TP N° 3 : COMMUNICATION SPI ENTRE DEUX CARTES ARDUINO

I. Objectifs

Arduino SPI (Serial Peripheral Interface) se traduit par « interface périphérique série » et est un protocole de communication série. Dans cet article, nous allons examiner cette interface et l'utiliser dans Arduino (comment connecter dispositifs SPI avec Arduino).

II. Matériels utilisés

Pour cette activité, nous aurons besoin:

- ✓ Arduino Uno;
- ✓ un breadboard;
- ✓ une résistance;
- ✓ les fils de connexion;
- ✓ la librairie SPI.h.

III. Schéma et principe

La liaison SPI Arduino est l'une des principales interfaces de communication entre l'Arduino et les appareils connectés. Les protocoles SPI, I2C et UART sont très couramment utilisés pour connecter des périphériques.
L'interface série Arduino SPI (Interface périphérique série) est entièrement duplexée – les données sont transmises et reçues simultanément sur les fils. Le maître envoie des données à l'esclave et l'esclave peut simultanément envoyer des données au maître.

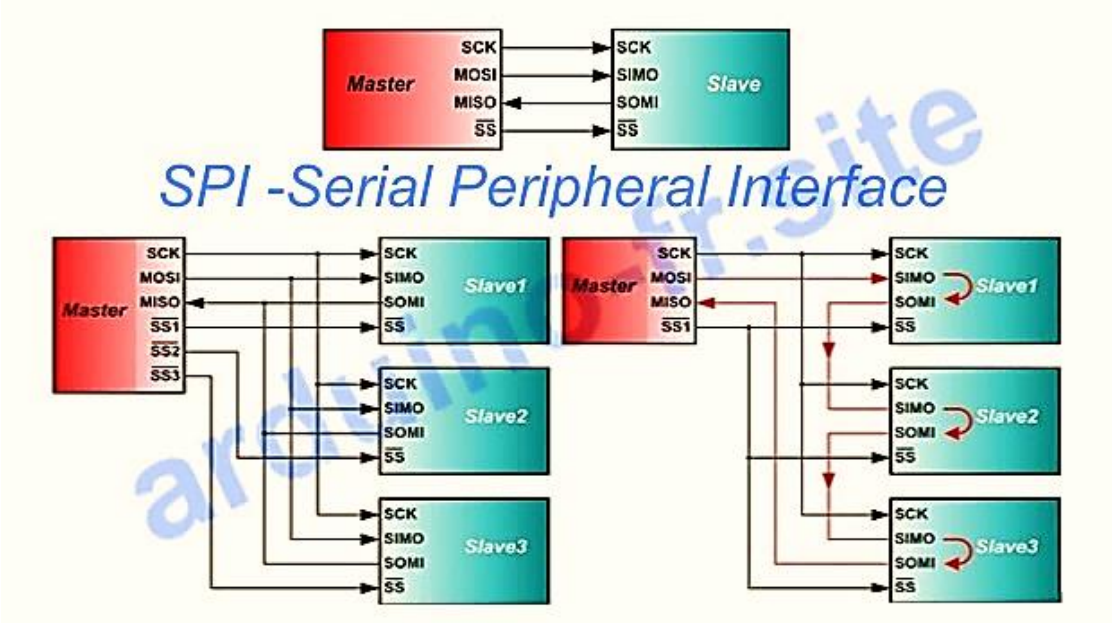

Figure 1 : Interface périphérique série – description, vue d'ensemble

L'Arduino utilise 4 lignes (fils) pour travailler avec le protocole SPI. Dans cette interface, il y a toujours un dispositif maître (généralement la carte Arduino) et plusieurs esclaves (capteurs, écrans ou autres microcontrôleurs peuvent être des dispositifs esclaves).

La liaison SPI utilise 4 lignes – MISO, MOSI, SCLK et SS :

- ✓ **MISO** (Master in Slave Out) est la ligne pour envoyer des données au maître;
- ✓ **MOSI** (Master Out Slave In) est la ligne pour envoyer des données à l'esclave;
- ✓ **CLK** (Serial Clock) est une ligne de synchronisation des appareils générée par le maître;
- ✓ **SS** (Slave Select) – ligne permettant d'activer/désactiver les dispositifs esclaves.

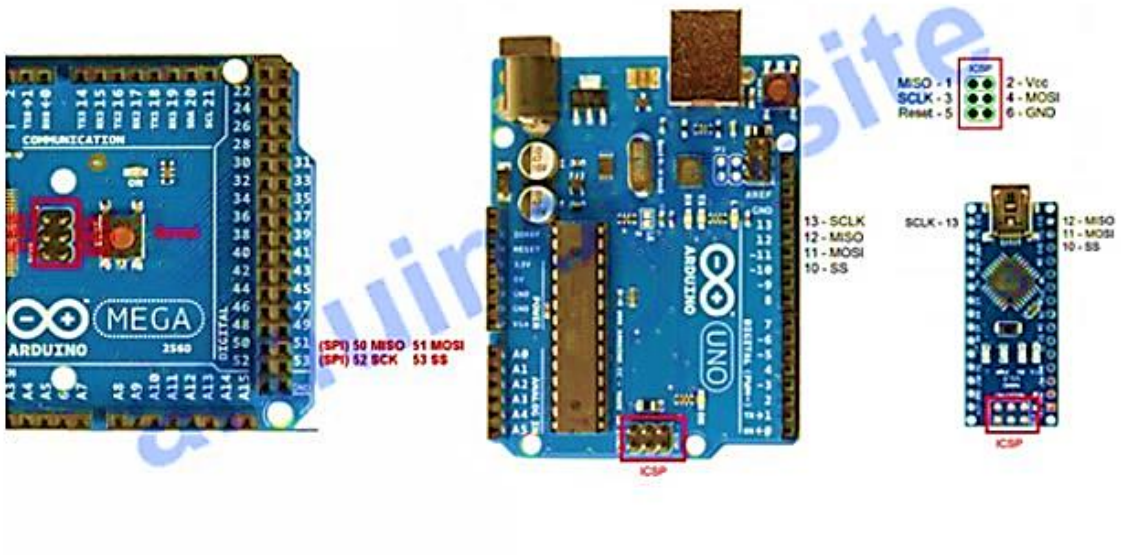

Figure 2 : SPI pins sur les cartes Arduino Uno, Nano et Mega

pins	Arduino Uno	Arduino Nano	Arduino Mega
MISO	12	12	50
MOSI	11	11	51
SCK	13	13	52
SC	10	10	53

Pour initier la communication, la broche de l'esclave (SS) doit être réglée à un niveau bas, si la broche est réglée à un niveau haut, l'esclave ignorera le maître. Cela permet de connecter de nombreux dispositifs SPI esclaves au microcontrôleur.

IV. Bibliothèque SPI (Serial Peripheral Interface)

#include « SPI.h » – connecte l'utilisation de la bibliothèque dans le programme.
SPI.begin() – initialise le bus SPI en définissant les modes SCK, MOSI et SS pour la sortie de données, en alimentant les broches SCK et MOSI avec une tension basse (low), et la broche SS avec une tension haute (high).

SPI.setClockDivider(divider) – définit le facteur de division du signal d'horloge SPI par rapport à la fréquence d'horloge du microcontrôleur Arduino. Vous pouvez utiliser des diviseurs de 2, 4, 8, 16, 32, 64 ou 128.
SPI.attachInterrupt(handler) – active le mode d'interruption dans le dispositif esclave. Une interruption se produit chaque fois que le dispositif esclave reçoit des données du dispositif maître.
SPI.transfer(val) – Cette fonction est utilisée pour recevoir et transférer des données entre le maître et l'esclave simultanément.

V. MANIPULATION

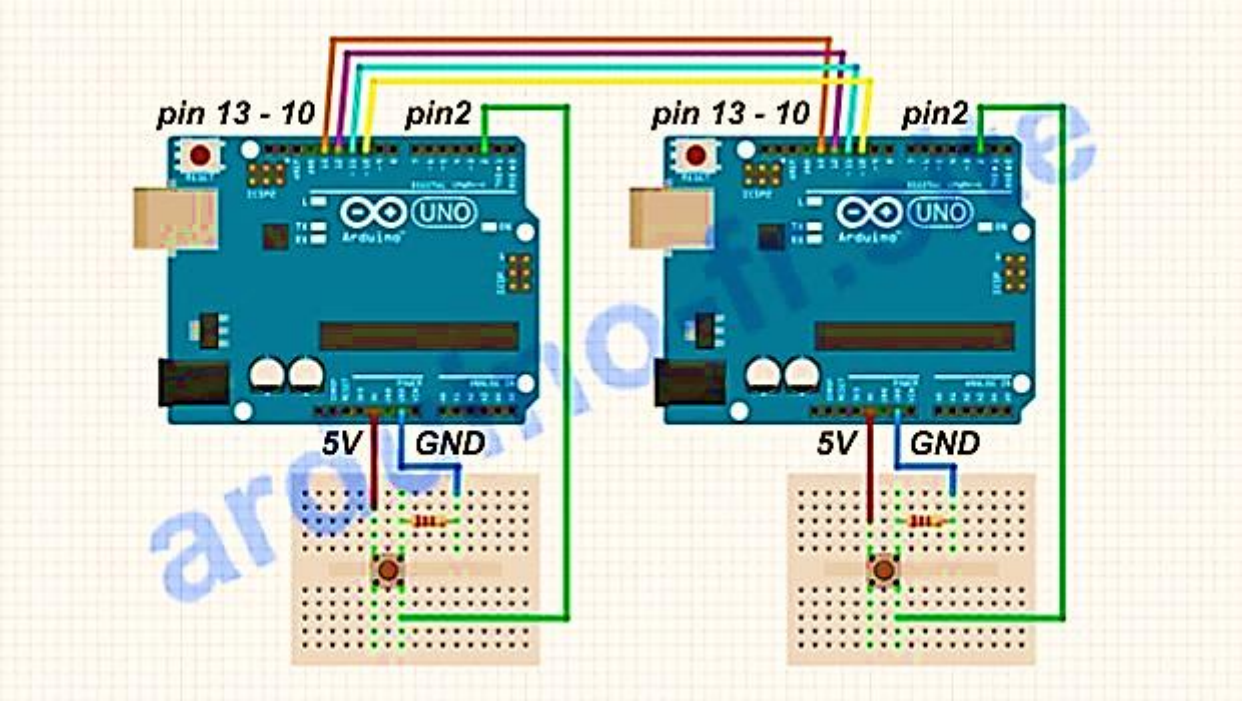

Figure 3 : Connexion de deux cartes Arduino via la liaison SPI

Le schéma de câblage des deux cartes Arduino pour créer la connexion entre elles est présenté dans la figure 3 : Lorsque vous appuyez sur le bouton de la carte maître, la LED connectée à la carte esclave s'allume, lorsque le bouton de l'esclave Arduino Uno est pressé, la LED de l'Arduino maître est allumée.

Travail demandé :

1. Compléter le Programme d'Arduino Esclave et d'Arduino Maître.
2. Brancher les cartes Arduino (figure 3) puis compiler les programmes et les téléverser.

Programme pour la carte esclave Arduino (slave)

```
#include "SPI.h"
#define LEDpin 13
#define buttonpin 2
volatile boolean received;
volatile byte Slavereceived,Slavesend;
int buttonvalue;
int x;
```

```
void setup() {
  pinMode(----------, INPUT);
  pinMode(---------, OUTPUT);
  pinMode(MISO, OUTPUT);
  SPCR |= _BV(SPE);
  received = false;
  SPI.attachInterrupt();
}

ISR (SPI_STC_vect) {
  Slavereceived = SPDR;
  received = true;
}

void loop() {
  if(received) {
    if (Slavereceived==1) { digitalWrite(LEDpin,--------); }
    else { digitalWrite(LEDpin, -------); }

    buttonvalue = digitalRead(---------------);
    if (buttonvalue ==-------------) { x=1; }
    else { x=0; }

    Slavesend=x;
    SPDR = Slavesend;
    delay(100);
  }
}
```

Programme pour la carte maîtresse Arduino (master)

```
#include "SPI.h"
#define LED 13
#define ipbutton 2
int buttonvalue;
int x;
void setup () {
  pinMode(ipbutton, --------);
  pinMode(LED,----------);

  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV8);
  digitalWrite(SS, HIGH);
}
```

```
void loop() {
  byte Mastersend, Mastereceive;
  buttonvalue = digitalRead(----------);
  if(buttonvalue == ----------) { x = 1; }
  else { x = 0; }

  digitalWrite(SS, LOW);
  Mastersend = x;
  Mastereceive = SPI.transfer(Mastersend);

  if(Mastereceive == 1) { digitalWrite(LED, --------); }
  else { digitalWrite(LED, ----------); }
  delay(100);
}
```