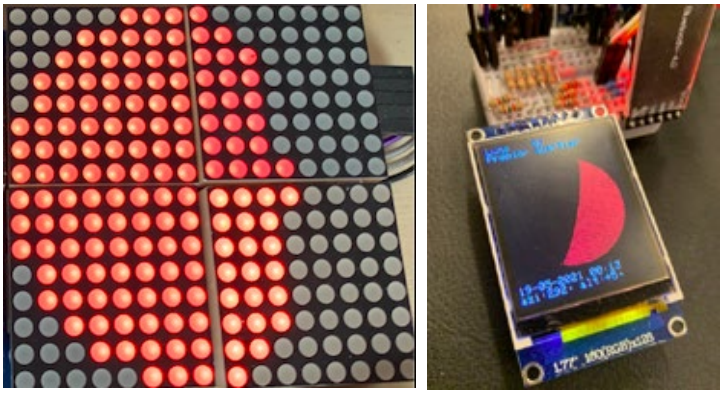

Fig.9. **Affichage de la phase sur une matrice de leds 16 × 16 à gauche, et sur un écran TFT 128 × 160 avec l'inclinaison par rapport à l'horizon à droite.**

- Les broches D0 et D1 étant réservées pour la communication avec le port série pendant la phase de test, il reste une broche analogique A0 libre. Elle pourrait être utilisée dans une amélioration pour y connecter une sonde de Hall. Elle détecterait le passage d'un petit aimant attaché à l'aiguille pour créer une position de référence du moteur pas à pas pour simplifier la procédure d'initialisation ;
- éphémérides plus précises des planètes, de la Lune et du Soleil (lever/coucher, coordonnées ALT/AZI ou RA/Dec, avec distance) ;
- capteurs météo ;

etc.
On peut trouver toutes sortes de composants pour des coûts très modiques. En fait, la limite est l'imagination et les élèves du club astro n'en manquent pas.

L'investissement des élèves

Bien impliqués dans la partie programmation, dessin 3D et astro, Ils ont particulièrement bien aimé la partie

6 Variations séculaires des orbites planétaires : VSOP est un modèle numérique utilisé pour les calculs des positions des planètes du Système solaire, de Mercure à Neptune. La théorie ELP2000 permet de calculer avec une précision satisfaisante la position de la Lune.

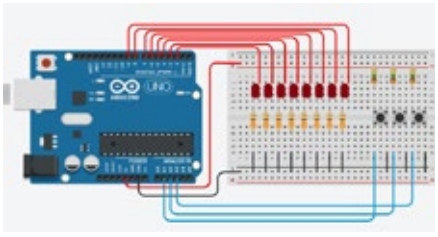

Fig.10. **Simulation d'Arduino avec Tinkercad.**

réalisation quand il s'est agi de percer, souder, coller, brancher les fils…

Pour la conception des objets 3D et pour le schéma de montage et simulation d'Arduino, nous avons été bien aidés avec le logiciel de conception en ligne tinkercad.com. La conception d'objet 3D est vraiment à la portée des élèves du collège. L'export des fichiers en .stl, directement utilisables par le logiciel CURA de l'imprimante 3D, est très rapide (voir la figure 7).

La même application permet également de simuler des montages avec Arduino (figure 10), de programmer avec des blocs (type Scratch). La conversion en code C++ est automatique.

C'est très pratique et intuitif pour les élèves les plus jeunes. C'est notamment les élèves de 6e et 5e qui se sont le plus intéressés au montage et à la découverte du code. Ils avaient principalement comme objectif de faire réagir l'Arduino aux signaux d'entrée et faire allumer des leds en sortie (figure 11).

Pour les plus aguerris, c'est la gestion du moteur qui a été le plus gros du travail. En partant d'exemples simples de code C++ et non plus en blocs, ils ont cherché les possibilités de modifier les valeurs, assembler des morceaux pour faire correspondre un angle de rotation et une phase de lune.

Au cours du projet, de nombreuses notions d'astronomie de mathématiques et de code ont été abordées et adaptées selon les niveaux (calendriers, lunaison et phases de la Lune, angles, proportionnalité, division euclidienne, conditions, boucles). L'assemblage final du code est bien sûr trop complexe pour les collégiens. Lors de la lecture expliquée, l'objectif principal était surtout comprendre l'intérêt de créer des fonctions que l'on peut appeler à chaque fois que nécessaire et qui retourne soit une valeur (voir l'encadré plus loin pour l'exemple de fonction qui retourne le jour julien à partir d'une date JJ,MM,AAAA,HH,MN) ou une action (rotation du moteur en fonction d'un angle donné en argument).

Plutôt que tinkercad.com, pour coder les élèves ont préféré l'application proposée par vittascience.com plus conviviale, elle permet simuler le fonctionnement et de téléverser directement dans l'Arduino pour tester le code.

L'apprentissage du code appliqué avec un microcontrôleur rend la programmation très concrète, notamment pour les conditions et les entrées/sorties : par exemple, *if* (si) le bouton envoie un signal bas sur l'entrée n, alors allume une led en envoyant un état haut sur la sortie p *else* (sinon) il faut l'éteindre. De même pour les boucles *for* (faire jusqu'à) ou *while* (faire tant que) qui sont souvent utilisées aussi.

Ça s'allume, ça bouge, ça fait du bruit. Les élèves se sont montrés très créatifs dans cette phase de découverte.

Nous ne tarderons pas à trouver un prochain projet.

■

Un extrait du code

```
//###### CALCUL DU JOUR JULIEN ########
// fonction de conversion d'une date en jour julien décimal à partir de DD/MM/YYYY HH:MN
// jour julien raccourci (-2450000) pour éviter les problèmes de chiffres significatifs des float (ou double) sur Arduino
float Julien (int DD, int MM, int YY, int HH, int MN) {
 float JJ = floor(365.25 * (((MM < 3) ? YY - 1 : YY) + 4716))
 + floor(30.6001 * (((MM < 3) ? MM + 12 : MM) + 1))
 + DD + 2 - floor(((MM < 3) ? YY - 1 : YY) / 100.0)
 + floor((floor(((MM < 3) ? YY - 1 : YY) / 100.0)) / 4.0) - 1524.5 - 2450000;
 JJ = JJ + (HH + MN / 60.0) / 24.0;
 return JJ;
```

Le code complet peut être téléchargé sur le site du Clea (www.clea-astro.eu) en cliquant sur Cahiers Clairaut n° 176 Hiver 2021.

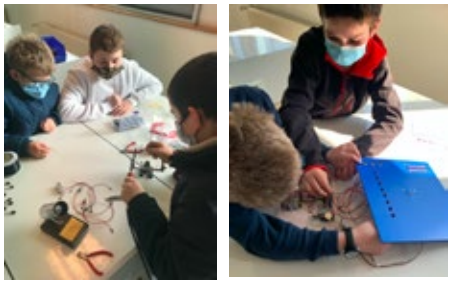

Fig.12. **Les élèves en action.**

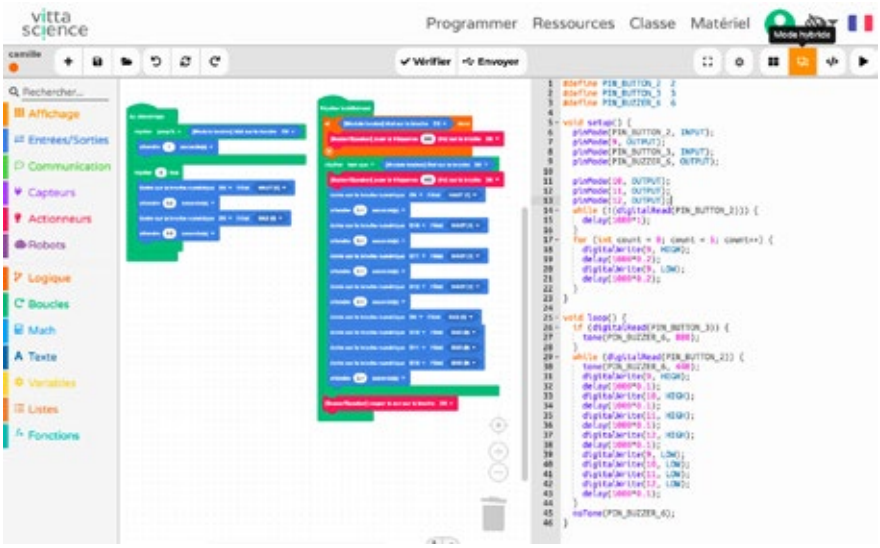

Fig.11. **Initiation au codage en bloc et en lignes avec Vittascience.com.**