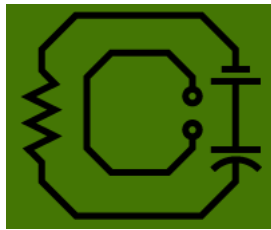


UNIVERSITÉ DE SHERBROOKE  
Groupe technique

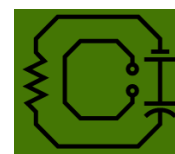


Compétition de  
Conception de  
Circuits Imprimés

# Guide de conception embarqué

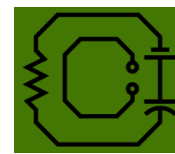
Document qui vise à supporter la programmation avec des microcontrôleurs.  
Guide concentré sur la famille **ATMEL**.

Rédigé par : Miriam Caisse, avec l'aide de Ramtin B. Meidani  
En date du : 23 octobre 2025



## Table des matières

1.	Introduction .....	3
2.	Note d'utilisation des programmeurs USBASP .....	3
3.	Guide d'installation.....	4
3.1.	VS Code et son extension PlatformIO .....	5
3.1.1.	Installez VS Code (si ce n'est pas déjà fait) : .....	5
3.1.2.	Téléchargez le plugin PlatformIO.....	5
3.1.3.	Télécharger la plateforme Atmel AVR .....	6
3.2.	Installation de Zadig .....	7
3.2.1.	Téléchargement de Zadig .....	7
3.2.2.	Téléchargement du bon Driver.....	7
3.3.	Installation de AVRdudess .....	8
3.3.1.	Télécharger AVRdudess.....	8
3.3.2.	L'interface utilisateur.....	8
4.	Guide de programmation .....	9
4.1.	Préparer le code compilé en HEX .....	9
4.1.1.	Ouvrir un exemple de code (ou votre code, éventuellement).....	9
4.1.2.	Comprendre la compilation .....	9
4.1.3.	Compiler le code.....	10
4.1.4.	Trouver le code compilé .....	11
4.2.	Préparer la puce à programmer .....	12
4.2.1.	Branchement à une puce.....	12
4.2.2.	Validation du branchement .....	13
4.3.	Programmer!.....	14
5.	Ressources et questions.....	15



## 1. Introduction

Le but de ce guide est d'être une introduction à la conception électronique embarquée comprenant un microcontrôleur (MCU). Il existe de nombreuses familles de MCU (ESP, PIC, STM, etc.), mais ce guide se concentre sur la famille ATMEL, en particulier sur l'atmega328. L'atmega328 n'est pas particulièrement puissant pour son prix, mais étant associé aux Arduino Uno, est un MCU idéal pour apprendre à faire de la conception et de la programmation embarquée.

Veuillez noter que bien que ce guide utilise des exemples concrets associés à l'atmega328, les grandes lignes de conception sont applicables à n'importe quel MCU : avec un peu de pratique et de recherche, vous devriez être capable d'intégrer n'importe quel MCU dans vos projets!

Ce guide est accompagné d'un séminaire sur la conception électronique, qui enseigne comment intégrer un MCU dans un schéma électronique, quelles bonnes pratiques de bases doivent être prises en compte, et quelles informations d'une datasheet sont indispensables.

Ce guide écrit est une référence sur comment effectuer la programmation du MCU de famille ATMEL (en particulier, atmega328), avec les programmeurs fournis par C3I, soit USBASP\_V3.

## 2. Note d'utilisation des programmeurs USBASP

Les programmeurs sont libre d'usage pour les participants de la compétition C3I de l'Université de Sherbrooke. Deux exemplaires du programmeur sont disponibles : un maximum de 1 exemplaire peut être emprunté et amené hors-campus à la fois, pour qu'il y ait toujours au moins un programmeur au Studio. L'emprunt doit avoir une durée maximale de 3 jours (jour d'emprunt et de retour inclus), et les responsables du groupe technique doivent être avisés par les équipes qui souhaitent emprunter un programmeur. Si vous avez besoin de toute information supplémentaire à ce sujet, merci de vous adresser à Miriam.

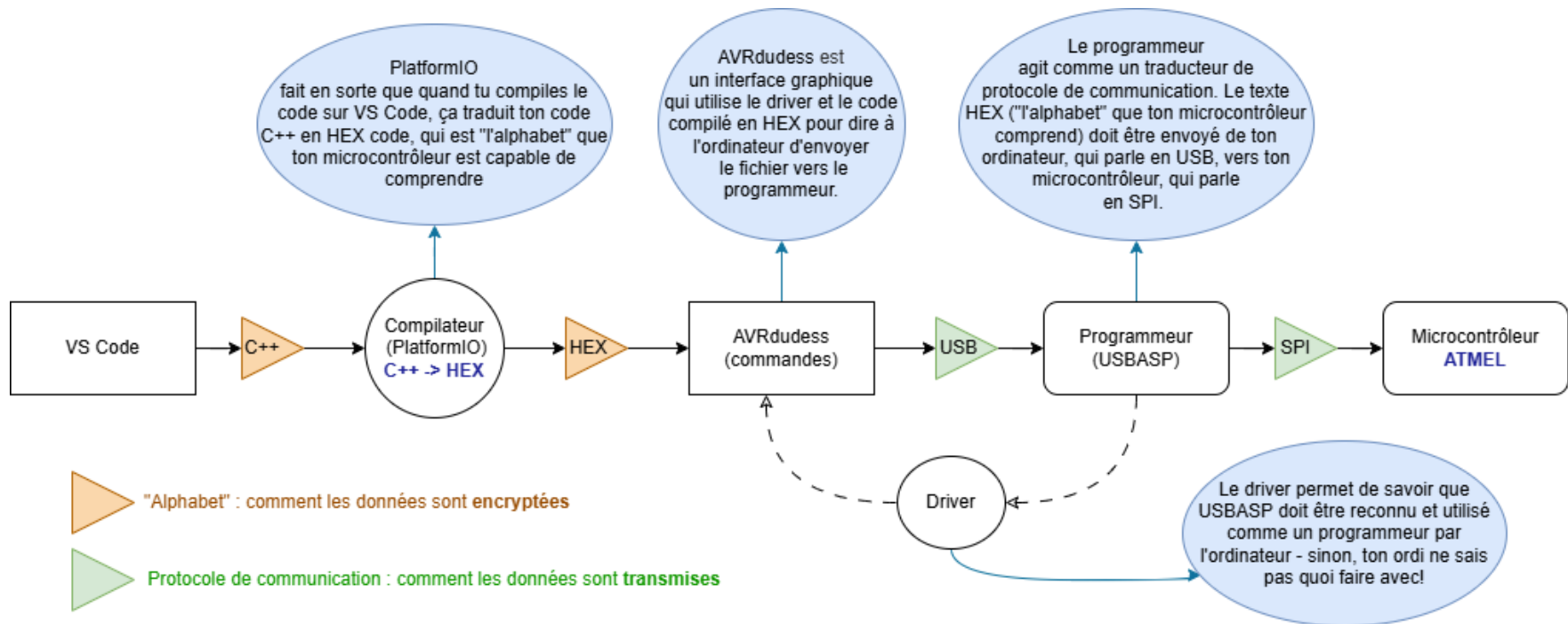
Les membres du groupe peuvent aussi les utiliser, mais uniquement sur les tables de C3I dans le studio. Les membres n'ont pas de possibilité d'emprunter des programmeurs, puisqu'une priorité est donnée aux participants.

**En cas de bris, il est primordial d'aviser les responsables du groupe!**

### 3. Guide d'installation

Le processus d'installation prend plus de temps. Il y a trois logiciels à installer, mais une fois le setup complété, la programmation en soi est assez simple.

Voici un diagramme décrivant le fonctionnement simplifié des données (et pourquoi chaque logiciel est nécessaire) :



Toi, tu écris du code en C++ sur VS Code. PlatformIO traduit le code en HEX, qui est la façon que le microcontrôleur est capable de traiter les données. AVRdudess simplifie la transmission des données vers le programmeur, et a besoin du driver du programmeur (téléchargé via Zadig) pour être capable de transmettre l'info. Ce dernier est capable de recevoir les données HEX transmises par ton ordinateur via le port USB, et les transmet avec le protocole de communication SPI au microcontrôleur, puisqu'un MCU Atmel n'est pas capable de communiquer par USB.

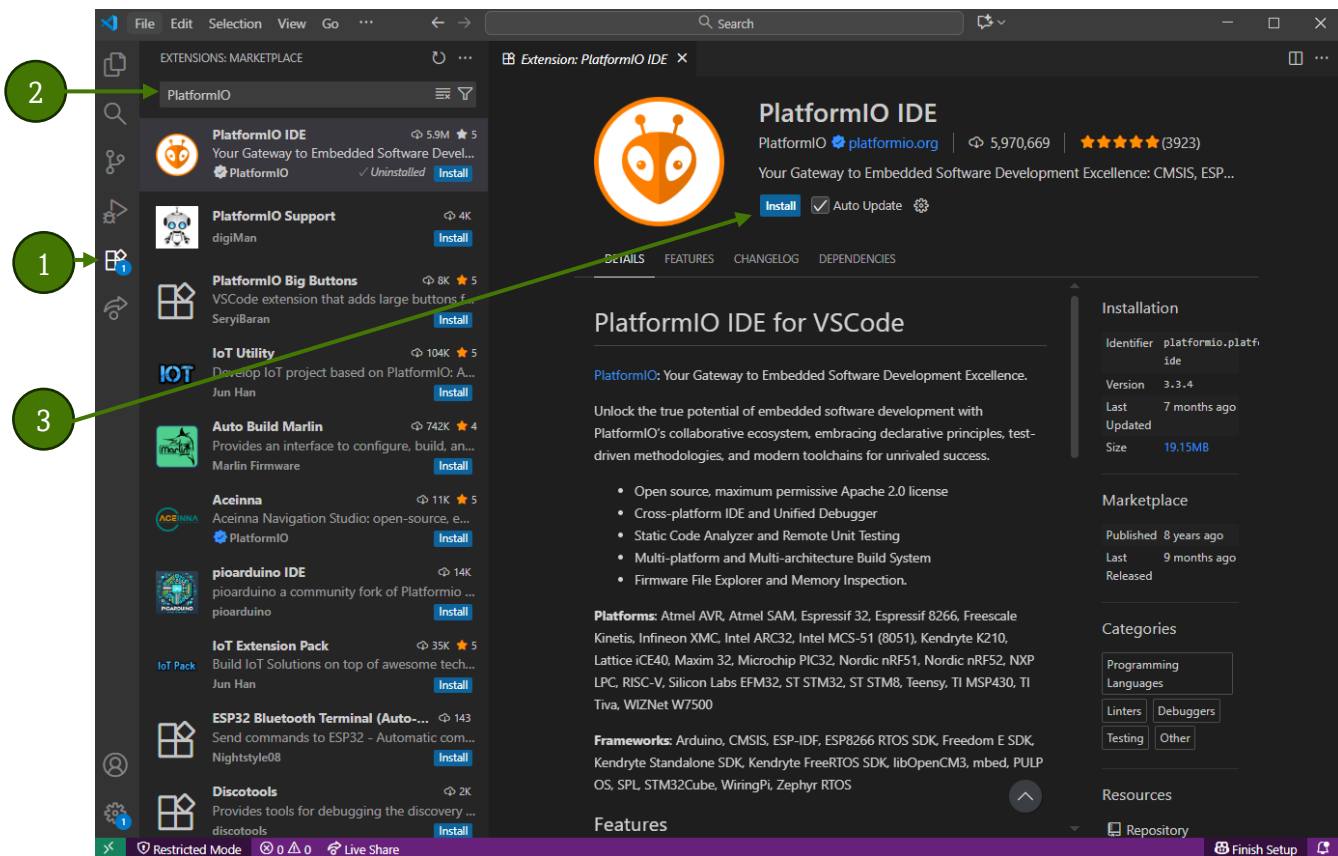
## 3.1. VS Code et son extension PlatformIO

VS Code est l'environnement de programmation que l'on vous suggère d'utiliser, puisqu'il est léger et versatile. Le plugin PlatformIO permet au compilateur de générer les bons fichiers HEX en fonction de la puce que l'on souhaite programmer

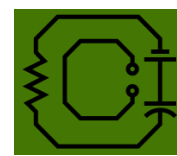
### 3.1.1. Installez VS Code (si ce n'est pas déjà fait) :

<https://code.visualstudio.com/download>

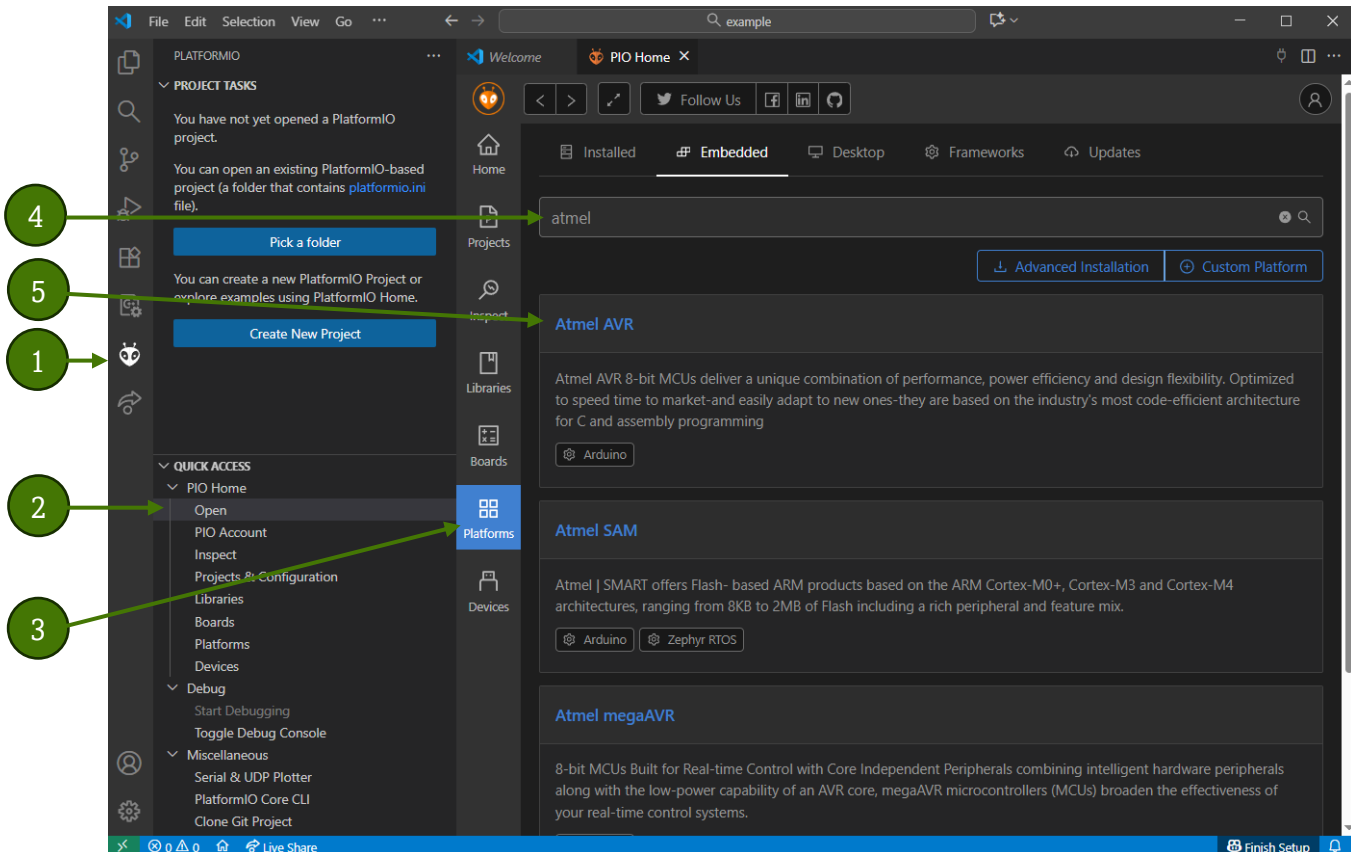
### 3.1.2. Téléchargez le plugin PlatformIO



1. Aller dans les extensions
2. Écrire « PlatformIO » dans la barre de recherche
3. Installer l'extension



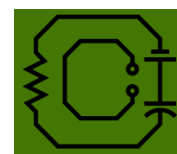
### 3.1.3. Télécharger la plateforme Atmel AVR



1. Ouvrir PlatformIO (si le logo n'apparaît pas sur la barre d'outil de gauche, ouvrez un « trusted workspace » (ça peut être un dossier random de projet))
2. Allez dans [Quick Access] -> [Open] pour ouvrir la page d'outil
3. Allez dans [Platforms]
4. Recherchez « Atmel » dans la barre de recherche
5. Choisissez [Atmel AVR] et téléchargez

Cette installation vous permettra d'obtenir des bibliothèques de projets exemple que nous allons utiliser dans ce guide, et d'avoir les fichiers JSON nécessaires pour générer le firmware (HEX).

Note : si vous n'utilisez pas des puces Atmel (EX. ESP, PIC32, RP2040), PlatformIO a aussi des plateformes pour ces familles.



## 3.2. Installation de Zadig

Zadig est un outil qui permet de télécharger le bon driver afin que ton ordinateur reconnaisse le programmeur comme étant un programmeur. Sinon, ton ordinateur ne sait pas comment le traiter, et tu ne peux pas l'utiliser.

### 3.2.1. Téléchargement de Zadig

<https://zadig.akeo.ie/>

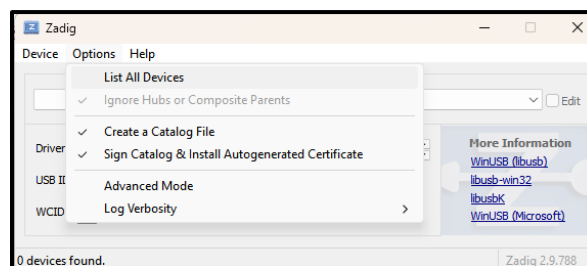


### 3.2.2. Téléchargement du bon Driver

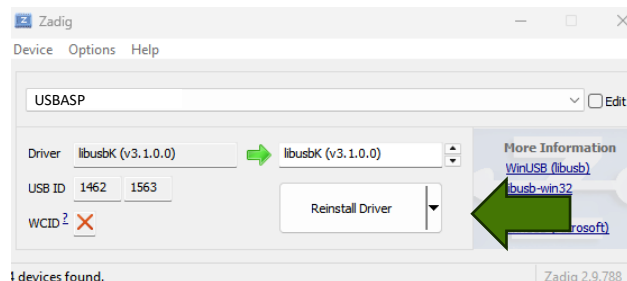
1. Ouvrir le logiciel Zadig
2. Brancher le programmeur USBASP au port USB-A de votre ordinateur

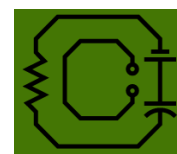
Note : utilisez toujours le même port USB lorsque vous branchez le programmeur. Si vous changez de port, il se peut que votre ordinateur « oublie » c'est quoi, et que vous aillez à refaire cette étape.

3. Si « USBASP » apparaît dans la barre, passez à l'étape 4. Sinon, allez dans [Options] -> [List All Devices], puis choisissez USBASP dans la liste.



4. Installer le driver « libusbK (v3.1.0.0) » sur USBASP





### 3.3. Installation de AVRdudess

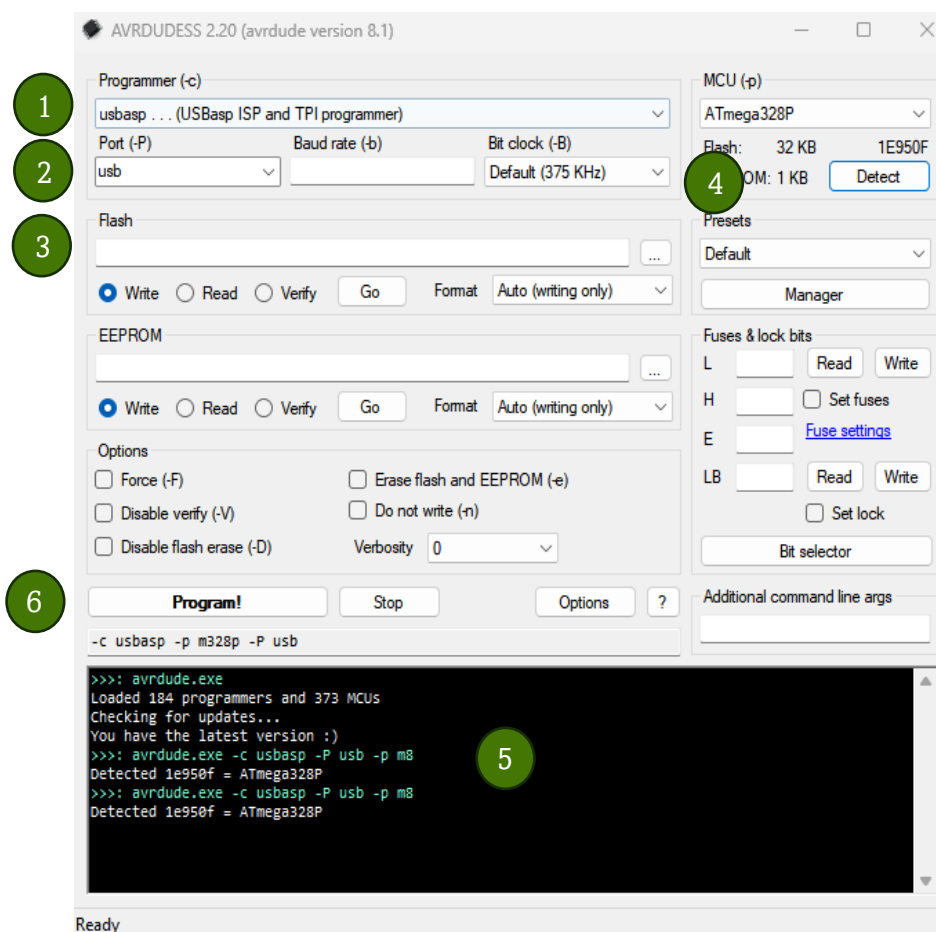
AVRdudess est un interface utilisateur qui permet de programmer des puces de types AVR (tel que Atmel) à partir de leur code HEX.

#### 3.3.1. Télécharger AVRdudess

<https://github.com/ZakKemble/AVRDUDESS/releases>

Download	What it's for
<a href="#">AVRDUDESS-2.20-setup.exe</a> (recommended)	Installer for Windows Vista/7/8/10/11
<a href="#">AVRDUDESS-2.20-portable.zip</a>	Portable for any OS

#### 3.3.2. L'interface utilisateur



1. Le programmeur est toujours [usbasp] (celui fourni par C3I)

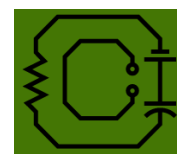
2. Le port est toujours [usb]

3. Vous devez mettre le path vers votre fichier HEX contenant le code que vous souhaitez envoyer dans le Flash

4. Si le programmeur est branché au microcontrôleur, « Detect » devrait automatiquement trouver quelle puce est branchée et l'indiquer dans (5)

6. Si tout le reste est prêt, on peut programmer!



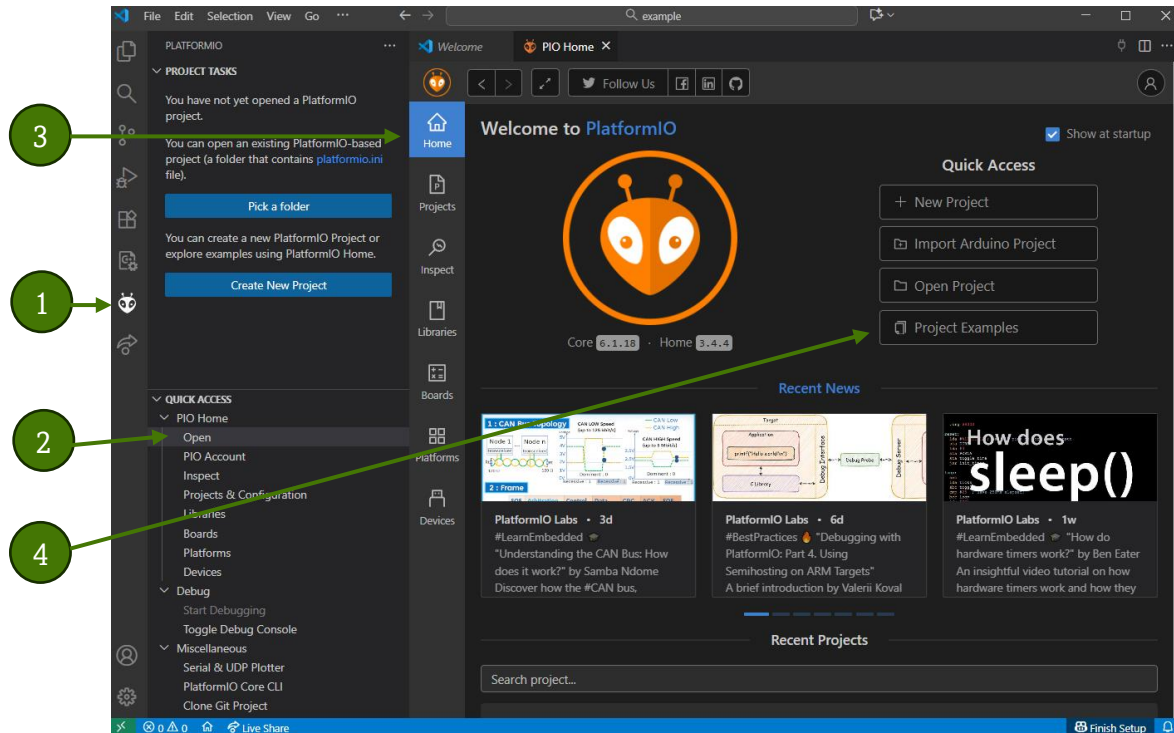


## 4. Guide de programmation

### 4.1. Préparer le code compilé en HEX

#### 4.1.1. Ouvrir un exemple de code (ou votre code, éventuellement)

Utilisez PlatformIO pour ouvrir l'exemple « Blink », qui fait clignoter la DEL « builtin » d'un Arduino (broche 19 = PB5 d'un atmega328/P)

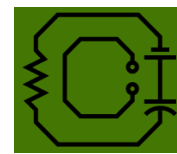


1. Aller dans PlatformIO
2. Aller dans la page d'accueil
3. Aller dans « Home »
4. Ouvrir un projet exemple
5. Sélectionner [arduino-blink] de la liste

#### 4.1.2. Comprendre la compilation

Si vous ouvrez le fichier [platform.ini], vous allez voir une liste sous la forme :

```
[env : xxx]
platform = xxx
framework = xxx
board = xxx
```



Il se peut qu'il y ait des lignes supplémentaires, par exemple si un board n'a pas la même broche « LED-builtin » qu'un Arduino UNO.

Si on compile dans cet état, la compilation prendra énormément de temps, puisque VS Code va devoir créer le fichier HEX pour chacune des puces de la liste. Dans notre cas, nous allons programmer un atmega328. Cette puce a la même configuration interne que l'Atmega328P, qui est sur l'Arduino UNO. Nous pouvons donc compiler uniquement le premier de la liste :

```
[env:uno]
platform = atmelavr
framework = arduino
board = uno
```

### ATTENTION :

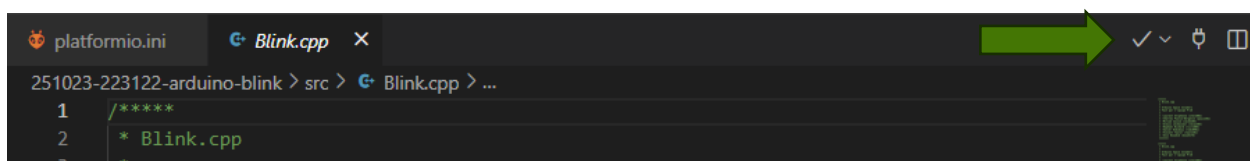
Si vous participez dans la compétition C3I, veuillez noter que le framework « Arduino » (ça veut dire, toutes les bibliothèques de bases d'un Arduino), sont considérés des béquilles de programmation. Puisque nous souhaitons vous pousser à en apprendre autant que possible, l'utilisation des bibliothèques de bases Arduino sont pénalisées de 7 points dans la partie « Complexité Software ».

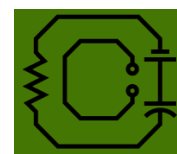
Toute librairie simple (protocoles I2C, UART, SPI, ainsi que bibliothèques de simplification de circuits intégrés) est équivalente à un module, donc pénalisée de 2 points. Des bibliothèques complexes (considérant la limitation de temps de la compétition et le but de concentrer les apprentissages sur les PCBs et la conception embarquée plus que logiciel), tel que des cartes SD, des environnements Linux, des protocoles USB (sur certains microcontrôleurs) ou tout équivalent en complexité, ne seront pas pénalisés.

### 4.1.3. Compiler le code

La première compilation prend toujours plus de temps, puisque tout le code HEX est généré. Les prochaines compilations sont plus rapides, puisque seulement les parties modifiées seront régénérées en HEX.

Assurez-vous bien de **compiler (Build, ✓)** et non pas de **transmettre (Upload →)**.





#### 4.1.4. Trouver le code compilé

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS +
No dependencies
Building in release mode
Checking size .pio\build\uno\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [ ] 0.4% (used 9 bytes from 2048 bytes)
Flash: [ ] 2.9% (used 924 bytes from 32256 bytes)
===== [SUCCESS] Took 0.92 seconds =====
* Terminal will be reused by tasks, press any key to close it.
```

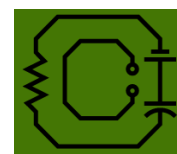
Une fois la compilation terminée (success), vous pouvez trouver le code HEX en allant dans :

[Projet actuel] -> [.pio] -> [uno (ou puce que vous programmez)] -> firmware.HEX

The screenshot shows the VS Code interface. On the left, the Explorer pane displays the project structure. The file `firmware.hex` is highlighted under the path `.pio\build\uno`. On the right, the `firmware.hex` file is open, showing a hex dump of the compiled code. The hex dump consists of 26 lines, each starting with a line number followed by a colon and the hex value.

```
251023-223122-arduino-blink > .pio > build > uno > firmware.hex
1 :10000000C945C000C946E000C946E000C946E00CA
2 :10001000C946E000C946E000C946E000C946E00A8
3 :10002000C946E000C946E000C946E000C946E0098
4 :10003000C946E000C946E000C946E000C946E0088
5 :10004000C9413010C946E000C946E000C946E00D2
6 :10005000C946E000C946E000C946E000C946E0068
7 :10006000C946E000C946E0000000002400270029
8 :100070002A0000000000250028002B00040404CE
9 :1000800004040404040202020202030303030342
10 :10009000010204081020408001020408102001021F
11 :1000A00004081020000000080002010000030407FB
12 :1000B000000000000000000000011241FBECFEFD8E0B8
13 :1000C000DEBFCDBF21E0A0E0B1E001C01D92A930AC
14 :1000D000B207E1F70E945D010C94CC010C94000082
15 :1000E000E1EBF0E02491EDE9F0E09491E9E8F0E053
16 :1000F000E491EE23C9F0222339F0233001F1A8F472
17 :10010000213019F1223029F1F0E0EE0FFF1FEE58F7
18 :10011000FF4FA591B4912FB7F894EC91811126C0AF
19 :1001200090959E239C932FBF08952730A9F02830E7
20 :10013000C9F0243049F7209180002F7D03C0209121
21 :1001400080002F7720938000DFCF24B52F7724BD48
22 :10015000DBCF24B52F7DFBCF2091B0002F772093EC
23 :10016000B000D2CF2091B0002F7DF9CF9E2BDACFF7
24 :100170003FB7F8948091050190910601A091070185
25 :10018000B091080126B5A89B05C02F3F19F0019634
26 :10019000A11DB11D3FBFBA2FA92F982F8827BC01E1
```

Le fichier HEX devrait ressembler à celui de l'image. C'est la façon que les données sont encryptées pour que le microcontrôleur soit capable de comprendre le code que tu as écrit.



## 4.2. Préparer la puce à programmer

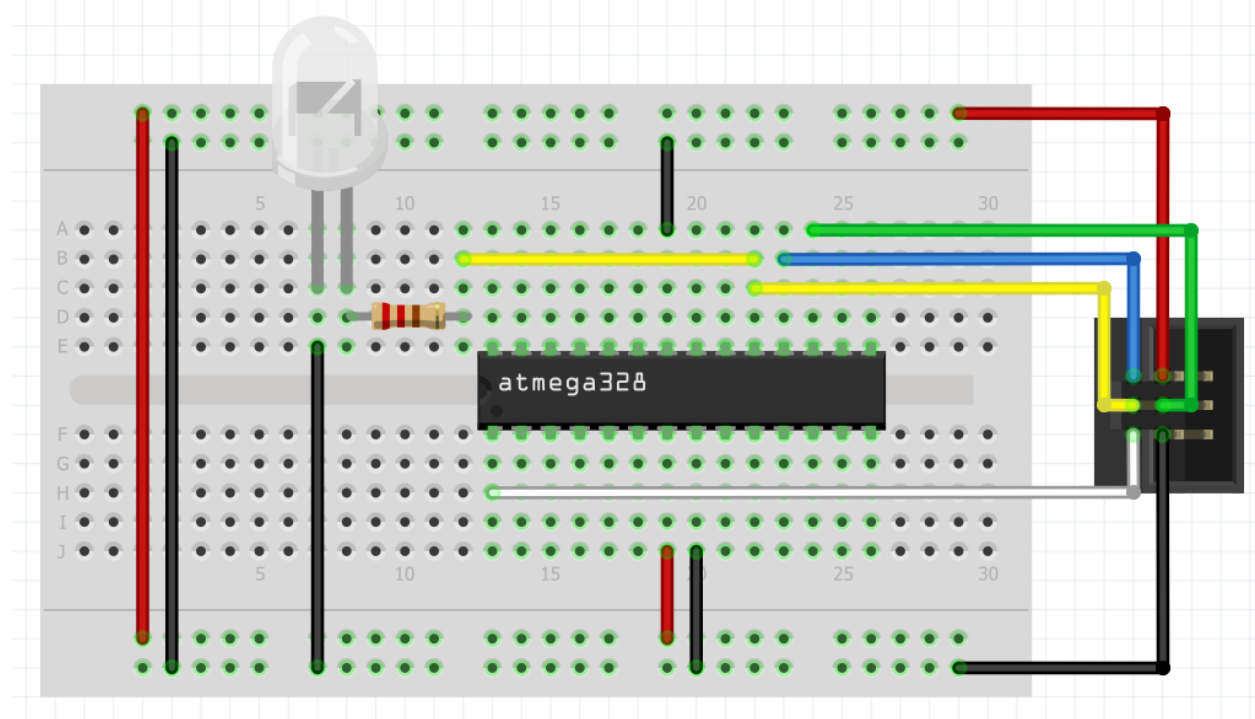
Vous pourriez brancher le programmeur à un Arduino Uno pour tester, mais je recommande de faire un branchement à une puce indépendante au moins une fois pour comprendre le processus.

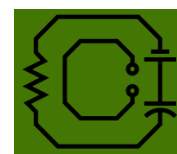
Si vous utilisez un Arduino, assurez-vous simplement de brancher le programmeur dans la bonne direction pour éviter de risquer de briser votre devboard.

### 4.2.1. Branchement à une puce

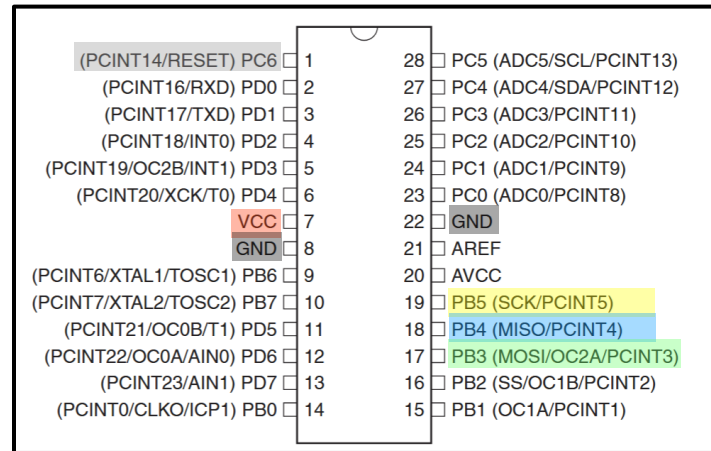
C3I a cinq Atmega328 à votre disposition pour que vous pratiquiez ce branchement. Nous utilisons aussi une DEL (avec une résistance) afin de valider que la puce est bel et bien programmée.

<b>Matériel nécessaire :</b> <ul style="list-style-type: none"><li>▪ Programmeur usbasp</li><li>▪ Atmega328</li><li>▪ Résistance 10k<math>\Omega</math></li><li>▪ Résistance 220<math>\Omega</math></li><li>▪ DEL</li><li>▪ Plusieurs fils uni brin</li><li>▪ Breadboard</li></ul>	<b>Branchement :</b> <ul style="list-style-type: none"><li>▪ Fil rouge : VCC</li><li>▪ Fil vert : MOSI</li><li>▪ Fil noir : GND</li><li>▪ Fil bleu : MISO</li><li>▪ Fil jaune : CLK ( / PB5)</li><li>▪ Fil blanc : RESET</li></ul> <p>*PB5 est la builtin LED + CLK SPI</p>
--	---



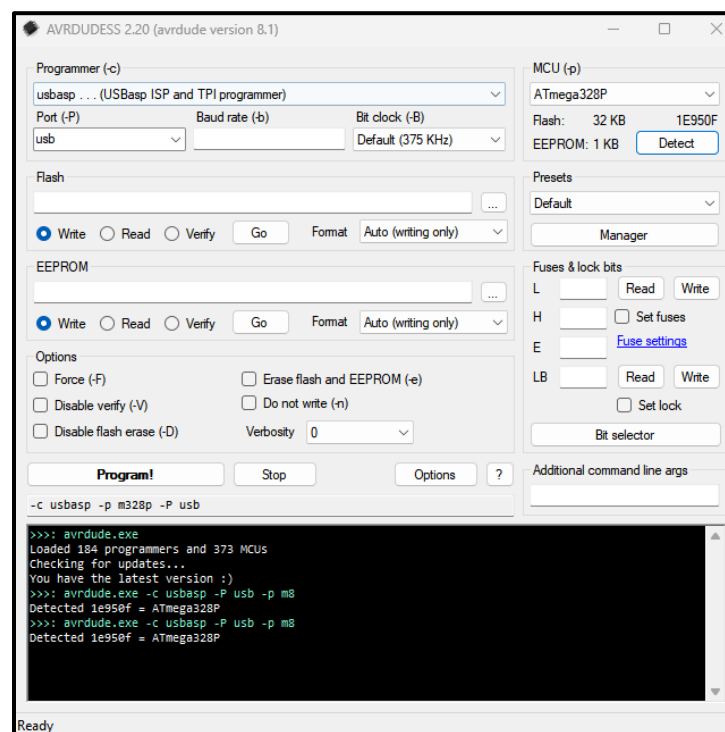


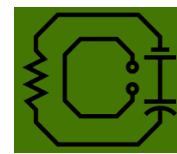
Le branchement est basé sur les broches SPI (MOSI, MISO, SCK) de l'atmega328 ainsi que sa broche RESET (nécessite un pullup à VCC, est forcée à GND par le programmeur lorsque le code est envoyé). Assurez-vous aussi de brancher l'alimentation aux bons endroits, et d'avoir les deux masses (GND) connectées.



## 4.2.2. Validation du branchement

Une fois que vous êtes sûrs de ne pas avoir de courts-circuits ou d'une alimentation au mauvais endroit, vous pouvez valider votre branchement sur AVRdudess. Si vous cliquez sur « Detect » dans la section MCU et que le branchement est fonctionnel, l'afficheur devrait indiquer « Detected [ID] = [MCU] »





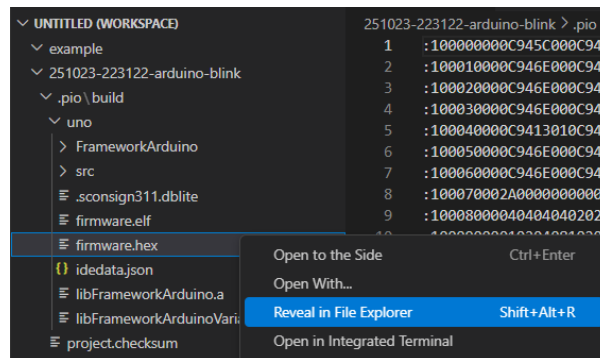
### 4.3. Programmer!

Sur AVRdudess, allez chercher le firmware.hex correspondant à votre code, puis pesez sur « Program ! ». La DEL devrait clignoter.

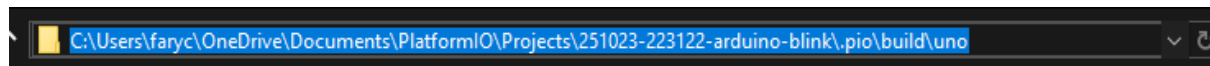
Une fois ce setup complété, vous n'avez plus à y toucher! Dès que vous faites des changements à votre code, vous avez simplement à recompiler (sur VS Code), puis à reprogrammer (sur AVRdudess) : puisque le code HEX est modifié par le compilateur, AVRdudess va envoyer la version modifiée du code à la puce.

#### Note #1 :

Pour savoir où est votre firmware.hex, vous pouvez simplement aller sur VS Code, cliquer droit sur le firmware.hex dans votre arborescence, puis choisir « Reveal in File Explorer ».



Par la suite, vous d'avez qu'à copier le PATH qui est dans votre file explorer, et à le coller dans la section Flash.



Assurez-vous cependant de sélectionner le « firmware.hex » quand vous le mettez dans Flash, et non seulement avoir le path. Sinon, AVRdudess ne sait pas quel fichier il est sensé envoyer (c'est à toi de le dire!).

#### Note #2 :

Si vous êtes observateur, vous allez remarquer que la DEL ne clignote pas à un intervalle de 1 seconde ON, 1 seconde OFF tel que le code demande de le faire. C'est parce que le firmware Arduino assume que vous travaillez sur un Arduino Uno, qui a un oscillateur de 16 MHz sur le PCB. Cependant, vous n'avez pas ce oscillateur sur votre breadboard – votre Atmega n'a donc pas le bon timing!



## 5. Ressources et questions

Besoin d'aide avec la programmation AVR ? Nous avons des **mentors**, incluant un expert dans le sujet ! N'hésitez pas à communiquer avec **Bramptonn** sur Discord, qui m'a aidé à construire cet atelier. Il est toujours prêt à aider les participants, et est très à l'aise avec les puces de type Atmel ainsi que les ESP !

Pour toute autre question, veuillez communiquer avec les responsables de cette année : [C3I@groupe.usherbrooke.ca](mailto:C3I@groupe.usherbrooke.ca)

Pour tout commentaire sur le document (faute, incohérence, etc), merci de communiquer avec le responsable qui a rédigé le document.