Laboratoire 5 : SPI (interface périphérique série)

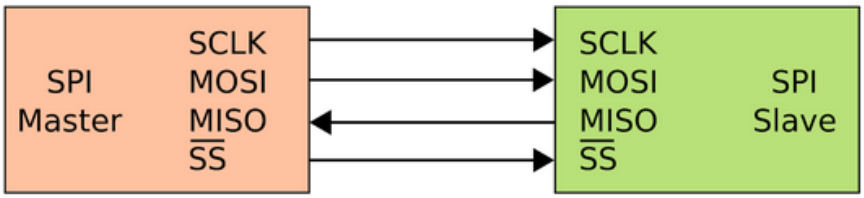
Ce laboratoire couvre le concept de SPI. Pour plus d'informations sur ce sujet, consultez la fiche technique de l'ATmega328P [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf] et de l'interface périphérique série .

1. SPI (interface périphérique série)

SPI est une norme synchrone développée par Motorola qui fonctionne en mode duplex intégral (le transfert de données se produit dans les deux sens simultanément). Les appareils communiquent selon une architecture Maître-Esclave (un seul appareil Maître et un ou plusieurs appareils Esclave sont autorisés). L'appareil maître est celui qui initie la communication. SPI est également appelé "quatre fils". Les quatre signaux utilisés sont les suivants :

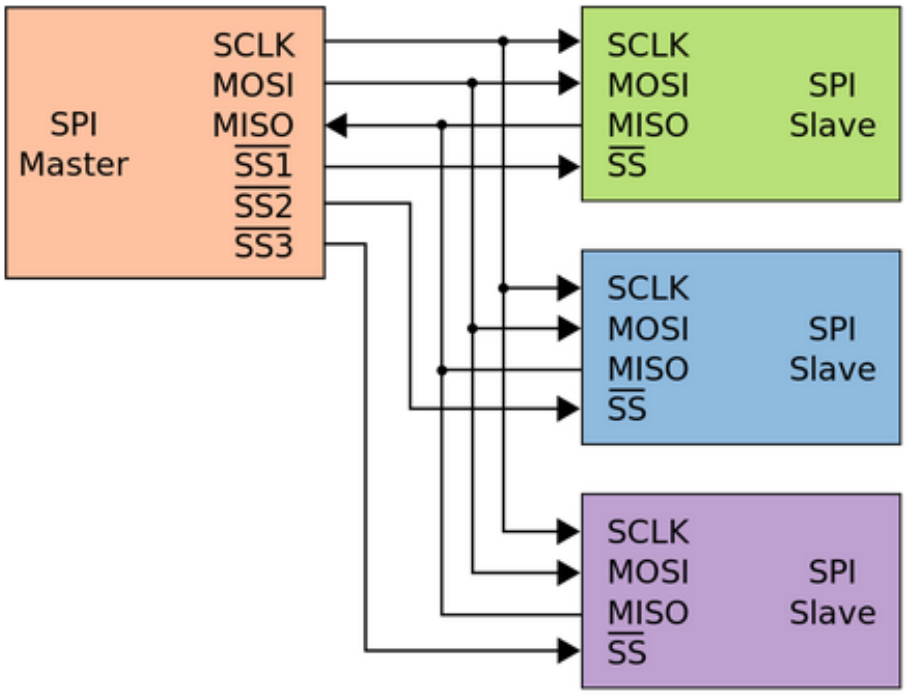
- MOSI — Master Output Slave Input (transmission des données du maître à l'esclave)
- MISO - Master Input Slave Output (transmission de données d'esclave à maître)
- SCLK - Serial Clock (synchronisation entre les appareils. Contrôlé par le maître)
- CS/SS — Chip Select/Slave Select (Sélection de l'appareil esclave par le maître. Valeur BASSE pour l'esclave sélectionné)

Nous pouvons voir qu'il y a deux signaux utilisés pour transmettre des données : MOSI et MISO. La transmission de données sur SPI se fait de manière synchrone à l'aide du signal d'horloge SCLK. Le signal CS / SS est utilisé pour sélectionner l'équipement esclave sur le bus qui est actuellement adressé.



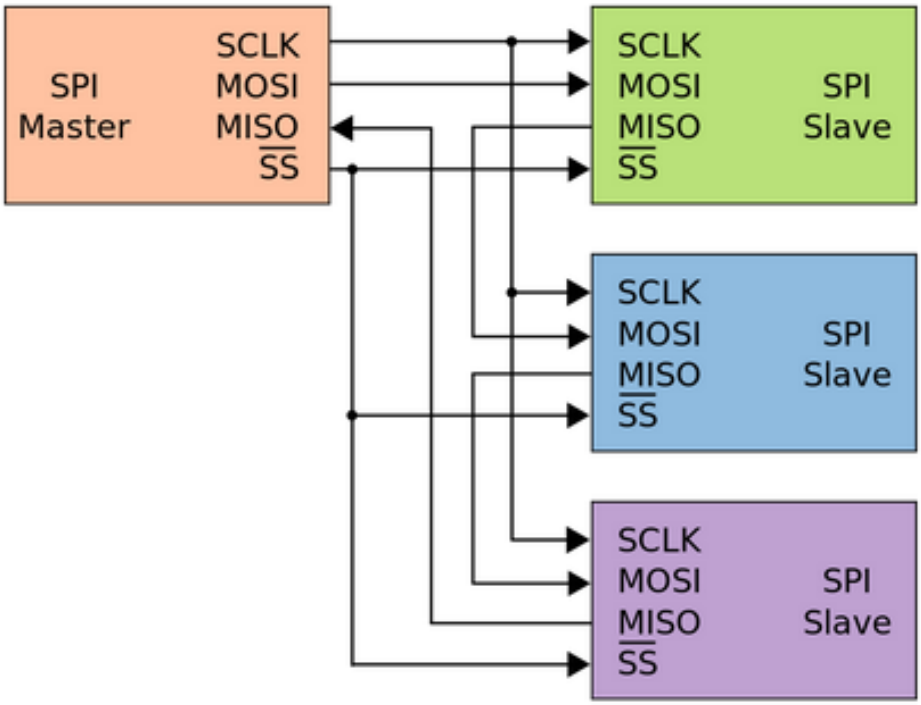
2. Connexion de plusieurs appareils esclaves

Plusieurs appareils esclaves peuvent être connectés à un seul appareil maître en même temps. Les signaux MOSI, MISO et SCLK sont partagés. Pour chaque appareil esclave, l'appareil maître a un signal CS / SS. Lorsque le maître veut communiquer avec un esclave, l'appareil maître met à BAS le signal CS / SS menant à l'appareil esclave souhaité (les autres signaux CS / SS sont mis à HAUT). Ainsi, l'appareil esclave avec CS / SS sur LOW sait que le maître communique avec lui.



3. Connexion de plusieurs appareils esclaves à l'aide de la topologie SPI Daisy Chain

Dans la topologie Daisy Chain, les données sont transférées du Maître au premier Esclave, du premier Esclave au second Esclave, jusqu'à ce que les données atteignent le dernier Esclave qui les renvoie au Maître. Les données du premier esclave parviennent au maître en dernier, tandis que les données du dernier esclave parviennent au maître en premier.



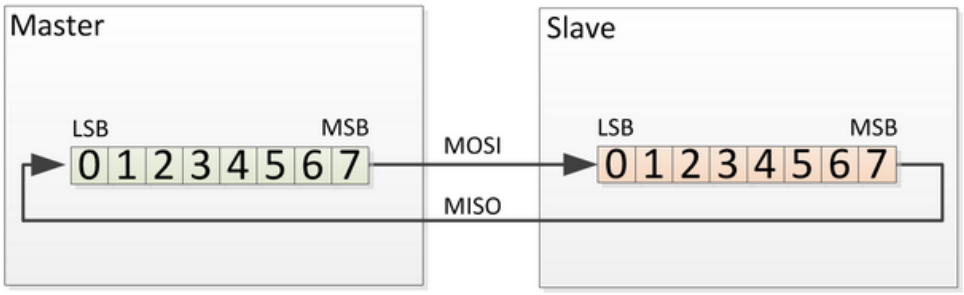
4. Transmission de données avec SPI

Pour démarrer la communication avec l'appareil Maître, l'horloge du Maître doit être réglée sur une fréquence au plus égale à la fréquence supportée par l'appareil Esclave (typiquement jusqu'à quelques

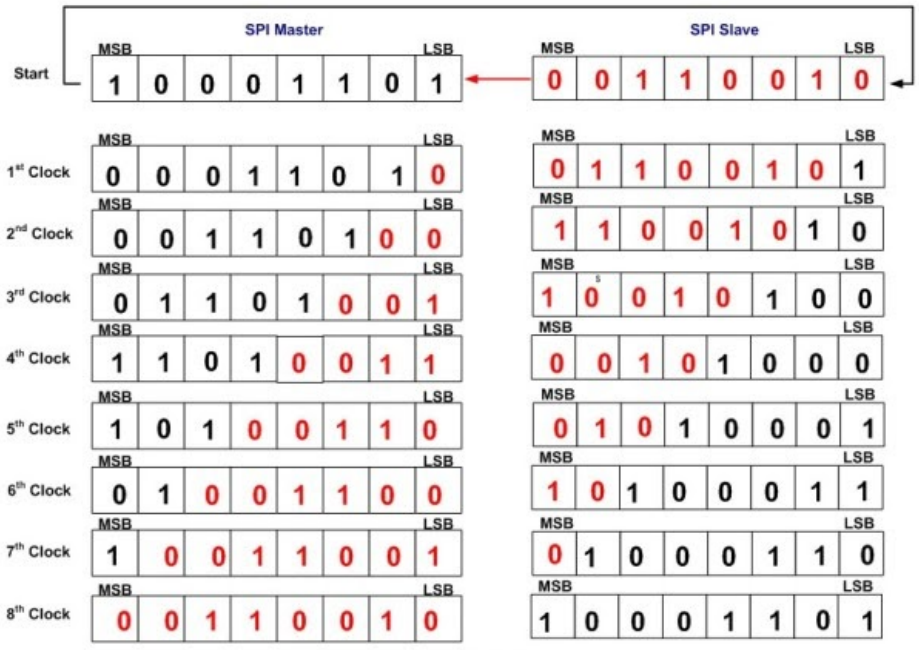
MHz) . Le Maître sélectionne alors l'appareil Esclave souhaité en mettant 0 sur la ligne CS/SS. Lors d'un cycle SPI, la transmission est full-duplex :

- L'appareil Maître envoie un bit sur la ligne MOSI et l'appareil Esclave lit ce bit.
- L'appareil Esclave envoie un bit sur la ligne MISO et l'appareil Maître lit ce bit.

En règle générale, la communication SPI implique deux registres à décalage (un en maître et un en esclave, connectés de manière circulaire).



Habituellement, le premier bit déplacé sur MISO / MOSI est le bit le plus significatif, tandis qu'un nouveau bit est ajouté à la position la moins significative du registre. Une fois que le mot entier a été envoyé, par décalage, le maître et l'esclave ont complètement changé les valeurs dans les deux registres à décalage. S'il y a plus de données à transmettre, le processus recommence. Lorsqu'il n'y a pas de données à transmettre, l'appareil Maître arrête de générer le signal d'horloge (SCLK) et place le signal CS/SS associé à l'Esclave à HIGH (logique 1). Les appareils esclaves qui n'ont pas été sélectionnés par le signal SS associé ignoreront les signaux sur SCLK et MOSI et ne généreront rien sur MISO. Le Maître ne peut sélectionner qu'un seul Esclave à la fois.



SPI Master and Slave Data Transfer Diagram

5. Configurations SPI

Clock Polarity (CPOL) configure l'état IDLE de l'horloge. Comme nous pouvons le voir, pour CPOL = 0, lorsque l'horloge est inactive, la valeur du signal SCLK est sur LOW, et pour CPOL = 1, l'horloge inactive a SCLK sur HIGH.

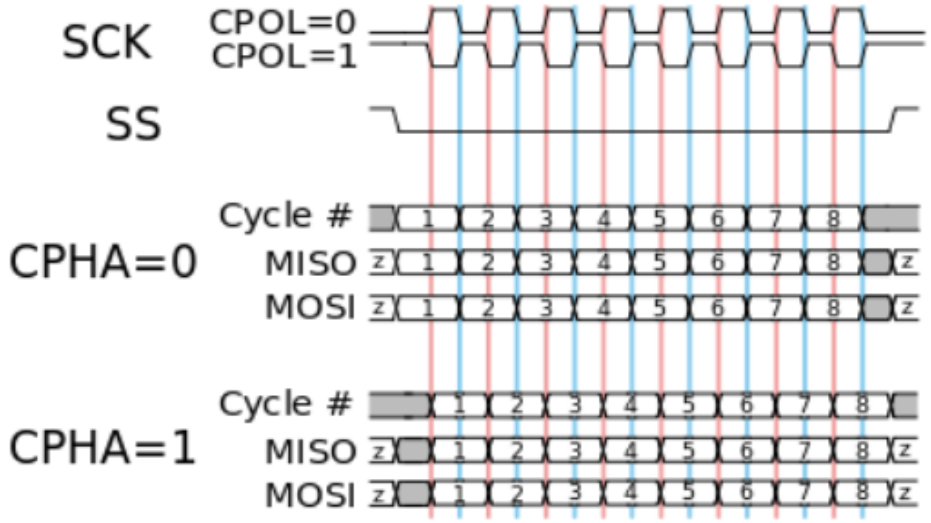
Clock Phase (CPHA) configure quand les données sont générées en sortie et quand les données sont lues en entrée. Sur l'un des fronts d'horloge (montant ou descendant) les données sont générées et sur l'autre les données sont lues.

Le cadran de départ de l'horloge est le premier changement d'horloge. Ex : Pour une horloge allant de LOW à HIGH et retour à LOW, le front montant est le front montant (la transition de LOW à HIGH).

Pour CPHA = 0, les données sont générées avant le front montant (premier front d'horloge) et lues sur le front montant (front montant pour CPOL = 0 et front descendant pour CPOL = 1).

Pour CPHA = 1, les données sont générées sur le front montant (premier front d'horloge) et lues sur le front descendant (front descendant pour CPOL = 0 et front montant pour CPOL = 1).

CPOL et CPHA doivent être définis en fonction de la configuration de l'appareil SPI avec lequel nous communiquons. Par exemple, si notre UC communique avec un ADC qui utilise CPOL = 1 et CPHA = 1, il est obligatoire que notre appareil ait les mêmes configurations SPI.



6. SPI dans Atmega328p

Le SPI inclus dans le microcontrôleur atmega328p peut fonctionner à la fois en tant que maître et esclave.

Registre de contrôle SPI (SPCR)

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 6 - SPE (activation SPI)
  - 0 - SPI désactivé
  - 1 - SPI activé
- Bit 5 - DORD (ordre des données)
  - 0 - MSB (de la date) envoyé en premier
  - 1 - LSB (de la date) envoyé en premier
- Bit 4 - MSTR (sélection maître/esclave)
  - 0 - Mode esclave
  - 1 - Mode maître
- Bit 3 - CPOL (polarité d'horloge)
  - 0 - SCK LOW au repos (lorsqu'il n'y a pas de communication)
  - 1 - SCK HIGH au repos (lorsqu'il n'y a pas de communication)
- Bit 2 - CPHA (phase d'horloge)
  - 0 - les données sont lues sur le front d'accueil
  - 1 - les données sont lues sur le front descendant

- Bit 1:0 - SPR1:SPR0 (fréquence d'horloge SPI)
  - en mode Master : contrôle la fréquence du SCK.
  - En mode Esclave : ils n'ont aucun effet.

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Registre d'état SPI (SPSR)

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 - SPIF (drapeau d'interruption SPI)
  - est automatiquement réglé sur 1 lorsque la transmission est terminée.
- Bit 0 - SPI2X (bit de vitesse SPI double)
  - En mode Master : Lorsque ce bit est mis à 1, la vitesse SPI (SCK Frequency) sera doublée. Cela signifie que la période SCK minimale sera de deux périodes d'horloge CPU.
  - en mode Esclave : SPI est uniquement garanti de fonctionner à fosc / 4 ou inférieur.

Registre de données SPI (SPDR)

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Un registre de lecture/écriture utilisé pour le transfert de données entre le fichier de registre et le registre à décalage SPI. L'écriture dans le registre lance la transmission des données. La lecture du registre entraîne la lecture du tampon de réception du registre à décalage.

Exemple

Initialisez SPI en tant que maître.

```
/* définir le mode maître */
SPCR0 |= (1 << MSTR0);

/* définit le prédiviseur 16 */
SPCR0 |= (1 << SPR00);

/* activer SPI */
SPCR0 |= (1 << SPE0);
```

Transmission d'un octet (sur le maître).

```
/* Démarrer la transmission */
SPDR0 = données ;
```

```
/* Attendre la fin de la transmission */
tandis que(!(SPSR0 & (1 << SPIF0)));
```

Réception d'un octet (sur Esclave).

```
/* Attendre la fin de la réception */
tandis que(!(SPSR0 & (1 << SPIF0)));

/* Renvoyer le registre de données */
renvoie SPDR0 ;
```

7. SPI dans l'IDE Arduino

L'environnement de développement Arduino IDE nous offre la possibilité d'utiliser la bibliothèque "SPI.h" pour faciliter la communication via SPI. Ci-dessous, vous pouvez voir un exemple de code pour transmettre un octet de données :

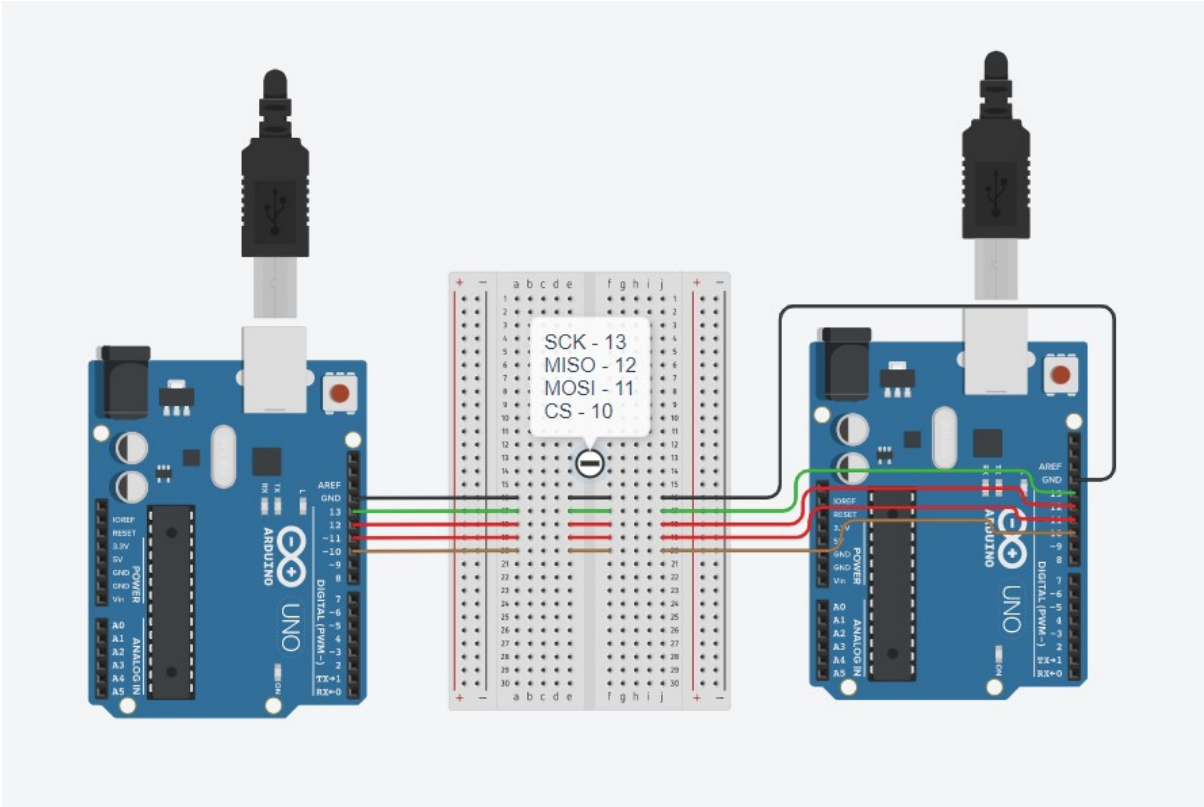
- SPI.h

```
/* Exemple SPI MASTER : transfert d'un octet vers l'esclave */
#include "SPI.h"

void setup ( void )
{
  byte masterSend =  1 ;
  byte masterRecv ;

  /* Initialiser la broche de sélection de l'esclave */
  pinMode ( SS , OUTPUT ) ;
  digitalWrite ( SS , HIGH ) ;

  /* Initialise le SPI */
  SPI. begin( ) ;

  /* Sélectionne l'Esclave */
  digitalWrite ( SS , LOW ) ;

  /* Envoie l'octet masterSend à l'esclave sélectionné.
   * Enregistrement de l'octet reçu de l'esclave sélectionné dans masterRecv. */
  masterRecv = SPI. transfert ( masterSend ) ;

  /* Désélectionner l'Esclave */
  digitalWrite ( SS , HIGH ) ;
}
```

8. Exercices

1. Établissez une communication SPI entre 2 cartes Arduino.
    - Décidez quelle tuile sera maître et laquelle sera esclave (une équipe peut choisir de coder pour maître et l'autre pour esclave).
    - Envoyez un message du Maître à l'Esclave ainsi que de l'Esclave au Maître et affichez-les sur l'interface série. Définissez une longueur minimale d'informations à envoyer avant d'arrêter la transmission des données.
    - Pour pouvoir détecter la communication SPI depuis l'esclave, vous pouvez implémenter l'interruption SPI_STC_vect (vous devez ajouter dans la configuration "SPI.attachInterrupt();" - comment les variables modifiées devaient-elles être déclarées dans les interruptions ?). Après avoir téléchargé le code sur les pads, réinitialisez-les tous les deux (en appuyant sur le bouton rouge dessus), d'abord le pad esclave (qui attendra

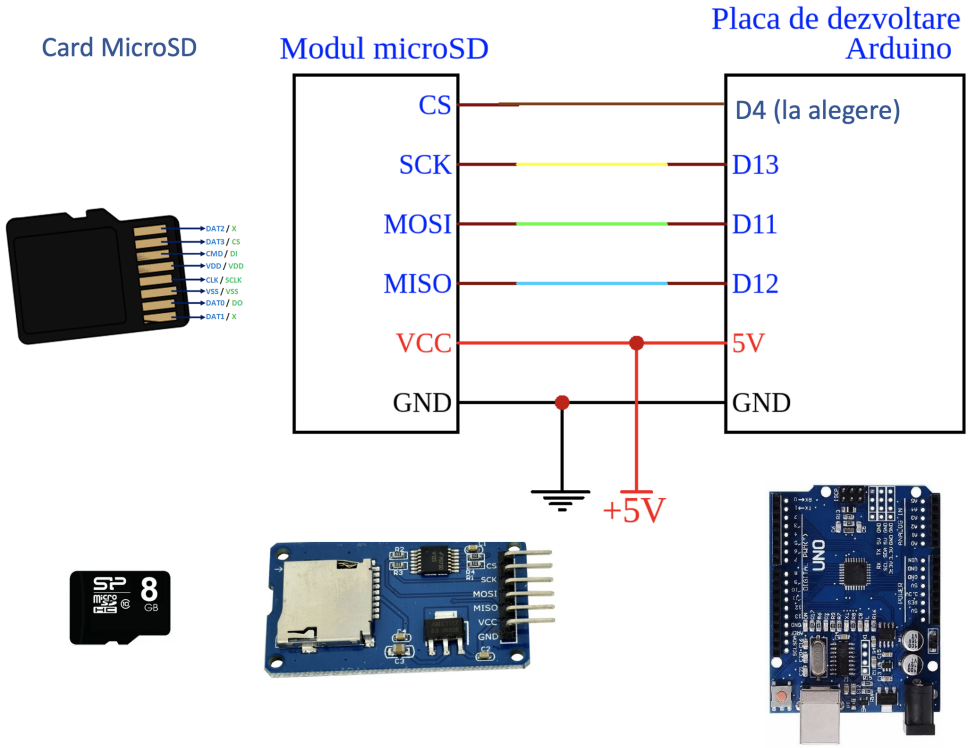
alors que la communication s'initie) puis le pad maître (qui initiera la communication).



1. **(Bonus)** Une carte MicroSD est un exemple de mémoire non volatile. Pour que l'Arduino UNO communique via SPI avec la carte, nous avons besoin d'un adaptateur. Concernant le code, en plus de la bibliothèque "SPI.h" présentée ci-dessus, nous avons également besoin de la bibliothèque "SD.h", que nous vous conseillons de lire avant de poursuivre les exercices.
    - a) Écrivez un secret dans un fichier nommé "sd.txt" sur la carte SD.
    - b) Échangez des cartes avec vos collègues et lisez leur secret.
    - c) Faire un compteur des resets de la carte Arduino en écrivant sur la carte SD dans le fichier "re.txt" le nombre de resets.

Astuce1 : Lors de la réinitialisation, le code de configuration est exécuté à nouveau.
Astuce2 : Enregistrez sur la carte SD la dernière valeur du nombre de réinitialisations.

Si vous avez modifié le montage pour faire l'exercice bonus, refaites la mise en page de l'exercice 1 avant de partir !

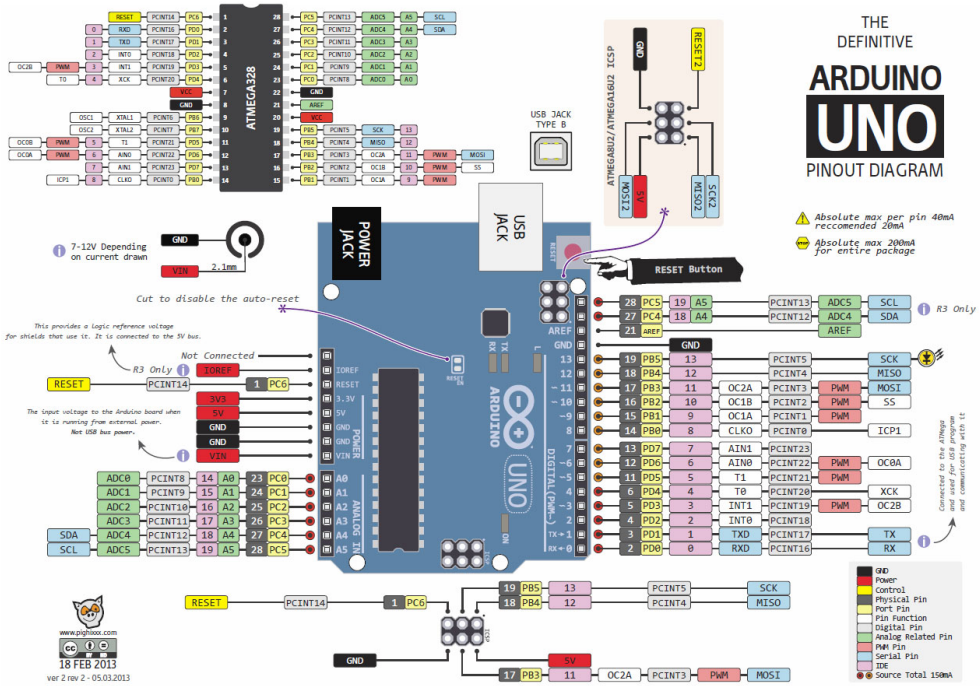


- SD.h

```
/* Exemple MASTER : Créer et écrire un fichier sur la carte MicroSD */
#include "SD.h"

void setup ( void )
{
  File monFichier ;

  /* Initialise la communication avec la carte MicroSD. Le signal SS est sur la broche 4 */
  if  ( SD. begin ( 4 ) )
  {
    /* Créer le fichier s'il n'existe pas */
    if  ( ! SD. exists ( "hello.txt" ) )
    {
      myFile = SD. ouvrir ( "hello.txt" , FILE_WRITE ) ;
      if  ( monFichier )
      {
        monFichier. print ( "Bonjour le monde" ) ;
        mon fichier. fermé( ) ;
      }
      else
      {
        /* Le fichier n'a pas pu être créé. La question doit être étudiée. */
      }
    }
  }
  else
  {
    /* L'initialisation de la communication a échoué. La question doit être étudiée. */
  }
}
```

9. Ressources

- Fiche technique Atmega 328p
- Brochage Arduino UNO



- Responsable : Bogdan-Călin Ciobanu [mailto:bogdan.ciobanu1201@stud.acs.upb.ro] Ionuț-Gabriel Oțelea [mailto:ionut.otelea@upb.ro]

RÉSOUDRE

10. Liens utiles

- Arduino SPI [https://www.arduino.cc/en/reference/SPI]
- SD Arduino [https://www.arduino.cc/en/reference/SD]
- Fiche technique ATmega328p [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf]