TP : Communication USART Bas Niveau entre Deux Arduino (Programmation Registre)

Kennouche Abderrahmane

1 Introduction

Dans ce TP, nous avons réalisé une communication série **USART bas niveau** entre deux cartes Arduino UNO en utilisant exclusivement les **registres du microcontrôleur ATmega328P**.

L'Arduino **Master** lit un clavier matriciel (4x3) et envoie la touche pressée via USART. L'Arduino **Slave** reçoit cette valeur et l'affiche sur un afficheur **7 segments** en pilotant directement les ports du microcontrôleur.

2 Arduino Master : Lecture du Keypad et Envoi USART

2.1 Configuration du Keypad

Les lignes du clavier (R1–R4) sont connectées aux broches **PD2 à PD5**. Les colonnes (C1–C3) sont connectées aux broches **PB1 à PB3**.

```
char tab[4][3] = {
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'*','0','#'}
};

int rows[4] = {(1<<PD2),(1<<PD3),(1<<PD4),(1<<PD5)};
int cols[3] = {(1<<PB1),(1<<PB2),(1<<PB3)};
```

2.2 Initialisation de l'USART (Master)

La fonction suivante configure l'USART en émission, à **9600 bauds**, en utilisant la fréquence du CPU de 16 MHz.

```
void usart_init() {
    UBRR0L = 103;         // Baudrate = 9600
    UCSR0B = (1 << TXEN0);           // Activer transmission
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);   // 8 bits de donn es
}
```

La fonction d'envoi bloque jusqu'à ce que le registre d'envoi soit vide :

```
void usart_send(unsigned char data) {
    while (!(UCSR0A & (1 << UDRE0))); // Attendre buffer vide
    UDR0 = data;                      // Envoyer octet
}
```

2.3 Initialisation des Ports du Keypad

Les lignes sont en sortie (HIGH par défaut). Les colonnes sont en entrée avec résistances de pull-up.

```
void setup() {
    DDRD |= 0b00111100;     // PD2-PD5 en sortie
    PORTD |= 0b00111100;    // Lignes HIGH

    DDRB &= ~0b00001110;    // PB1-PB3 en entr e
    PORTB |= 0b00001110;    // Pull-up activ s

    usart_init();
}
```

2.4 Lecture du Keypad et Envoi USART

Pour chaque ligne, on la force à LOW, on lit les colonnes et on détecte la touche pressée.

```
void loop() {
    for (int r = 0; r < 4; r++) {
        PORTD &= ~rows[r];

        for (int c = 0; c < 3; c++) {
            if ((PINB & cols[c]) == 0) {
                usart_send(tab[r][c]);
                delay(200);
            }
        }
        PORTD |= rows[r];
    }
}
```

3 Arduino Slave : Réception USART et Affichage 7 Segments

3.1 Table des segments

Chaque valeur représente l'état des segments (a–g). Le segment **g** est connecté à **PB1**, séparé du port D.

```
unsigned char digits[10] = {
    0b00111111, // 0
    0b00000110, // 1
    0b01011011, // 2
    0b01001111, // 3
    0b01100110, // 4
    0b01101101, // 5
    0b01111101, // 6
    0b00000111, // 7
    0b01111111, // 8
    0b01101111  // 9
};
```

3.2 Initialisation de l'USART (Slave)

```
void usart_init() {
    UBRR0L = 103;
    UCSR0B = (1 << RXEN0); // Activer r ception
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}
```

3.3 Réception

```
unsigned char usart_receive() {
    while (!(UCSR0A & (1 << RXC0)));
    return UDR0;
}
```

3.4 Initialisation des Ports d'Affichage

Segments a–f + dp = PD2 à PD7 Segment g = PB1

```
void setup() {
    DDRD |= 0b11111100;  // Segments a-f en sortie
    PORTD &= ~0b11111100;

    DDRB |= (1 << PB1);  // Segment g en sortie
    PORTB &= ~(1 << PB1);

    usart_init();
}
```

3.5 Affichage du chiffre reçu

```
void loop() {
    unsigned char data = usart_receive();

    if (data >= '0' && data <= '9') {

        unsigned char val = digits[data - '0'];

        PORTD = (PORTD & 0b00000011) | (val << 2);

        if (val & 0b01000000)
            PORTB |= (1 << PB1);   // Segment g ON
        else
            PORTB &= ~(1 << PB1);  // Segment g OFF
    }
}
```

FIGURE 1 – Enter Caption

Montage

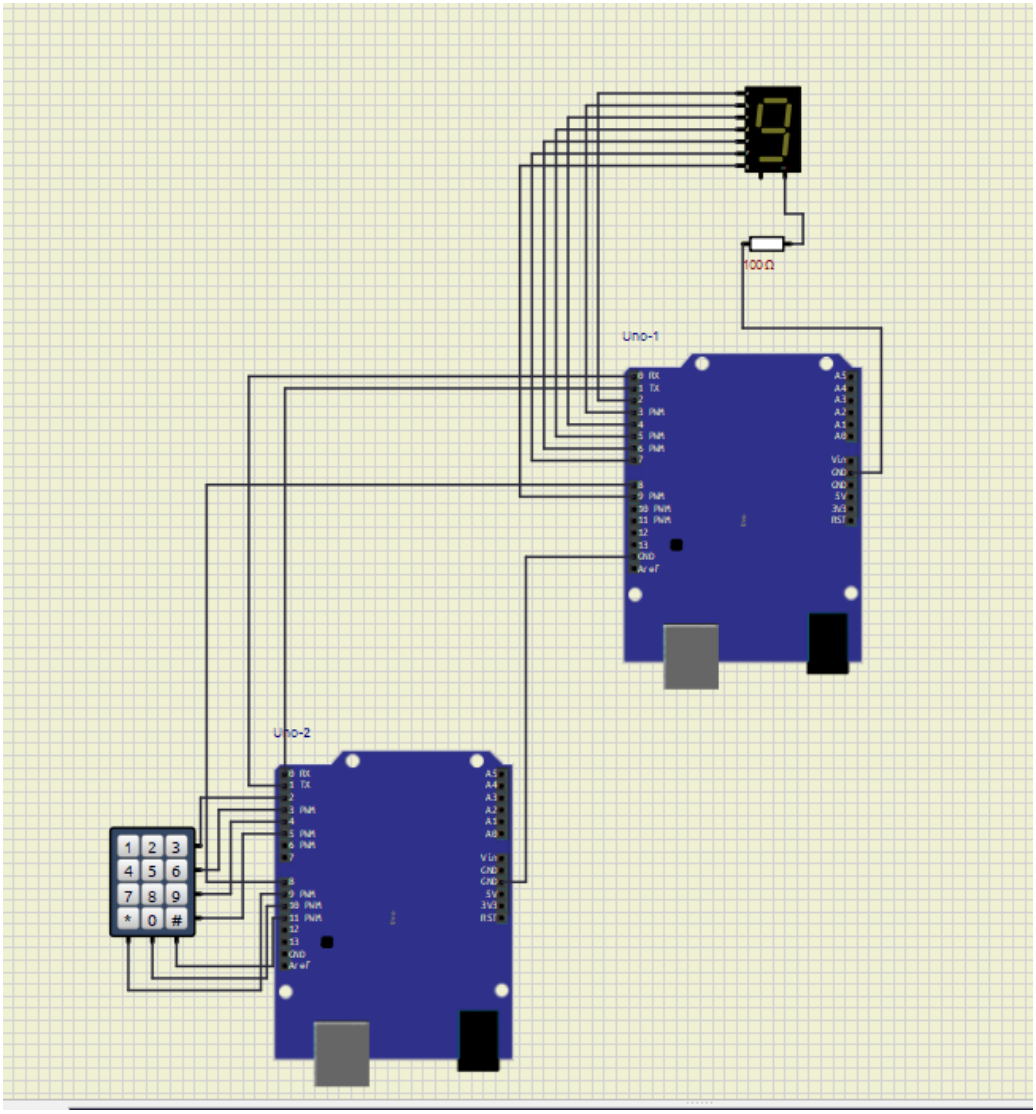

FIGURE 2 – Montage complet : Master + Slave, keypad et 7-segments.

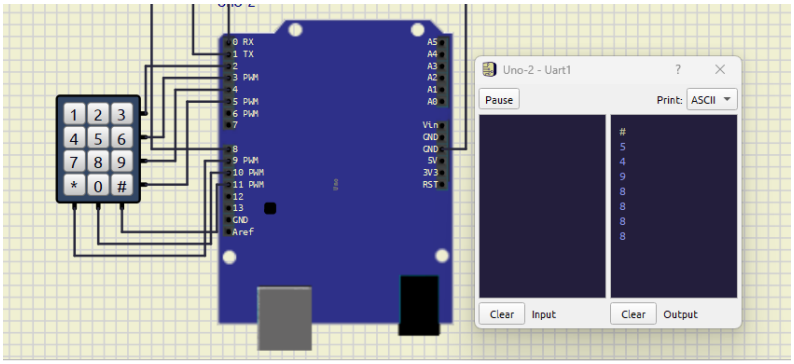

FIGURE 3 – Arduino Master : clavier matriciel (4×3) connecté aux PD2–PD5 et PB1–PB3.

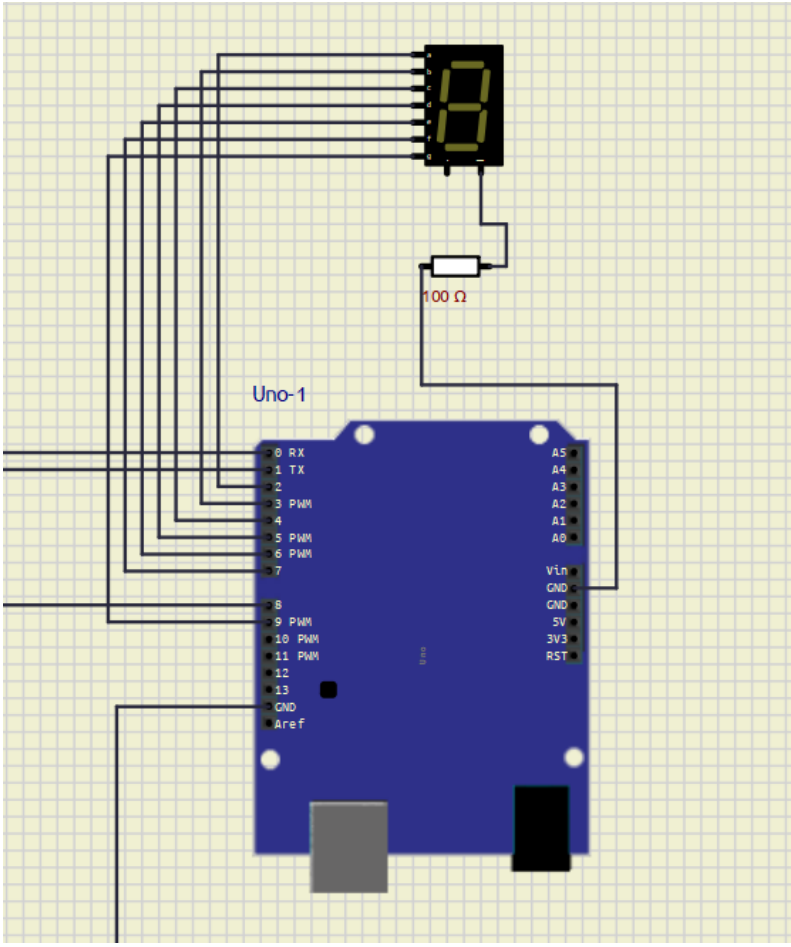

FIGURE 4 – Arduino Slave : affichage 7 segments connecté aux PD2–PD7 et PB1.