Rapport du Lab : Communication I2C entre trois cartes Arduino UNO

Réalisé par : Kennouche Abderrahmane

Matériel utilisé

- Trois cartes Arduino UNO
- Une LED
- Une résistance
- Un afficheur 7 segments (commun cathode)
- Un clavier matriciel (Keypad)
- Câbles de connexion

Câblage

Les connexions I2C sont réalisées comme suit :

- **SDA** : A4 du maître connecté aux A4 des deux esclaves
- **SCL** : A5 du maître connecté aux A5 des deux esclaves
- **GND** : toutes les masses reliées ensemble

Chaque esclave possède ensuite son propre montage :

- Esclave 1 : LED sur la broche 13 (via résistance)
- Esclave 2 : afficheur 7 segments sur les broches 2 à 8
- Maître : Keypad connecté aux broches 3 à 9

Principe du Lab

Ce lab consiste à relier trois cartes Arduino UNO par communication I2C :

- Le **maître** envoie des signaux à deux esclaves.
- L'**esclave 1** (adresse 8) reçoit des commandes pour allumer/éteindre une LED chaque seconde.
- L'**esclave 2** (adresse 9) reçoit la touche appuyée sur le clavier et l'affiche sur un afficheur 7 segments.

Code du Maître

```
#include <Wire.h>
#include <Keypad.h>

// Configuration du Keypad
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}};
```

```
byte rowPins[ROWS] = {9, 8, 7, 6}; // lignes
byte colPins[COLS] = {5, 4, 3};   // colonnes
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Variables pour LED
unsigned long previousMillis = 0; // pour  viter  les probl mes de
    d passement de valeur
const long interval = 1000;       // 1 seconde
bool ledState = false;

void setup()
{
  Wire.begin();      // Initialisation en mode ma tre
  Serial.begin(9600);
}

void loop()
{
  unsigned long currentMillis = millis(); // obtenir le temps actuel
  // Allumer /  teindre  LED sur l'esclave 1 toutes les 1s
  if (currentMillis - previousMillis >= interval)
  {
    previousMillis = currentMillis;
    ledState = !ledState; // alterne l' tat  de la LED
    Wire.beginTransmission(8); // adresse du premier esclave (LED)
    Wire.write(ledState ? 1 : 0); // envoie 1 pour allumer, 0 pour
         teindre
    Wire.endTransmission();
  }

  // Lire le Keypad et envoyer la touche   l'esclave 2
  char key = keypad.getKey();
  if (key)
  {
    Serial.print("Touche␣appuy e␣:␣");
    Serial.println(key);
    Wire.beginTransmission(9); // adresse du deuxi me esclave (7
       segments)
    Wire.write(key);           // envoie la touche
    Wire.endTransmission();
  }
}
```

Listing 1 – Programme du maître

Code de l'Esclave 1 (LED)

```
#include <Wire.h>
#include <Arduino.h>

// Fonction appel e automatiquement   la r ception d'un message
void receiveEvent(int bytes)
{
  if (Wire.available())
  {
    int cmd = Wire.read(); // lecture de la donn e re ue
```

```
    if (cmd == 1)
      digitalWrite(13, HIGH); // Allumer la LED
    else
      digitalWrite(13, LOW);  //   teindre   la LED
  }
}

void setup()
{
  pinMode(13, OUTPUT); // broche LED en sortie
  Wire.begin(8);       // adresse I2C de cet esclave
  Wire.onReceive(receiveEvent); // callback de r ception
}

void loop()
{
  delay(1000); // d lai pour stabilit
}
```

Listing 2 – Programme de l'esclave 1

Code de l'Esclave 2 (Afficheur 7 Segments)

```
#include <Wire.h>
#include <Arduino.h>

// Broches du 7 segments
int a = 2;
int b = 3;
int c = 4;
int d = 5;
int e = 6;
int f = 7;
int g = 8;

// Fonction pour afficher un chiffre sur le 7 segments
void afficherChiffre(int n)
{
  // Tableau repr sentant les chiffres de 0    9
  bool chiffres[10][7] = {
      {1, 1, 1, 1, 1, 1, 0}, // 0
      {0, 1, 1, 0, 0, 0, 0}, // 1
      {1, 1, 0, 1, 1, 0, 1}, // 2
      {1, 1, 1, 1, 0, 0, 1}, // 3
      {0, 1, 1, 0, 0, 1, 1}, // 4
      {1, 0, 1, 1, 0, 1, 1}, // 5
      {1, 0, 1, 1, 1, 1, 1}, // 6
      {1, 1, 1, 0, 0, 0, 0}, // 7
      {1, 1, 1, 1, 1, 1, 1}, // 8
      {1, 1, 1, 1, 0, 1, 1}  // 9
  };

  //   criture   des segments correspondants
  for (int i = 0; i < 7; i++)
  {
    digitalWrite(a + i, chiffres[n][i]);
```

```
  }
}

// Fonction appel e quand une donn e arrive depuis le ma tre
void receiveEvent(int bytes)
{
  char key = Wire.read(); // lecture du caract re envoy
  if (key >= '0' && key <= '9')
  {
    afficherChiffre(key - '0'); // convertir et afficher le chiffre
  }
}

void setup()
{
  Wire.begin(9); // adresse I2C de l'esclave
  Wire.onReceive(receiveEvent); // fonction de r ception

  // D claration des broches du 7 segments
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
}

void loop()
{
  delay(100); // petite pause pour  viter  le scintillement
}
```

Listing 3 – Programme de l'esclave 2

Explication du fonctionnement

- Le **maître** envoie des signaux à deux adresses différentes : 8 et 9.
- L'Arduino **esclave 1** réagit à l'adresse 8 et commande une LED.
- L'Arduino **esclave 2** réagit à l'adresse 9 et affiche la touche appuyée sur le 7 segments.
- La fonction `millis()` est utilisée pour éviter les blocages liés à `delay()`.
- Chaque Arduino travaille indépendamment dans sa propre boucle `loop()`.

Captures d'écran

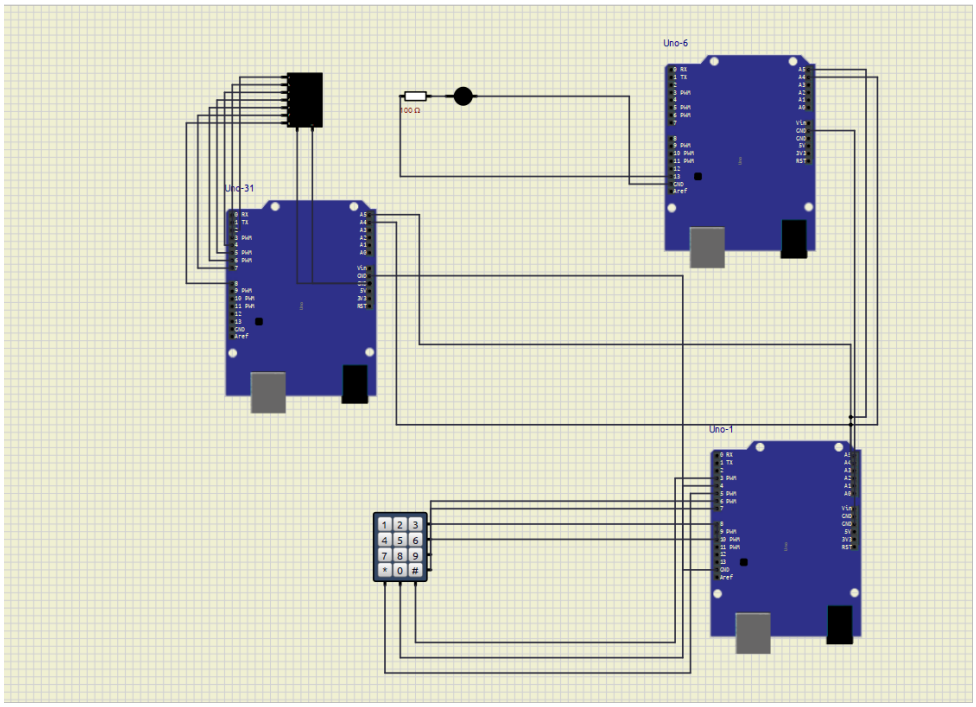

FIGURE 1 – Schéma complet de la connexion I2C

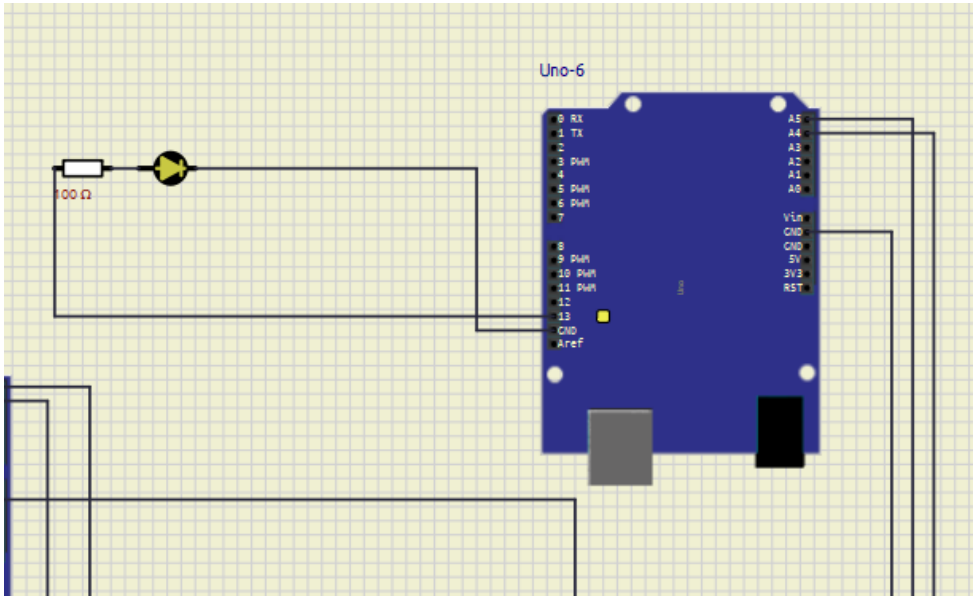

FIGURE 2 – LED allumée sur l'esclave 1.

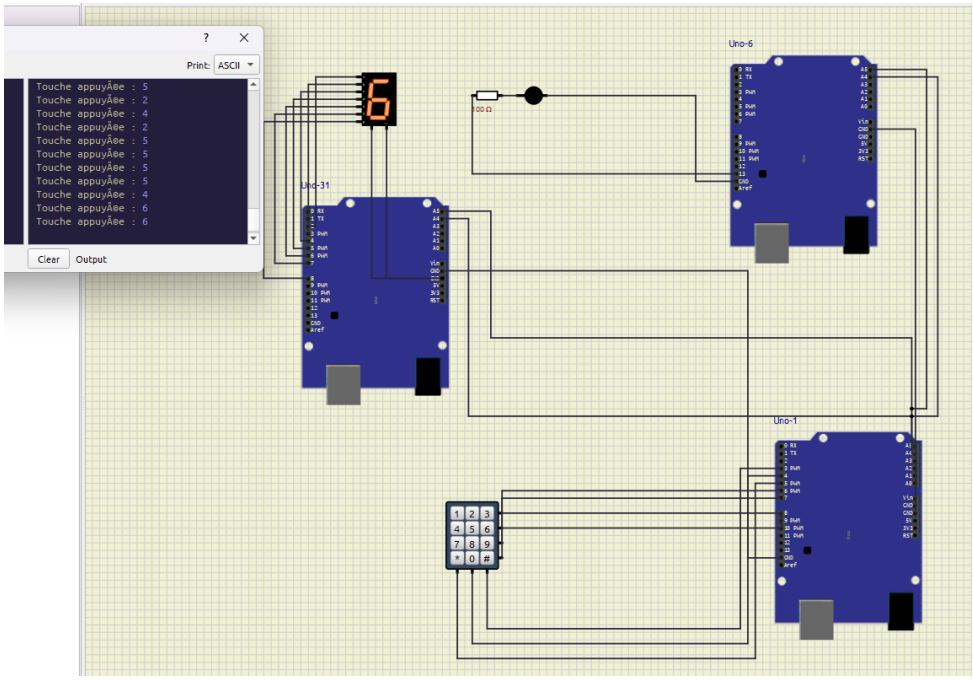

FIGURE 3 – Affichage sur le 7 segments.

Conclusion

Ce lab a permis de comprendre la communication I2C entre plusieurs cartes Arduino. Grâce à ce protocole, un seul maître peut contrôler plusieurs esclaves simultanément. La synchronisation et la modularité de l'I2C rendent ce protocole idéal pour relier plusieurs modules électroniques.