Passion Electronique

Comment programmer un ATmega328P ou y graver un bootloader, avec un Arduino Uno ou un FTDI ?



Vous est-il déjà arrivé de concevoir des projets à base d'Arduino (Uno, Nano, Pro Mini, …), qui au final prenaient trop de place ? Alors pourquoi ne pas simplement **conserver uniquement le cœur de ces cartes (le microcontrôleur, j'entends), tout en virant le reste** ? Ainsi, vous auriez des montages plus compacts, et plus économes ! C'est en tout cas ce que je vous propose de voir ensemble, aujourd'hui 😉

Ici, je vais vous présenter le microcontrôleur présent dans les cartes Uno et Nano, à savoir l'ATmega328P ! Aussi, si vous souhaitez l'utiliser seul (en dehors de toute carte arduino, j'entends),

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Accepter Refuser

ailleurs quelle est la configuration matérielle minimale requise pour faire tourner ce µC. Alors ça vous dit ? Si oui, en avant !

Ce contenu vous plaît ? Alors abonnez-vous à la Newsletter pour ne rien louper !

📜 Recevoir la Newsletter !

*Remarque : cet article est une **introduction à la programmation de l'ATMEGA328P**, qui va vous montrer 4 façons de programmer un ATmega328P. Ici, j'ai souhaité vous transmettre des bases pratiques, afin que vous puissiez faire vos premiers pas avec ce µC (en dehors de toute carte arduino, donc). Du reste, je vous partagerai plus tard quelques réalisations pratiques, mettant en œuvre ce microcontrôleur « seul », pour vous montrer quelques exemples d'intégration pratique, au sein de montage électroniques !*

Sommaire [Masquer]

1. Matériel minimal, pour faire tourner un microcontrôleur ATmega328P
2. Bootloader : c'est quoi ? est-ce nécessaire ?
3. Charger / graver un bootloader sur l'ATmega328P (si besoin)
4. Méthode #1 : programmation d'un ATmega328P en l'embrochant directement sur une carte Arduino Uno (bootloader requis)
5. Méthode #2 : programmation d'un ATmega328P via un Arduino Uno sans µC (bootloader requis)
6. Méthode #3 : programmation d'un ATmega328P via un module FTDI USB/TTL type FT232RL (bootloader requis)
7. Méthode #4 : programmation d'un ATmega328P en ISP, avec un Arduino Uno transformé en « programmateur »
8. Programmer son ATmega328P : conclusion !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

IMAGE	DESCRIPTION	LIEN ACHAT
	Microcontrôleur ATmega328P (format DIP, 28 broches)	

1. Matériel minimal, pour faire tourner un microcontrôleur ATmega328P

Avant toute chose, sachez qu'**un microcontrôleur ATmega328P peut être cadencé à différentes vitesses, et de différentes façons** ! En effet, ce µC peut être cadencé par :

- un quartz externe (c'est ce qui est le plus communément utilisé, à 16 MHz sur les cartes Arduino Uno et Nano)
- l'oscillateur interne (à 8 MHz, par exemple, via l'oscillateur RC interne)
- une horloge externe, arrivant directement sur l'entrée XTAL1 du microcontrôleur

Ici, nous travaillerons comme si nous étions dans une carte Arduino, c'est à dire avec un quartz externe de 16 MHz (muni de ses deux indispensables condensateurs, de 22pF chacun).

Sachez également que **le microcontrôleur ATmega328P doit être alimenté via une tension DC comprise entre 2,7 et 5,5 volts**. À noter que, selon la fréquence à laquelle vous souhaitez travailler, le µC aura besoin d'une tension minimale, pour « bien fonctionner » (par exemple : fournir une alimentation comprise entre 4,5 à 5,5 volts, si vous souhaiter tourner à 16 MHz). Plus globalement, voici un graphe représentant la tension minimale à fournir en fonction de la fréquence de fonctionnement, tiré du datasheet :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Du reste, voici plus globalement un exemple de **circuit minimal pour faire fonctionner l'ATmega328P** :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Grosso-modo, on retrouve :

- les broches d'alim VCC (7) et AVCC (20), reliées au « + » de l'alimentation (entre 4,5 et 5,5 volts dans notre cas, car nous allons fonctionner à 16 MHz)
- les broches de masse GND (8 et 22), reliées au « – » de l'alimentation (0 volt, si vous préférez)
- les broches de raccordement à l'oscillateur externe XTAL1 (9) et XTAL2 (10)
- la broche de /RESET (1), ramenée à l'état haut via une résistance pull-up (pour rappel, la ligne /RESET est active à l'état bas, donc quand on la met à la masse)

Bien entendu, **tout ceci n'est qu'une base de travail, qu'il faudra compléter ensuite**, au besoin 😉

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Remarque : après coup, je m'aperçois que j'ai oublié de mettre le condensateur de découplage de 100 nF, entre les broches Vcc et Gnd de l'ATmega328P. Bien sûr, cela peut fonctionner sans ; mais pour éviter tout « bug » du microcontrôleur, ne l'oubliez pas (surtout si vous faites un montage plus élaboré !).

Maintenant que nous avons vu la partie matérielle, passons à la partie logicielle ! Et au passage, voyons la notion de « bootloader », qui fait si peur à tant de monde 😉

Le saviez-vous ? lorsqu'on achète un ATmega328P neuf (« brut de fonderie », comme on dit), il est « vierge » de tout programme (au niveau de sa mémoire flash, j'entends). Il ne contient donc ni bootloader (dont je vous parlerai juste après), ni programme particulier. Mais sachez qu'**il existe parfois à la vente des ATmega328P préprogrammés (« pre-burned »), disposant d'un bootloader gravé en mémoire, et parfois même, un programme « Blink » pour la « démo », en sus** ! Du reste, dans le présent article, je ne vous parlerai ici que ATmega vierges, c'est à dire livrés sans programme ni bootloader, afin de pouvoir vous présenter un maximum de choses, et faire le tour le plus complet possible de la bête !

IMAGE	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

2. Bootloader : c'est quoi ? est-ce nécessaire ?

Avant d'aborder la programmation du microcontrôleur ATmega328P en elle-même, il y a un point important à savoir avant tout.

En fait, de base et si je simplifie les choses, **la mémoire programme d'un ATmega328P ne peut se programmer que de 2 différentes manières** :

- en parallèle (ce qui est « complexe » à mettre en œuvre, nécessite beaucoup de broches, et n'est pas vraiment adapté à nous, électroniciens amateurs ou bidouilleurs, de mon point de vue)
- ou en série (via le port SPI, nativement, au travers des broches SCK, MOSI, et MISO). Cela nécessite d'avoir un petit programmateur adapté, de type ICSP (de l'anglais « In-Circuit System Programming », qui permet la programmation « in situ », c'est à dire à même notre montage/projet)

Ces moyens de programmation étant limités et/ou monopolisant un certain nombre de broches, il a été fait le choix d'ajouter une autre voie de programmation série (de type UART cette fois-ci, donc au travers des lignes RX et TX). Ce moyen ne pouvant pas être natif au niveau matériel (hardware), il a fallu trouver un moyen de l'implémenter au niveau logiciel (software). **C'est ainsi qu'à vu le jour un microprogramme pour ATmega328P, permettant d'ajouter un accès pour la programmation série UART ; ce petit programme s'appelle un « bootloader »**, en ce sens où le microcontrôleur l'exécutera en premier au démarrage (boot), et que ce microprogramme permettra soit l'exécution du programme principal logé en mémoire, soit le chargement d'un nouveau programme (loader), via le port UART.

Le bootloader logera à même la mémoire programme de l'ATmega328P, dans une zone qui peut lui être réservée. En effet, la mémoire flash (mémoire programme) de l'ATmega328P peut se diviser ainsi :

- une zone « principale », qui contiendra notre **programme à exécuter**
- une zone « secondaire », qui contiendra le **logiciel de démarrage du µC (le fameux « bootloader »)**

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

On retrouve bien nos 2 zones mémoire distinctes, à savoir :

- **une partie principale, nommée « Application Flash Section »**, dans laquelle on pourra enregistrer notre programme à exécuter
- **une partie secondaire, nommée « Bootloader Flash Section »**, dans laquelle sera inscrite notre logiciel de démarrage (bootloader) ; à noter que celui-ci ne sera généralement jamais retouché, une fois enregistré, sauf à vouloir le mettre à jour, le changer, ou le retirer !

Le microcontrôleur sera configuré de manière à toujours exécuter le bootloader au

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

doit lancer le programme contenu en mémoire principale, ou enregistrer un nouveau programme en mémoire principale, si une demande lui est « formulée » en ce sens, via le port UART (via ses lignes TX et RX, donc). À noter que le bootloader effectuera un RESET systématique du µC, après téléversement de tout nouveau programme, afin de se réinitialiser, puis lancer « bien proprement » le nouveau programme principal !

Donc si je résume, et fait la synthèse de tout ce que nous avons vu au-dessus :

- **un ATmega328P sans bootloader ne pourra être programmé qu'en parallèle, ou en série via le port SPI** (broches SCK, MISO, MOSI, RESET, avec protocole STK500, pour ce dernier)
- **un ATmega328P avec bootloader pourra quant à lui être programmé d'une façon supplémentaire, de manière « logicielle », via le port UART** (broches RX et TX du microcontrôleur)

Remarque : au final, le bootloader n'est qu'un programme « secondaire » permettant de charger des programmes « principaux », en offrant au passage un mode de programmation supplémentaire (le port UART) !

Note technique : l'espace réservé pour le bootloader en mémoire se définit au niveau du fusible « high fuse » de l'ATmega328P, plus précisément via les bits BOOTSZ1 et BOOTSZ0, de ce fusible. **Dans notre cas, où nous allons graver le bootloader « optiboot » sur notre ATmega328P (cf. paragraphe suivant), ces bits seront automatiquement mis à 1 par l'IDE Arduino ; ce qui signifiera que 512 octets (cf. tableau ci-dessus) seront alloués pour le bootloader**. La mémoire flash de l'ATmega328P faisant 32 Ko (allant donc de l'adresse 0x0000 à 0x7FFF), la segmentation mémoire se fera ainsi :
– mémoire « principale », pour le programme, ira de l'adresse 0x0000 à l'adresse 0x7DFF (word 0x0000 à word 0x3EFF, dans la doc)
– mémoire « secondaire », pour le bootloader, ira de l'adresse 0x7E00 à l'adresse 0x7FFF (word 0x3F00 à word 0x3FFF, dans la doc)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Juste avant de vous expliquer **comment charger/graver un bootloader sur l'ATmega328P**, je tenais simplement à vous rappeler que l'utilisation d'un bootloader n'est pas obligatoire en soi. En effet, pour rappel, vous pouvez parfaitement programmer votre ATmega328P via le port SPI à l'aide d'un programmateur externe, et ainsi, vous n'auriez pas besoin d'un bootloader.

Cela étant dit, **si vous souhaitez programmer votre ATmega328P via le port UART (lignes TX/RX), alors vous aurez préalablement besoin de graver un bootloader dessus (via le port SPI), pour ensuite pouvoir uploader tous vos programmes que vous souhaitez en suivant, via le port UART** (un bootloader ne pouvant pas se charger via le port UART, du fait que ce mode de programmation n'est pas natif sur l'ATmega328P, et qu'un bootloader ne pourrait pas non plus se modifier en cours d'exécution, sans altérer son fonctionnement).

À présent, voyons la partie qui nous intéresse, à savoir : comment graver un bootloader dans un ATmega328 !

Dans ce tuto, je vais vous présenter une façon de faire (parmi tant d'autres), **en transformant un Arduino Uno en « programmateur d'ATmega328P »** ! Ici, la carte Arduino Uno sera transformée en simple « passerelle de programmation », qui nous permettra d'écrire à même la mémoire flash (mémoire programme) de notre ATmega328P. Pour ce faire, voici les 2 étapes à suivre, donc :

1. transformer un Arduino Uno en passerelle de programmation (« Arduino as ISP », comme ils disent, ISP voulant dire « In-System-Programmer »)
2. puis envoyer (graver en mémoire) le bootloader dans notre ATmega328P, au travers de cette carte Arduino Uno

> Le saviez-vous ? un bootloader « emprunte » une partie de la mémoire flash de l'ATMEGA328P, ce qui « restreint » la place disponible pour notre programme à exécuter. Dans le cas présent, par exemple, notre bootloader aura une zone de 512 octets qui lui sera réservée en mémoire programme, sur les 32768 octets disponibles (32 Ko). En clair, 512/32768 soit environ 1,5 % d'espace mémoire sera « perdu », ne laissant plus « que » 98,5 % d'espace libre pour le programme principal en lui-même ! Ne riez pas, car on a vite fait de remplir la mémoire d'un microcontrôleur, dans certains cas … !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Alors, **pour transformer une carte Arduino Uno en programmateur d'ATmega328P**, rien de plus simple ! En effet, il suffit simplement de suivre les étapes suivantes, par exemple :

- brancher votre Arduino Uno sur votre PC (important: la carte arduino ne doit être reliée à rien d'autre que votre ordi, c'est à dire sans connexion aucune, que ce soit au niveau des broches analogiques ou des broches digitales)
- ouvrir l'IDE Arduino
- cliquer sur le menu Fichier > Exemples > 11.ArduinoISP > ArduinoISP, afin que **le code du « programmateur ISP » s'ouvre à l'écran**
- cliquer sur le menu Outils > Port > et sélectionner le port COM de l'Arduino Uno que vous venez de brancher sur votre PC
- cliquer sur le menu Outils > Type de carte > et sélectionner « Arduino Uno » (qui nous servira de programmateur, donc)
- cliquer sur le menu Outils > Programmateur > et vérifier que « AVRISP mkII » est bien sélectionné
- cliquer sur l'icône « Téléverser » (ou aller dans le menu Croquis > Téléverser), pour **envoyer ce programme dans la carte Arduino Uno, pour qu'elle devienne un « programmateur ISP »**
- et débrancher votre carte Arduino Uno de votre PC (très important, pour que le câblage qui suit se fasse « hors tension »)

Ça y est, **votre Arduino Uno est désormais « transformé » en « programmateur ISP »** ! Vous pourrez ainsi vous en servir comme passerelle de programmation à l'avenir, pour programmer tous vos ATmega328P, que ce soit pour y graver un bootloader (comme détaillé ici), ou directement uploader un programme dans la mémoire principale (comme nous verrons au paragraphe §7).

> Nota : la transformation de la carte Arduino Uno n'est pas définitive, rassurez-vous ! En fait, ici, vous avez simplement enregistré un programme particulier en mémoire (celui d'un « programmateur ISP »), et bien entendu, vous pouvez à tout moment l'effacer ou le remplacer par un autre programme, une fois que vous aurez fini de programmer votre/vos ATmega328P !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

À présent que **nous avons un Arduino Uno transformé en « programmateur fait-maison »**, nous pouvons maintenant procéder à la gravure (l'enregistrement en mémoire) d'un bootloader au sein de l'ATmega328P.

Pour cela, il convient de **réaliser le montage suivant** (attention : le câblage doit se faire hors tension, c'est à dire que votre Arduino Uno doit être débranchée de votre PC) :

Comparé au « circuit minimal » présenté au §1 (faisant basiquement tourner un ATmega328P), seuls quelques fils sont rajoutés :

- de la ligne D10 de l'Arduino Uno (/ss) vers la broche 1 (/reset) de l'ATmega328P
- de la ligne D11 de l'Arduino Uno (mosi) vers la broche 17 (mosi) de l'ATmega328P
- de la ligne D12 de l'Arduino Uno (miso) vers la broche 18 (miso) de l'ATmega328P
- de la ligne D13 de l'Arduino Uno (sck) vers la broche 19 (sck) de l'ATmega328P

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

IMAGES	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF – **Résistance** pull-up RESET de 10 kohms	
	– **Arduino Uno** R3 (ou carte équivalente)	

Remarque technique : même si les lignes utilisées ici correspondent à celles d'un bus SPI, il ne s'agit pas là d'une « vraie » communication SPI en soi (c'est pourquoi le MOSI de l'un est relié au MOSI de l'autre, etc.). En fait, il s'agit d'une communication particulière, basée sur le protocole STK500, dans notre cas.

Une fois le montage câblé sur breadboard, voici ce à quoi ça ressemble :

Et **du point de vue logiciel, voici les étapes à suivre** :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

- cliquer sur le menu Outils > Port > et sélectionner le port COM de l'Arduino Uno que vous venez de brancher sur votre PC
- cliquer sur le menu Outils > Type de carte > et vérifier que « Arduino Uno » est bien sélectionné (remarque technique : ici, c'est le type de carte cible qu'on sélectionne ; ce qui veut dire que nous allons faire « croire » à l'IDE Arduino que notre ATmega328P est une carte Arduino Uno, en quelque sorte !)
- cliquer sur le menu Outils > Programmateur > et sélectionner « Arduino as ISP », cette fois-ci (au passage, ne vous trompez pas avec les noms très ressemblants à côté, tel que « ArduinoISP » et autre !)
- cliquer sur le menu Fichier > Préférences > et cocher la case « téléversement » (figurant à droite du texte « Afficher les résultats détaillés pendant ») ; cela vous permettra d'avoir tout plein d'infos détaillées au moment de la gravure, facilitant ainsi le débuggage, en cas de problème
- puis cliquer sur Outils > Graver la séquence d'initialisation

Si tout s'est bien passé, vous devriez obtenir un écran ressemblant à cela (seule la « partie basse » nous intéresse ici) :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Avec un texte log comme suit (juste pour info : le chemin d'accès à vos fichiers sera probablement différent, car là ça porte mon prénom … !)

```
C:\Users\JEROME\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-a

avrdude: Version 6.3-20190619
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is "C:\Users\JEROME\AppData\Local\Ardu

         Using Port                    : COM10
         Using Programmer              : stk500v1
         Overriding Baud Rate          : 19200
         AVR Part                      : ATmega328P
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
         BS2                           : PC2
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :

                                  Block Poll               Page
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  M
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- -
           eeprom        65    20     4    0 no       1024    4      0  3600
           flash         65     6   128    0 yes     32768  128    256  4500
           lfuse          0     0     0    0 no          1    0      0  4500
           hfuse          0     0     0    0 no          1    0      0  4500
           efuse          0     0     0    0 no          1    0      0  4500
           lock           0     0     0    0 no          1    0      0  4500
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
            Description     : Atmel STK500 Version 1.x firmware
            Hardware Version: 2
            Firmware Version: 1.18
            Topcard         : Unknown
            Vtarget         : 0.0 V
            Varef           : 0.0 V
            Oscillator      : Off
            SCK period      : 0.1 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.02s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: erasing chip
avrdude: reading input file "0x3F"
avrdude: writing lock (1 bytes):

Writing | ################################################## | 100% 0.01s

avrdude: 1 bytes of lock written
avrdude: verifying lock memory against 0x3F:
avrdude: load data lock data from input file 0x3F:
avrdude: input file 0x3F contains 1 bytes
avrdude: reading on-chip lock data:

Reading | ################################################## | 100% 0.01s

avrdude: verifying ...
avrdude: 1 bytes of lock verified
avrdude: reading input file "0xFD"
avrdude: writing efuse (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of efuse written
avrdude: verifying efuse memory against 0xFD:
avrdude: load data efuse data from input file 0xFD:
avrdude: input file 0xFD contains 1 bytes
avrdude: reading on-chip efuse data:

Reading | ################################################## | 100% 0.01s
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
avrdude: writing hfuse (1 bytes):

Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of hfuse written
avrdude: verifying hfuse memory against 0xDE:
avrdude: load data hfuse data from input file 0xDE:
avrdude: input file 0xDE contains 1 bytes
avrdude: reading on-chip hfuse data:

Reading | ################################################## | 100% 0.01s

avrdude: verifying ...
avrdude: 1 bytes of hfuse verified
avrdude: reading input file "0xFF"
avrdude: writing lfuse (1 bytes):

C:\Users\JEROME\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-a

avrdude: Version 6.3-20190619
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is "C:\Users\JEROME\AppData\Local\Ardu

         Using Port                    : COM10
         Using Programmer              : stk500v1
         Overriding Baud Rate          : 19200
Writing | ################################################## | 100% 0.02s

avrdude: 1 bytes of lfuse written
avrdude: verifying lfuse memory against 0xFF:
avrdude: load data lfuse data from input file 0xFF:
avrdude: input file 0xFF contains 1 bytes
avrdude: reading on-chip lfuse data:

Reading | ################################################## | 100% 0.01s

avrdude: verifying ...
avrdude: 1 bytes of lfuse verified

avrdude done.  Thank you.
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
BS2                           : PC2
RESET disposition             : dedicated
RETRY pulse                   : SCK
serial program mode           : yes
parallel program mode         : yes
Timeout                       : 200
StabDelay                     : 100
CmdexeDelay                   : 25
SyncLoops                     : 32
ByteDelay                     : 0
PollIndex                     : 3
PollValue                     : 0x53
Memory Detail                 :

                         Block Poll               Page
  Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  M
  ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- -
  eeprom        65    20     4    0 no       1024    4      0  3600
  flash         65     6   128    0 yes     32768  128    256  4500
  lfuse          0     0     0    0 no          1    0      0  4500
  hfuse          0     0     0    0 no          1    0      0  4500
  efuse          0     0     0    0 no          1    0      0  4500
  lock           0     0     0    0 no          1    0      0  4500
  calibration    0     0     0    0 no          1    0      0     0
  signature      0     0     0    0 no          3    0      0     0

Programmer Type : STK500
Description     : Atmel STK500 Version 1.x firmware
Hardware Version: 2
Firmware Version: 1.18
Topcard         : Unknown
Vtarget         : 0.0 V
Varef           : 0.0 V
Oscillator      : Off
SCK period      : 0.1 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.02s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be perfo
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
183
184 Writing | ################################################## | 100% 0.00s
185
186 avrdude: 32768 bytes of flash written
187 avrdude: verifying flash memory against C:\Users\JEROME\AppData\Local\Arduino1
188 avrdude: load data flash data from input file C:\Users\JEROME\AppData\Local\Ar
189 avrdude: input file C:\Users\JEROME\AppData\Local\Arduino15\packages\arduino\h
190 avrdude: reading on-chip flash data:
191
192 Reading | ################################################## | 100% 0.00s
193
194 avrdude: verifying ...
195 avrdude: 32768 bytes of flash verified
196 avrdude: reading input file "0x0F"
197 avrdude: writing lock (1 bytes):
198
199 Writing | ################################################## | 100% 0.02s
200
201 avrdude: 1 bytes of lock written
202 avrdude: verifying lock memory against 0x0F:
203 avrdude: load data lock data from input file 0x0F:
204 avrdude: input file 0x0F contains 1 bytes
205 avrdude: reading on-chip lock data:
206
207 Reading | ################################################## | 100% 0.01s
208
209 avrdude: verifying ...
    avrdude: 1 bytes of lock verified

    avrdude done.  Thank you.
```

Bravo ! **Votre ATmega328P est désormais doté d'un bootloader**, ce qui vous permettra ensuite de le programmer via les broches TX et RX (port UART) si besoin !

Nota 1 : si jamais vous constatez une erreur de gravure, avec **une mention du type « Device signature = 0x000000 » dans le retour log, alors vous avez très probablement un faux contact quelque part** ; ce peut être au niveau d'un fil, d'une mauvaise connexion, ou d'un composant défectueux. J'ai d'ailleurs eu ce genre de soucis

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

contact. Comme quoi, la réussite ou l'échec ne tient vraiment à pas grand chose, parfois 😉

Nota 2 : si jamais vous voyez **le texte « Device signature = 0x1E9516 » au lieu de « Device signature = 0x1E950F » dans ce retour log, c'est sans doute parce que vous avez utilisé un « ATmega328PB » au lieu d'un « ATmega328P »** (ce qui provoque l'échec de la gravure, dans notre cas). Car ce tuto est fait pour la programmation de l'ATmega328P ; vous aurez donc besoin de vous munir de cette référence-ci, pour poursuivre !

3.3 - Comment savoir si le bootloader a bien été gravé dans le µC ATmega328P ?

Il y a un moyen très simple de **vérifier la présence ou non d'un bootloader, sur un ATmega328P** !

En effet, si un bootloader est présent dans l'ATmega328P, alors une LED branchée sur D13 (broche 19 de l'ATmega328P) clignoterait 3 fois (très rapidement), au démarrage et après chaque reset (quoi que au démarrage, ça peut ne pas se produire, dans certains cas ; le mieux est donc forcer le reset, pour voir la réaction du µC).

Voici **comment câbler une LED « témoin de bon fonctionnement » et ajouter un bouton-poussoir de RESET sur le microcontrôleur ATmega328P** :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Ce qui peut donner, une fois câblé sur breadboard :

Pour la vérif, appuyez puis relâchez le bouton poussoir RESET, tout en observant la LED ; ainsi, **si un bootloader est présent dans le µC, alors cette led clignotera 3 fois**, très rapidement, juste après le RESET. Sinon, si le bootloader est absent, alors aucun triple clignotement au démarrage ne surviendra !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

au paragraphe 3.1 (mais on aurait pu prendre autre chose, bien entendu). Au passage, vous pouvez retrouver le **code de tous les bootloader référencés sur le site GitHub Arduino** à cette adresse, si vous souhaitez en apprendre plus, sur le sujet. Et pour savoir quel bootloader est normalement installé sur tel ou tel type de carte, voici la **liste des cartes avec bootloader correspondant, référencées sur le site GitHub Arduino**. Enfin, pour votre culture, sachez que l'IDE Arduino a modifié les fusibles de l'ATmega328P de la manière suivante, au moment de la gravure de ce bootloader :

– uno.bootloader.**low_fuses=0xFF**

– uno.bootloader.**high_fuses=0xDE**, soit 0b1101 1110 en binaire ; nota : ici le bit 2 (BOOTSZ1) = 1 et le bit 1 (BOOTSZ0) = 1, ce qui signifie que 512 octets seront réservés pour notre bootloader ; on a également le bit 0 (BOOTRST) = 0, ce qui signifie entre autre que le µC démarrera immédiatement à l'adresse du bootloader, et non à l'adresse 0x0000, où se situe le programme principal)

– uno.bootloader.**extended_fuses=0xFD**

– uno.bootloader.**unlock_bits=0x3F**

– uno.bootloader.**lock_bits=0x0F**

Avec uno.bootloader.**file=optiboot/optiboot_atmega328.hex**, dans notre cas !

Nota : si la programmation ou la configuration des fusibles de l'ATmega328P vous intéresse, vous retrouvez un excellent site ici, permettant de voir leur décomposition / choix possible / signification des bits : **https://eleccelerator.com/fusecalc/fusecalc.php?chip=atmega328p**. Par exemple, si on reprend nos valeurs de fusible ci-dessus, et qu'on regarde plus précisément **lorsque « high fuse = 0xDE », alors on constate que « Boot Flash section size=256 words Boot start address=$3F00; [BOOTSZ=11] » est sélectionné, ce qui correspond à nos 512 octets d'espace réservé, pour notre bootloader** (du fait que 1 word = 2 octets, pour rappel !).

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

IMAGE	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF – **Résistances** (1 et 10 kohms) – **Led**, peu importe la couleur ici – **Bouton poussoir**, modèle 4 broches 12x12x7.3 par exemple – **Condensateur de découplage** 100nF (recommandé, mais non obligatoire)	

4. Méthode #1 : programmation d'un ATmega328P en l'embrochant directement sur une carte Arduino Uno (bootloader requis)

Prérequis :

- avoir sous la main **une carte Arduino Uno** avec microcontrôleur débrochable (monté sur support, comme visible ci-dessous)
- avoir sous la main **un ATmega328P** sur lequel vous avez gravé déjà un bootloader (comme détaillé dans cet article, au paragraphe §3)

Alors, pour **programmer un ATmega328P (déjà équipé de bootloader) sur une carte Arduino Uno (à microcontrôleur débrochable)**, voici premièrement les étapes à suivre, sur le plan matériel :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

En bref, vous devez donc sortir le µC présent sur la carte Arduino Uno, pour le remplacer par l'ATmega328P à programmer (et à la toute fin, une fois ce dernier programmé, vous pourrez le retirer de la carte arduino … mais pas avant !).

> Important : l'encoche présente sur le circuit intégré (indiquant où se situe la « broche 1 » du microcontrôleur de la carte Arduino Uno), doit se trouver à droite, comme mis en évidence sur les photos ci-dessus (au travers des indications spécifiées en jaune, plus précisément).

Ensuite, **une fois la permutation des µC faite, il ne reste plus qu'à procéder au téléversement de votre programme**. Pour ce faire, voici les étapes à suivre :

- brancher votre carte Arduino Uno, nouvellement munie d'un ATmega328P, sur votre PC
- ouvrir l'IDE Arduino
- ouvrir le programme Blink, accessible via le menu Fichier > Exemples > 01.Basics > Blink (nota : vous pouvez remplacer ce programme par n'importe quel autre ; là, c'est juste pour servir d'exemple)
- cliquer dans le menu Outils > Port > et vérifier que le port COM de votre Arduino Uno est bien sélectionné
- cliquer dans le menu Outils > Type de carte > et sélectionner « Arduino Uno »
- cliquer dans le menu Outils > Programmateur > et s'assurer que « AVRISP mkII » est bien sélectionné (comme il doit l'être par défaut, en temps « normal », lorsqu'on souhaite téléverser un programme dans une carte arduino)
- téléverser le programmer (en cliquant sur le menu Croquis > Téléverser, ou sur l'icône/raccourci en haut à gauche, dans la barre d'outil)
- une fois fait, fermer l'IDE Arduino

> Nota : ici, veillez bien à sélectionner « AVRISP mkII » pour la programmation de la carte arduino (il se peut qu'il y ait « Arduino as ISP » à la place, si vous venez de graver un bootloader, par exemple)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Autre moyen de vérifier que tout marche : sortir l'ATMEGA328P de la carte Arduino, et le monter sur une breadboard, en suivant le montage suivant :

Pour info, sur ce schéma, la LED branchée sur la broche 19 de l'ATmega328P est un copier/coller de la LED présente sur la broche D13 de la carte Arduino Uno. Ainsi, si vous alimentez ce montage (sans la moindre carte arduino, donc), vous devriez voir la LED clignoter, toutes les secondes environ (prouvant là aussi que le programme Blink a bien démarré !).

IMAGES	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

IMAGES	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Condensateur de découplage** de 100nF (non obligatoire, mais recommandé) – **Arduino Uno** R3 (ou carte équivalente)	

5. Méthode #2 : programmation d'un ATmega328P via un Arduino Uno sans µC (bootloader requis)

Prérequis :

- être muni d'une carte Arduino Uno avec microcontrôleur débrochable (comme illustré ci-dessous)
- être muni d'un ATmega328P, sur lequel vous avez gravé déjà un bootloader (en suivant la procédure détaillée au paragraphe §3)

Avant tout ici, **il faudra ôter le µC présent sur la carte Arduino Uno** (et ATTENTION À NE PAS OUBLIER DE FAIRE CELA !). Car nous allons câbler l'ATmega328P de telle manière que la carte Arduino Uno « croira » que ce microcontrôleur est en fait implanté en elle. D'ailleurs, en parlant du câblage, le voici :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Comme vous pouvez le constater ci-dessus, les raccordements à faire sont les suivants :

- la pin RESET de l'Arduino Uno est reliée à la broche 1 de l'ATmega328P (sans résistance pull-up, cette fois-ci, car il y en a déjà une sur la carte arduino)
- la pin +5V de l'Arduino Uno est reliée aux broches Vcc de l'ATmega328P
- la pin GND de l'Arduino Uno est reliée aux broches Gnd de l'ATmega328P
- la pin RX de l'Arduino Uno est reliée à la broche 2 (RX) de l'ATmega328P
- la pin TX de l'Arduino Uno est reliée à la broche 3 (TX) de l'ATmega328P

Nota : il n'y a pas de croisement des lignes RX/TX → TX/RX sur ce schéma, car ici, il n'y a pas de communication UART entre la carte Arduino Uno et l'ATmega328P. Le câblage est direct, car on cherche à faire croire à la carte arduino que notre ATmega328P est

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Ensuite, au niveau logiciel, voici un programme « blink modifié » (faisant des clignotements doubles, pour le différencier de l'exemple précédent) que nous allons uploader dans notre ATmega328P, à fin d'essais :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
  |   ___/ _ \| __|| __| |/ _ \| '_ \_____|  __|| |/ _ \/  _|  _| '__/   \| '_
  |  |  | ( ) |__ ||__ | | ( ) | | | |____| |__ | |  __/| (_| |_| | | (_) | |
  \__|   \__,_|___||___|_|\___/|_| [_|    \____/|_|\___|\____\__\_|  \___/|_|


  Fichier :       prgArduino-Blink-2clignotements.ino

  Description :   Programme permettant de faire clignoter une LED branchée sur
                  (le clignotement sera doublé, pour bien le distinguer du prc

  Auteur :        Jérôme TOMSKI (https://passionelectronique.fr/)
  Créé le :       17.12.2023

*/

#define pinLED  13    // Nota : la LED sera branchée à la broche 19 de notre A
                      //        (c'est pourquoi c'est noté ainsi dans le code,

// ========================
// Initialisation programme
// ========================
void setup() {
  pinMode(pinLED, OUTPUT);   // Nota : "LED_BUILTIN" cible la pin D3 d'un Ardu
}


// =================
// Boucle principale
// =================
void loop() {

  // 2 clignotements LED
  digitalWrite(pinLED, HIGH);  // 1er allumage LED (en mettant la sortie à l'é
  delay(200);
  digitalWrite(pinLED, LOW);   // 1er extinction LED (en mettant la sortie au
  delay(200);
  digitalWrite(pinLED, HIGH);  // 2ème allumage LED
  delay(200);
  digitalWrite(pinLED, LOW);   // 2ème extinction LED
  delay(200);

  // Puis pause de 2 secondes
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
}
```

Et **la marche à suivre pour programmer l'ATmega328P au travers de la carte Arduino Uno, dépourvue de microcontrôleur interne**, est la suivante :

- brancher votre carte Arduino Uno, avec son montage attenant, sur votre PC
- ouvrir l'IDE Arduino
- copier/coller le code arduino présenté ci-dessus
- cliquer dans le menu Outils > Port > et vérifier que le port COM de votre Arduino Uno est bien sélectionné
- cliquer dans le menu Outils > Type de carte > et vérifier que la carte « Arduino Uno » est bien sélectionnée
- cliquer dans le menu Outils > Programmateur > et s'assurer que « AVRISP mkII » est bien sélectionné
- téléverser le programmer (via le menu Croquis > Téléverser, ou en cliquant sur l'icône/raccourci en haut à gauche, dans la barre d'outil de l'IDE)
- une fois fait, vous pouvez fermer l'IDE Arduino

Si tout a bien fonctionné, **vous devriez voir la LED branchée sur votre ATmega328P clignoter 2 fois d'affilée, avec une pause de 2 secondes entre chaque répétition**. Et là encore, pour vous convaincre que tout fonctionne bien, de manière autonome, vous pouvez ensuite déconnecter votre carte Arduino Uno de votre PC, le débrancher du montage, et tester le montage de manière isolée (comme nous l'avions fait, à la fin de l'exemple précédent ; au passage, n'oubliez pas de remettre une résistance de pull-up sur la ligne RESET, avant de brancher votre ATmega328P sur une source de tension 5V externe !).

IMAGES	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

IMAGES COMPOSANTS UTILISÉS LIEN ACHAT

– **Arduino Uno** R3 (ou carte équivalente)

6. Méthode #3 : programmation d'un ATmega328P via un module FTDI USB/TTL type FT232RL (bootloader requis)

Prérequis :

- avoir un convertisseur USB/TTL modèle FT232RL (de chez FTDI) ou équivalent, monté sur PCB avec headers (par abus de langage, j'ai tendance à l'appeler module FTDI, par la suite)
- avoir un ATmega328P, sur lequel un bootloader est déjà gravé (comme détaillé au paragraphe §3 de présent article, par exemple)

Première chose à faire ici : faire reconnaître le FTDI par votre ordi. Pour ma part, sous Windows 10, je n'ai eu qu'à attendre 2 ou 3 minutes, et les drivers du module FT232RL se sont automatiquement installés. Voici ce que ça donne d'ailleurs en image, avant et après, au niveau du « gestionnaire de périphériques » :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Notez au passage que le port attribué au FTDI apparaît clairement (COM12, dans mon cas) ; retenez-le bien si vous utilisez plusieurs ports COM, car c'est celui-là qu'il faudra cibler dans l'IDE Arduino.

> Nota : si aucun driver n'est automatiquement installé par votre PC, vous pouvez toujours procéder manuellement, en récupérant directement **les pilotes sur le site FTDI** ; suivant l'OS que vous utilisez, et le type de CPU monté sur votre carte mère, vous trouverez certainement le driver qui vous convient !

Seconde chose à faire : câbler le FT232RL (FTDI) à votre ATmega328P. Si comme moi vous faites des essais sur breadboard, voici un exemple de schéma de câblage à suivre :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Et voici ce que ça donne, une fois les raccordements faits sur breadboard :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Comme vous pouvez le constater, **il n'y a pas besoin de carte Arduino ici, car on connecte directement : PC → FTDI → ATmega328P**. À noter que les connexions entre le module FTDI (FT 232 RL) et l'ATmega328P sont les suivantes :

- la pin DTR du FTDI est reliée à la broche 1 (/RESET) de l'ATmega328P (au travers d'un condensateur de 100 nF, pour ne faire qu'une impulsion de reset au début de la programmation, et non un maintien au niveau bas ; au passage, ce condensateur s'utilisera de pair avec une résistance de pull-up sur l'entrée RESET du µC)
- la pin RX du FTDI est reliée à la broche 3 (TX) de l'ATmega328P
- la pin TX du FTDI est reliée à la broche 2 (RX) de l'ATmega328P
- la pin VCC du FTDI est reliée aux broches Vcc de l'ATmega328P
- la pin CTS du FTDI ne sera reliée à rien du tout

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Nota 1 : il y a **croisement des lignes RX/TX → TX/RX ici, afin d'établir une connexion série UART, entre le module FTDI et notre ATmega328P**. En effet, c'est bien une communication série qui sera établie en arrière plan, entre ces deux éléments.

Nota 2 : dans cet exemple, **l'alimentation 5V sera fournie par le convertisseur « USB to TTL » (de chez FTDI, ou équivalent), afin de pouvoir programmer et faire tourner notre ATmega328P**, cadencé à 16 MHz (c'est d'ailleurs cette fréquence qui « impose » une alimentation 5V plutôt que 3,3V, bien que le convertisseur USB/TTL puisse fournir 3V3, afin de respecter les recommandations du fabricant).

IMAGES	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF – **Résistances** (pour RESET et LED) – **Led** (peu importe la couleur, ici) – **Condensateur de découplage** de 100nF (indispensable, sur la ligne DTR)	
	– **Arduino Uno** R3 (ou carte équivalente)	
	– **Convertisseur USB to TTL** (FTDI modèle FT232RL, ou compatible)	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Maintenant que nous avons vu la partie matérielle, passons à la partie logicielle !

Voici encore **un autre programme « blink modifié » (faisant des clignotements très très rapides, cette fois-ci)**, pour faire nos essais de programmation & tests (remplaçable par tout autre programme arduino de votre choix, si tel est votre souhait, bien entendu !) :

```
/*
   _______                 _                    _///_ _            _
  /    _  \              (_)                  |  ___| |          | |
  |  [_|  |__  ____  ____ _  ____  _ __       | |__ | | ____  ____| |_ _ __ ____  _ _
  |   ___/ _ \| __|| __| |/ _ \| '_ \_____|  __|| |/ _ \/  _|  _| '__/   \| '_
  |  |   | ( ) |__ ||__ | | ( ) | | | |____| |__ | |  __/| (_| |_| | | (_) | |
  \__|   \__,_|___||___|_|\___/|_| [_|    \____/|_|\___|\____\__\_|  \___/|_|


  Fichier :       prgArduino-Blink-clignotementsRapides.ino

  Description :   Programme permettant de faire clignoter "très" rapidement un

  Auteur :        Jérôme TOMSKI (https://passionelectronique.fr/)
  Créé le :       18.12.2023

*/

#define pinLED  13    // Nota : la LED sera branchée à la broche 19 de notre A
                      //        (c'est pourquoi c'est noté ainsi dans le code,

// ========================
// Initialisation programme
// ========================
void setup() {
  pinMode(pinLED, OUTPUT);
}


// =================
// Boucle principale
// =================
void loop() {
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
39
40    delay(100);
41
42    // Et rebouclage infini
43
  }
```

Enfin, concernant **les étapes à suivre pour téléverser le programme dans notre ATmega328P**, les voici :

- vérifier que le convertisseur USB/TTL est en position « +5V » (afin de fournir la « bonne » tension à notre montage), et bien connecté suivant le schéma présenté ci-dessus
- brancher le convertisseur à votre PC
- ouvrir l'IDE Arduino
- copier/coller le code présenté ci-dessus (ou tout autre programme, si vous préférez ; là, c'est juste pour faire des tests)
- cliquer dans le menu Outils > Port > et sélectionner le port COM du FTDI (COM12, dans mon cas, comme vu plus haut)
- cliquer dans le menu Outils > Type de carte > et sélectionner la carte « Arduino Uno ». Remarque : **ici, nous n'utilisons AUCUNE carte Arduino Uno ; mais comme nous avons gravé le même bootloader que celui d'une carte Arduino uno (le modèle « optiboot », pour rappel), alors nous pouvons sélectionner ce type de carte**, car c'est du pareil au même, dans ce cas précis.
- cliquer dans le menu Outils > Programmateur > et s'assurer que « AVRISP mkII » est bien sélectionné (ce qui doit être le cas normalement, par défaut)
- téléverser le programmer (via le menu Croquis > Téléverser, ou les touches CTRL+U du clavier)

Si un mauvais branchement a été réalisé sur votre montage, ou s'il y a un mauvais contact quelque part (ne riez pas, ça m'est arrivé, la première fois !), **voici le type d'erreur qui peut apparaître au niveau d'Arduino IDE** :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Et ce qui peut apparaître en bas, dans la fenêtre de log :

```
Le croquis utilise 922 octets (2%) de l'espace de stockage de programmes. Le m
Les variables globales utilisent 9 octets (0%) de mémoire dynamique, ce qui la
C:\Users\JEROME\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-a

avrdude: Version 6.3-20190619
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is "C:\Users\JEROME\AppData\Local\Ardu

         Using Port                    : COM12
         Using Programmer              : arduino
         Overriding Baud Rate          : 115200
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0xdf
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0xdf

avrdude done.  Thank you.

Problem uploading to board.  See https://support.arduino.cc/hc/en-us/sections/
```

En cas d'échec de programmation, revérifiez tous vos câblages, vos connectiques, les paramétrages de l'IDE, et réessayez (ça devrait fonctionner, à moins qu'un composant doit défectueux … !).

Sinon, **si tout a bien fonctionné, vous devriez voir la LED branchée sur votre ATmega328P clignoter « très » rapidement** ; avec l'IDE Arduino qui affiche quelque chose de ce type :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Et cela, dans la fenêtre de log en bas :

```
Le croquis utilise 922 octets (2%) de l'espace de stockage de programmes. Le m
Les variables globales utilisent 9 octets (0%) de mémoire dynamique, ce qui la
C:\Users\JEROME\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-a

avrdude: Version 6.3-20190619
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is "C:\Users\JEROME\AppData\Local\Ardu

         Using Port                    : COM12
         Using Programmer              : arduino
         Overriding Baud Rate          : 115200
         AVR Part                      : ATmega328P
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
parallel program mode         : yes
Timeout                       : 200
StabDelay                     : 100
CmdexeDelay                   : 25
SyncLoops                     : 32
ByteDelay                     : 0
PollIndex                     : 3
PollValue                     : 0x53
Memory Detail                 :

                         Block Poll               Page
  Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  M
  ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- -
  eeprom        65    20     4    0 no       1024    4      0  3600
  flash         65     6   128    0 yes     32768  128    256  4500
  lfuse          0     0     0    0 no          1    0      0  4500
  hfuse          0     0     0    0 no          1    0      0  4500
  efuse          0     0     0    0 no          1    0      0  4500
  lock           0     0     0    0 no          1    0      0  4500
  calibration    0     0     0    0 no          1    0      0     0
  signature      0     0     0    0 no          3    0      0     0

Programmer Type : Arduino
Description     : Arduino
Hardware Version: 3
Firmware Version: 4.4
Vtarget         : 0.3 V
Varef           : 0.3 V
Oscillator      : 28.800 kHz
SCK period      : 3.3 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file "C:\Users\JEROME\AppData\Local\Temp\arduino_build_
avrdude: writing flash (922 bytes):

Writing | ################################################## | 100% 0.38s

avrdude: 922 bytes of flash written
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

```
68
69  Reading | ################################################## | 100% 0.34s
70
71  avrdude: verifying ...
72  avrdude: 922 bytes of flash verified
73
    avrdude done.  Thank you.
```

7. Méthode #4 : programmation d'un ATmega328P en ISP, avec un Arduino Uno transformé en « programmateur »

Comme évoqué en début d'article, **la méthode « ISP » (de l'anglais « In-System Programming ») est l'unique voie de programmation matérielle native et série, proposée par le fabricant de l'ATmega328P** (la société Atmel, qui depuis a été rachetée par Microchip). Et cela ne nécessite aucun bootloader, bien évidemment, car le microcontrôleur a été conçu en ce sens. C'est pourquoi je vais vous détailler cette méthode de programmation plus précisément ici, même si nous avons déjà presque tout vu dans la partie « gravure de bootloader » (car la méthode est la même, même si le but était différent !).

Alors … tout d'abord, il faut savoir que **la programmation ISP se fait directement via les broches du bus SPI de l'ATmega328P, ou indirectement, via un connecteur à 6 broches, nommé « ICSP » (provenant de l'anglais « In-Circuit Serial Programming »)**. Bien sûr, vous êtes libre de passer par un connecteur ICSP ou non, car dans tous les cas, ça revient au même ! En effet, ce seront uniquement les broches SCK, MISO, MOSI, RESET, et GND de votre ATmega328P qui seront raccordées à votre programmateur ISP (qui pourra être une « carte arduino uno modifiée », par exemple, comme nous allons le voir ici).

C'est d'ailleurs cette méthode là que nous avions utilisée pour la programmation d'un bootloader, un peu plus haut (cf. paragraphe §3). Nous allons donc reprendre le même schéma de câblage, même si la partie logicielle sera quelque peu différente 😉

Au final, il y aura donc 2 étapes à suivre ici : **se munir d'un programmateur ISP (ici, nous allons simplement transformer un Arduino Uno en ce sens), et programmer notre ATmega328P avec**. Et pour que les choses restent simples jusqu'au bout, nous utiliserons toujours l'IDE Arduino,

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Si vous n'avez pas de programmateur ISP sous la main, ne vous inquiétez pas ! Car il suffit d'**injecter un simple programme spécifique dans une carte Arduino Uno**, par exemple, pour qu'elle devienne un véritable programmateur de substitution ! Mais pour ce faire, votre carte Arduino Uno devra être RELIÉE À RIEN DU TOUT, mis à part votre PC, via un câble USB.

Ensuite, il vous suffira de suivre les étapes suivantes, pour procéder à la transformation (temporaire) de votre carte Uno :

- brancher votre carte Arduino Uno seule, à votre PC (aucun lien, en dehors)
- ouvrir l'IDE Arduino
- cliquer sur le menu Fichier > Exemples > 11.ArduinoISP > ArduinoISP, afin que **le code du « programmateur ISP » s'ouvre à l'écran**
- cliquer sur le menu Outils > Port > et sélectionner le port COM de l'Arduino Uno que vous venez de connecter à votre PC
- cliquer sur le menu Outils > Type de carte > et sélectionner « Arduino Uno »
- cliquer sur le menu Outils > Programmateur > et vérifier que « AVRISP mkII » soit bien sélectionné
- cliquer sur l'icône « Téléverser » (ou taper CTRL+U au clavier), pour **envoyer le programme ISP dans votre carte Arduino Uno** (ainsi, elle deviendra un véritable « programmateur ISP » !)
- fermer l'IDE Arduino
- et débrancher la carte Arduino Uno de votre PC (car le câblage suivant devra se faire « hors tension »)

Maintenant que **nous avons transformé cet Arduino Uno en « véritable » programmateur d'ATmega328P**, passons à la deuxième étape !

7.2 - Étape 2 : programmer l'ATmega328P au travers de la carte Arduino Uno, transformée en programmateur ISP

À présent, il convient de réaliser le montage suivant, afin de pouvoir programmer notre microcontrôleur :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Pour faire simple, nous avons là un ATmega328P doté d'un oscillateur à quartz 16 Mhz, d'une LED de test branchée sur sa broche 19 (l'équivalent à D13, sur une carte Arduino Uno, si vous préférez), et de quelques **liens vers la carte Arduino Uno, transformée en programmateur ISP pour l'occasion**, comme énuméré ci-après :

- un fil allant de la ligne D10 de l'Arduino Uno (/ss) à la broche 1 (/reset) de l'ATmega328P
- un fil allant de la ligne D11 de l'Arduino Uno (mosi) à la broche 17 (mosi) de l'ATmega328P
- un fil allant de la ligne D12 de l'Arduino Uno (miso) à la broche 18 (miso) de l'ATmega328P
- un fil allant de la ligne D13 de l'Arduino Uno (sck) à la broche 19 (sck) de l'ATmega328P
- un fil allant de la ligne GND de l'Arduino Uno aux broches Gnd de l'ATmega328P
- un fil allant de la ligne +5V de l'Arduino Uno aux broches Vcc de l'ATmega328P (car ici, nous allons alimenter notre ATmega328P et faire sa programmation, via l'Arduino Uno)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Maintenant que nous avons vu la partie matérielle pour cette 2ème étape, voyons ensemble la partie logicielle !

Nota : pour rappel, **comme nous allons directement écrire dans la mémoire flash**

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

ATmega328P avec cette méthode, et même, empiéter sur la zone du bootloader, car il ne sera pas là pour s'en plaindre 😉

La programmation pourra se faire au travers de l'IDE Arduino, comme suit :

- raccorder votre Arduino Uno (et donc tout le montage) à votre PC, via un câble USB
- ouvrir l'IDE Arduino
- ouvrir le programme Blink, accessible via le menu Fichier > Exemples > 01.Basics > Blink (pour faire la démo ; à remplacer ensuite par le programme que vous souhaitez téléverser dans l'ATmega328P)
- cliquer sur le menu Outils > Port > et vérifier que le port COM de l'Arduino Uno est bien le bon
- cliquer sur le menu Outils > Type de carte > et vérifier que « Arduino Uno » est bien sélectionné (remarque technique : **ici, le type de carte « Arduino Uno » ne vise pas la carte Uno, mais bien notre ATmega328P ; en fait, comme ce dernier dispose du même bootloader qu'une carte Uno, on peut faire croire à l'IDE que notre ATMEGA328P est une carte Arduino Uno, raccordée sur notre programmateur ISP**, qui ici, est une véritable carte arduino uno … en espérant ne pas vous avoir perdu en cours de route !)
- cliquer sur le menu Outils > Programmateur > et sélectionner « Arduino as ISP » (attention : il existe beaucoup de noms ressemblants dans la liste proposée, alors ne vous trompez pas !)
- puis cliquer sur Croquis, mais SURTOUT PAS « TELEVERSER », mais cette fois-ci « Téléverser avec un programmateur » (faites bien attention à ne pas vous tromper ici, sinon vous allez cibler votre carte Arduino Uno au lieu de l'ATmega328P !)

Une fois le téléversement terminé, **la LED montée sur breadboard devrait se mettre à clignoter, toutes les secondes environ**. Sinon, vérifiez bien votre câblage, et tout le reste !

Remarque : lorsqu'on fait un « téléversement avec programmateur » comme ici, **le contenu de l'ATmega328P est préalablement effacé ; ainsi, s'il y avait précédemment un bootloader, celui-ci sera « effacé »**. Il vous faudra alors le regraver, si vous souhaitez le remettre en place ; mais on peut parfaitement se passer de bootloader, encore une fois 😉

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

à la configuration de base des « fusibles » d'un ATmega328P, qui sont réglés d'usine pour un fonctionnement à 8 MHz sur oscillateur interne, avec une division par 8 de la fréquence d'horloge (d'où les 1MHz de cadencement programme, de base). Pour résoudre/contourner ce problème, il suffit simplement de graver un bootloader dans votre ATmega328P (cf. paragraphe 3), afin que l'IDE Arduino paramètre les fusibles « correctement ». Puis graver votre programme arduino en utilisant la méthode présentée ici, ce qui effacera le bootloader dans la foulée (mais sans toucher aux fusibles de l'ATmega 328P, ce qui est toute la beauté de la chose !). En clair : vous gravez une fois ce « bootloader jetable » dans votre ATmega328P, afin que le µC tourne à 16MHz sur oscillateur externe, et ensuite, vous pourrez graver autant de programme que vous souhaitez à la place !

IMAGES	COMPOSANTS UTILISÉS	LIEN ACHAT
	– **Microcontrôleur ATmega328P** (format DIP, 28 broches) – **Quartz** 16 Mhz (oscillateur externe) – **Condensateurs céramiques** 22pF – **Résistances** de 1 et 10 kohms (limitation de courant LED et pull-up RESET) – **LED** pour test programme (peu importe la couleur) – **Arduino Uno** R3 (ou carte équivalente)	

8. Programmer son ATmega328P : conclusion !

Voilà ! Je pense que nous avons vu ici toutes **les bases à connaître pour s'initier à la programmation d'un ATmega328P avec une carte Arduino Uno ou un convertisseur USB to**

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Du reste, je vous dit à bientôt, avec un montage qui vous montrera une application pratique de cet article. Alors à bientôt !
Jérôme.

À découvrir aussi : **un écran LCD rétro pour vos projets électroniques (compatible Nokia 5110)**

Ce contenu vous plaît ? Alors abonnez-vous à la Newsletter pour ne rien louper !

📜 Recevoir la Newsletter !

JEROME

Passionné par tout ce qui touche à l'électronique, sans toutefois en être expert, j'ai à coeur de vous partager ici, peu à peu, tout ce que j'ai appris, découvert, réalisé, et testé jusqu'à présent ! En espérant que tout cela puisse vous servir, ainsi qu'au plus grand nombre !

(*) Mis à jour le 17/03/2024

19 commentaires sur “Comment programmer un ATmega328P ou y graver un bootloader, avec un Arduino Uno ou un FTDI ?”

JC LM
2 janvier 2024 à 16:25

Mes meilleurs vœux pour 2024

Je pratique déjà mais un peu de révisions si bien présentées apportent tjrs quelque chose.
Bon courage !!!

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Oui, c'est toujours intéressant de réviser de temps en temps ; car ça permet de voir les choses sous un autre angle, parfois !

Meilleurs voeux à toi aussi 🙂

Bruno
2 janvier 2024 à 21:18

Super article !

Meilleurs vœux pour cette année 2024 🙂

Jérôme
2 janvier 2024 à 22:09

Merci, à toi aussi !!

Racine Michel
3 janvier 2024 à 08:10

Meilleurs voeux pour 2024 ! Très bon article très clair et bien présenté ! Merci.

Eric
7 janvier 2024 à 23:59

Bonjour,

Rien à redire : c'est parfait et complet !

Bravo pour le travail accompli.

Jérôme
8 janvier 2024 à 07:48

Merci beaucoup, ça fait plaisir !

Maurice Sauron
13 janvier 2024 à 16:26

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Merci pour ton travail.

Jérôme
13 janvier 2024 à 19:17

Merci, et au plaisir de te revoir, alors !

Très bonne année à toi également 😀
Jérôme.

Henri Jayles
18 janvier 2024 à 19:59

Je viens de découvrir vos tutos et bien c'est super pro

Bravo et merci
Un débutant

Jérôme
19 janvier 2024 à 07:35

Héhé, merci !

À bientôt, alors !!
Jérôme.

Didier
21 février 2024 à 18:26

Bravo,

Super tuto !

Un grand MERCI

Gabriel
22 février 2024 à 12:23

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

pédagogie du tonnerre et pour cela et pour pour votre tuto je vous adresse un grand merci. J'ai hâte de voir (lire) la suite ! Encore merci !!!

Jérôme
22 février 2024 à 13:38

Merci, c'est gentil !!!

Mais attention, je ne suis pas prof ! Je suis seulement un passionné d'électronique, qui partage autant que possible 🙂

Sur ce à bientôt, et encore merci à toi !
Jérôme.

Dominique
26 février 2024 à 15:53

Un grand merci Jérôme pour votre tutoriel.

J'ai enfin réussi à programmer mon premier ATmega328P !

Jérôme
26 février 2024 à 18:32

Héhé, de rien !

Au plaisir alors 🙂
Jérôme.

Gaelle
5 juin 2024 à 10:04

Merci pour ce tuto très complet !

René
1 décembre 2024 à 20:13

Bravo, votre site est excellent et vos explications sont très claires. Merci !!

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Laisser un commentaire

Votre adresse e-mail ne sera pas publiée. Les champs obligatoires sont indiqués avec *

Commentaire *

Prénom/NOM *

E-mail *

Enregistrer mon nom, mon e-mail et mon site dans le navigateur pour mon prochain commentaire.

Résolvez cette soustraction : 59 - = 55

Laisser un commentaire

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité

Afin de filtrer au maximum les messages de type "spam" ou "inappropriés", **chaque commentaire est soumis à modération, et validé manuellement**. Du coup, il se peut que certains commentaires ne soient pas publiés, ou sinon, avec un peu de retard. Par ailleurs, j'ai malheureusement plus de messages à traiter que de temps pour y répondre ; c'est pourquoi je ne pourrais pas répondre à tout le monde. Désolé …

Étiquettes: ARDUINO ATMEGA328P DÉCOUVERTE

© Copyright 2020-2026, **PassionElectronique.fr**
Tous droits réservés.

Certains liens présents sur ce site sont affiliés. Cela permet, même si ça reste très limité, d'aider au financement de ce site. Sachez que ceux-ci ne vous coûtent rien, et que vous restez totalement libre de les utiliser ou pas. Du reste, merci à vous tous, notamment pour vos nombreux messages chaleureux et bienveillants ! Encore merci !

Raccourcis : **Tous les articles**, **Mentions légales**, **Politique de confidentialité**, et **Page contact**

Réseaux : **Reddit**, **Projets partagés (PCBWay)**, et **GitHub**

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre politique de confidentialité