Prototypage Arduino - Rita MATAR, Yasmine HAMED, Maélanne REVOL & Marya EL NOUEIRY

Introduction à Arduino

1. Les outils et leurs utilisations

Avant de commencer à programmer et à monter des circuits avec Arduino, il est essentiel de comprendre les composants de base et leur rôle :

- **LED (Diode électroluminescente)** :
  - Permet le passage du courant dans un seul sens.
  - Possède une patte plus courte correspondant à la borne négative (cathode).
  - A une très faible résistance.
- **Résistance** :
  - Permet de limiter le courant circulant dans un circuit.
  - Utilise la loi d'Ohm : **U = R × I** (U : tension, R : résistance, I : courant).
  - La pile peut compenser les variations de courant.
- **Condensateur** :
  - Stocke et libère l'énergie électrique selon les besoins du circuit.
- **Potentiomètre** :
  - Permet de faire varier la tension.
  - Utilisé pour permettre à l'Arduino d'interagir avec l'utilisateur.
- **Transistor** :

    - Permet de contrôler des tensions plus élevées avec Arduino.
    - Par exemple, un moteur fonctionnant en 12V (trop élevé pour l'Arduino) peut être piloté à l'aide d'un transistor.
- **Régulateur de tension** :
    - Permet de convertir une tension de 12V en 5V pour adapter l'alimentation des composants.

---

2. Premiers pas avec Arduino IDE

Pour programmer notre Arduino, nous avons ouvert **Arduino IDE** et accédé à un exemple de code en suivant ces étapes :

1. **Ouverture d'un exemple de code Blink** :
    - Aller dans **File > Examples > Basics > Blink**.
    - Vérifier que la carte Arduino est bien sélectionnée :
        - **Tools > Board > Arduino AVR > Arduino Uno**.
        - **Tools > Port > Dev CU** (sélectionner le premier port détecté).
2. **Test de la carte Arduino** :
    - Nous avons d'abord vérifié que l'Arduino fonctionnait correctement en chargeant le programme Blink, qui fait clignoter une LED embarquée.
    - Ensuite, nous avons modifié la durée du clignotement en changeant la valeur des délais (de 1000ms à 5000ms).

---

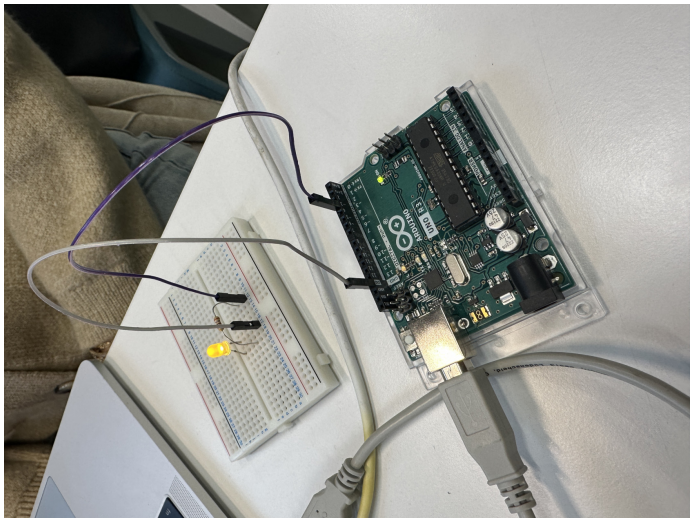
3. Réalisation des premiers montages

Premier montage : LED clignotante

Nous avons réalisé notre premier circuit en utilisant :

- Une **LED**.
- Une **résistance** pour limiter le courant.

Ensuite, nous avons écrit un programme permettant de faire clignoter la LED.

- 

Il faut mantenant faire le code pour faire glignonté la LED :

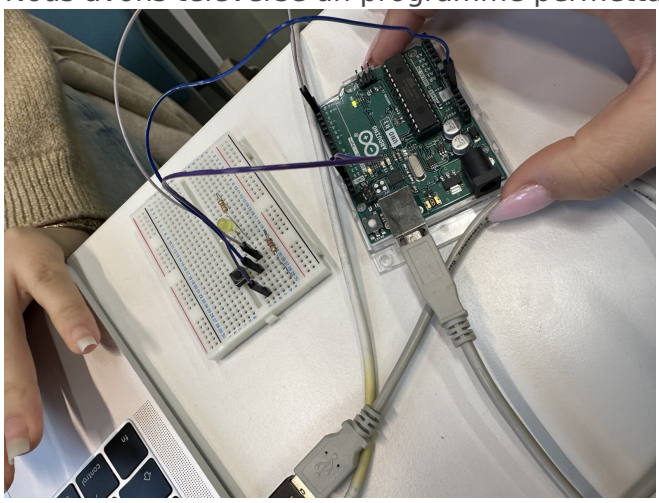
Blink.ino

```
#define LED 3

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(5000);                      // wait for a second
  digitalWrite(LED, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Deuxième montage : Bouton-poussoir

Après avoir appris à faire clignoter une LED, nous avons ajouté un **bouton-poussoir** alimenté en **5V**.

1. Nous avons téléversé un programme permettant de détecter l’appui sur le bouton.



2. Après l’upload du code, nous avons ouvert le **moniteur série** :
   - Aller dans **Tools > Serial Monitor**.
   - Une fenêtre s’affiche indiquant lorsque le bouton est pressé.

sketch_feb7b.ino

```
#define BOUTON 2

void setup() {
  pinMode(BOUTON, INPUT);
  Serial.begin(9600);
}

void loop() {
  int etat_bouton = digitalRead(BOUTON);

  if(etat_bouton == 1){
    Serial.println("Bouton pressé");
}

delay(100);
}
```

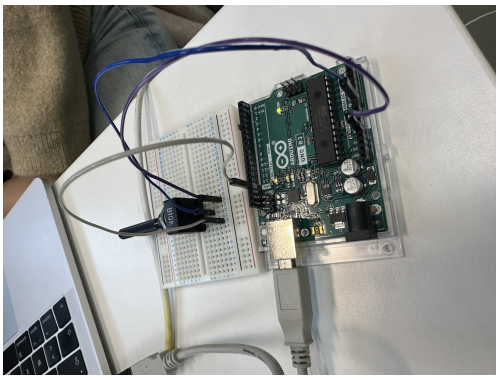
Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14301') New Line 9600 bau

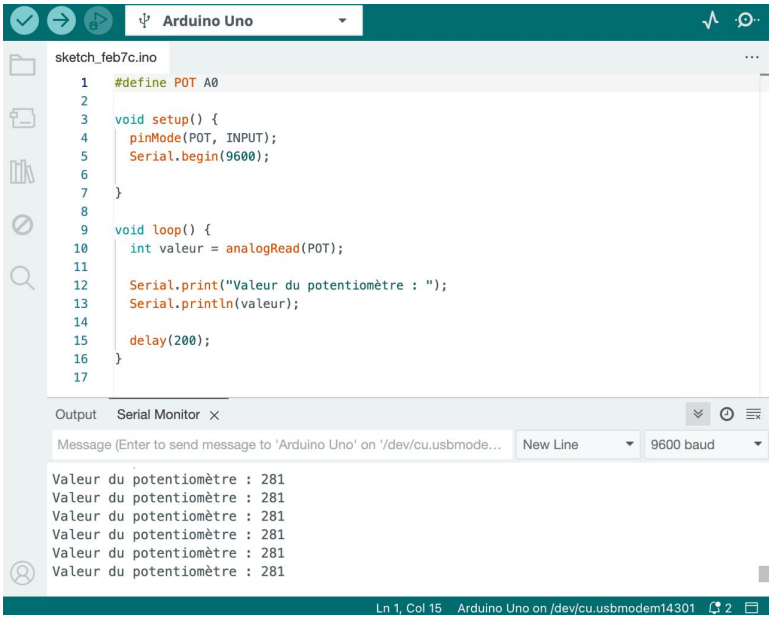
```
Bouton pressé
Bouton pressé
Bouton pressé
Bouton pressé
Bouton pressé
```

Troisième montage : Potentiomètre

Nous avons ajouté un **potentiomètre** au circuit.



- En tournant le potentiomètre, la tension de sortie varie.
- Cette variation est détectée par l’Arduino et affichée sur le moniteur série.



Quatrième montage : Capteur I2C

Nous avons expérimenté l'utilisation d'un **capteur communicant en I2C**, tel que :

- Un **capteur de luminosité**.
- Un **capteur de température**.
- Un **capteur d’humidité**.

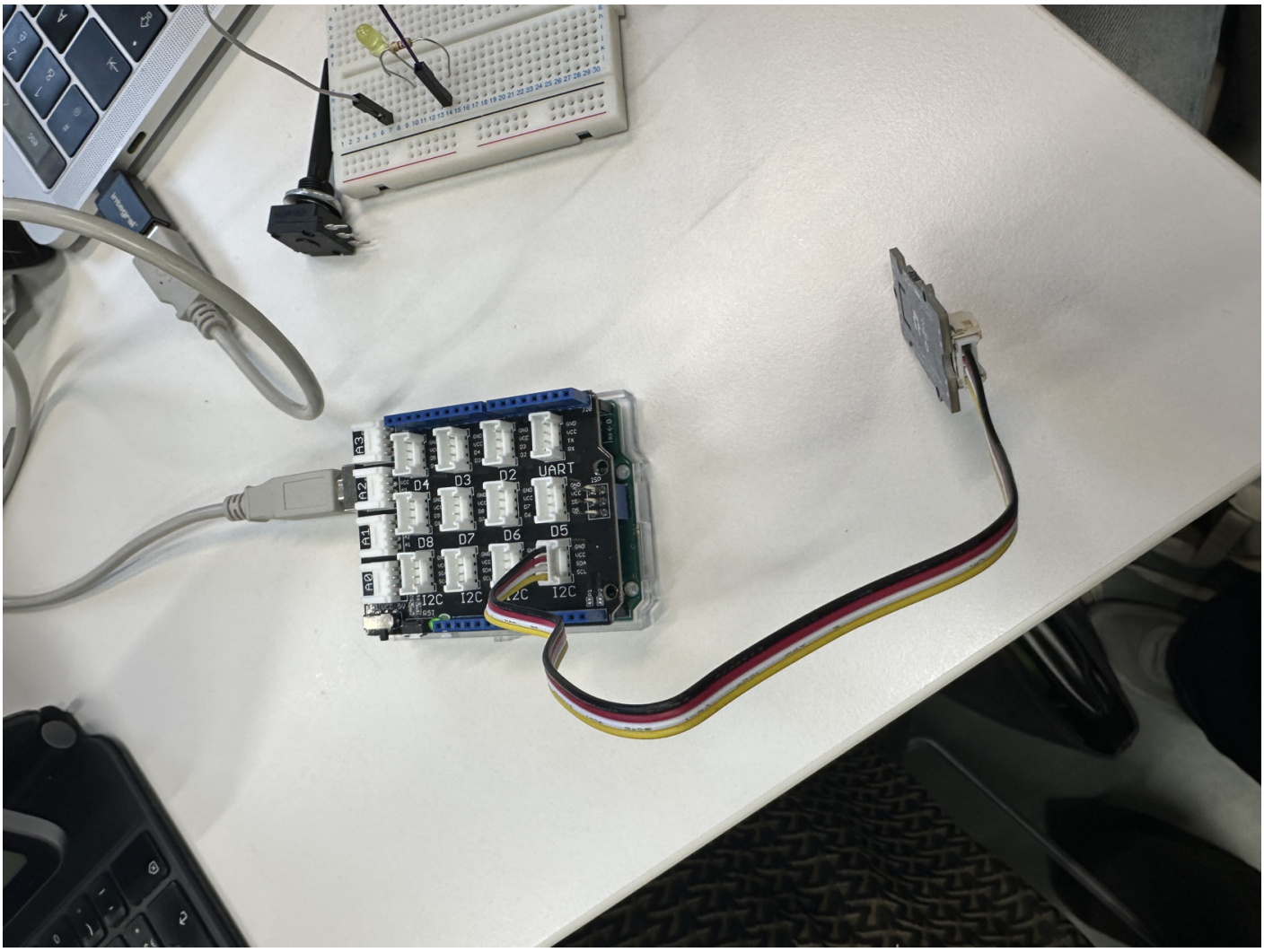
Le branchement des capteurs I2C se fait sur les broches **SDA** et **SCL**. Pour connecter ces capteurs, nous avons utilisé un **shield** qui se place par-dessus la carte Arduino.

Blink.ino

```
#define LED 3

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(5000);                      // wait for a second
  digitalWrite(LED, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Il faut maintenant chercher le code correspondant à notre capteur dans la librairie de l’application Arduino IDE. Dans notre cas, il s’agit du capteur SHT35.

Ci-dessous, le montage du capteur :



Code correspondant au capteur SHT35:

Arduino Uno

basic_demo.ino

```
#include "Seeed_SHT35.h"


/*SAMD core*/
#ifdef ARDUINO_SAMD_VARIANT_COMPLIANCE
    #define SDAPIN  20
    #define SCLPIN  21
    #define RSTPIN  7
    #define SERIAL SerialUSB
#else
    #define SDAPIN  A4
    #define SCLPIN  A5
    #define RSTPIN  2
    #define SERIAL Serial
#endif

SHT35 sensor(SCLPIN);


void setup() {
    SERIAL.begin(115200);
    delay(10);
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14501') New Line 115200 baud

```
read data :
temperature = 21.14 °C
humidity = 36.12 %
```

Arduino Uno

basic_demo.ino

```
void setup() {
    SERIAL.begin(115200);
    delay(10);
    SERIAL.println("serial start!!");
    if (sensor.init()) {
        SERIAL.println("sensor init failed!!");
    }
    delay(1000);
}


void loop() {
    u16 value = 0;
    u8 data[6] = {0};
    float temp, hum;
    if (NO_ERROR != sensor.read_meas_data_single_shot(HIGH_REP_WITH_STRCH, &temp, &hum)) {
        SERIAL.println("read temp failed!!");
        SERIAL.println("   ");
        SERIAL.println("   ");
        SERIAL.println("   ");
    } else {
        SERIAL.println("read data :");
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14501') New Line 115200 baud

```
read data :
temperature = 21.14 °C
humidity = 36.12 %
```

Arduino Uno

basic_demo.ino

```
        SERIAL.println("   ");
    } else {
        SERIAL.println("read data :");
        SERIAL.print("temperature = ");
        SERIAL.print(temp);
        SERIAL.println(" ℃ ");

        SERIAL.print("humidity = ");
        SERIAL.print(hum);
        SERIAL.println(" % ");

        SERIAL.println("   ");
        SERIAL.println("   ");
        SERIAL.println("   ");
    }
    delay(1000);
}
```

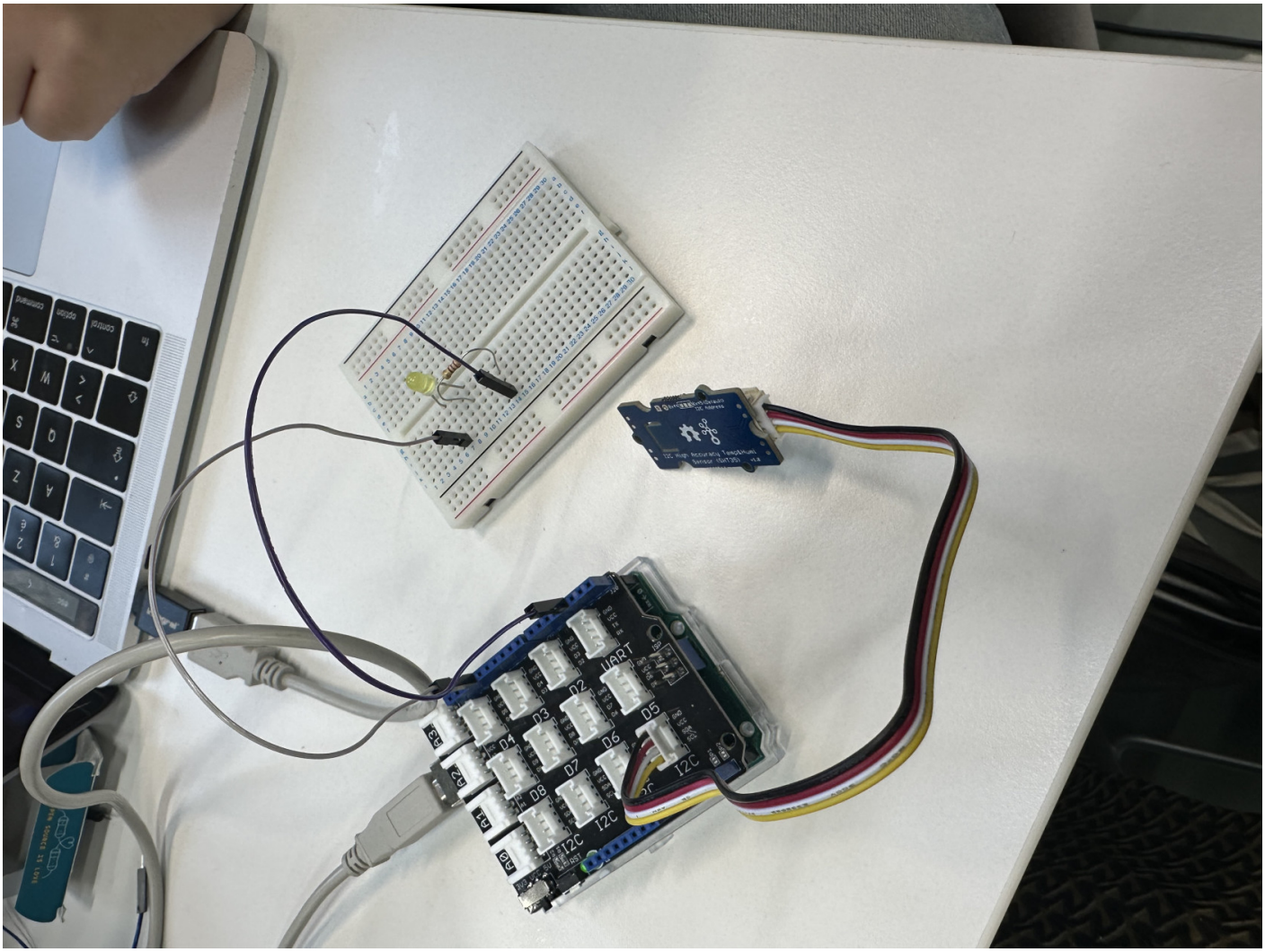
Output Serial Monitor

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14501') New Line 115200 baud

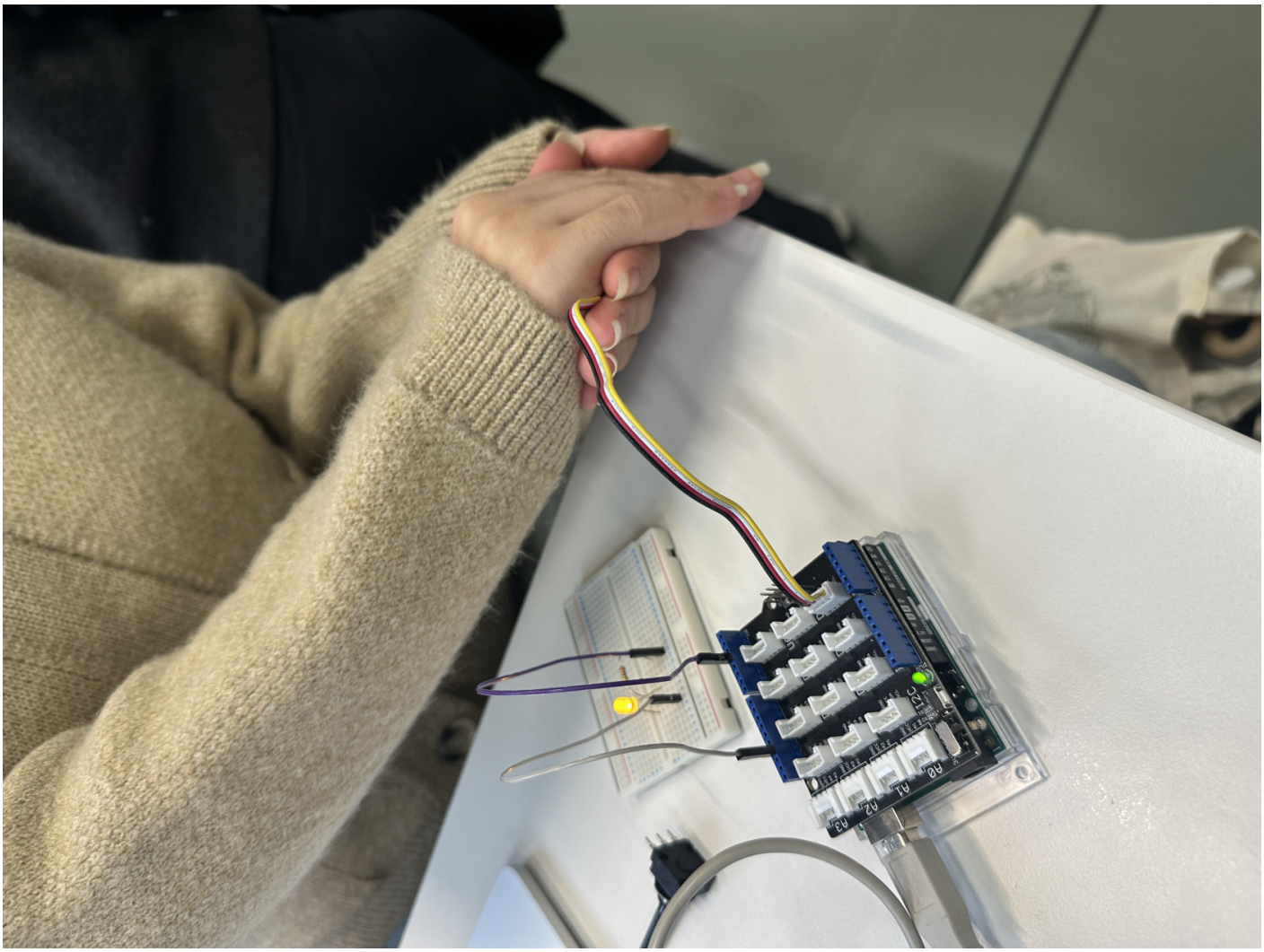
```
read data :
temperature = 21.14 °C
humidity = 36.12 %
```

Cinquième montage:

On branche une led ainsi qu’une résistance sur notre montage Arduino, shield et capteur thermique et d’humidité.

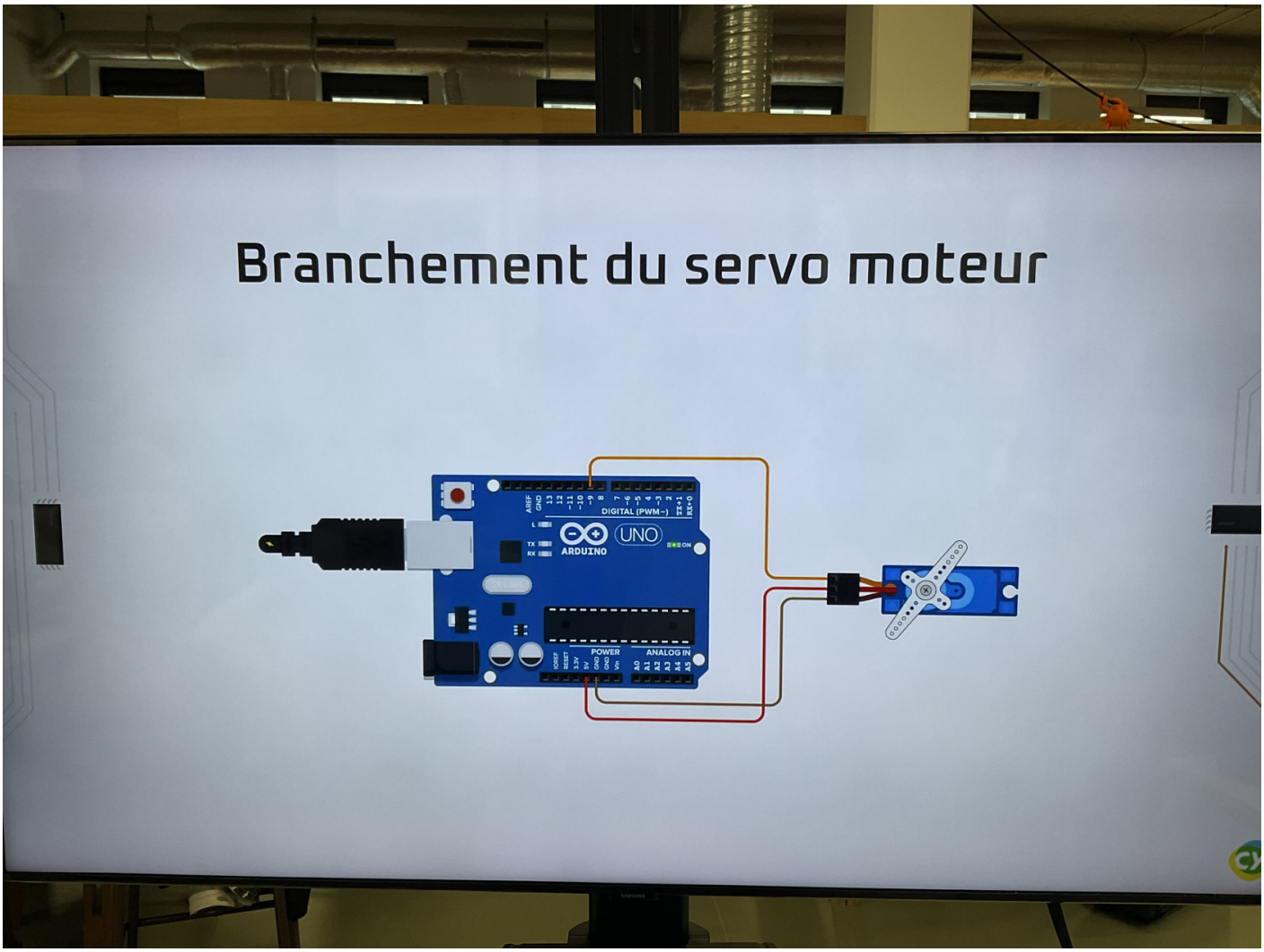


En réchauffant le capteur avec nos mains, on peut faire augmenter la température autour du capteur et ainsi allumer la led (à partir de 23°C). On observe qu’en retirant nos mains, la led s’éteint lorsqu’elle passe en dessous du seuil des 23°C.



Sixième montage:

Branchement du cerveau moteur:



Code servant à faire bouger le servo moteur grâce au potentiomètre:

Arduino Uno

Knob.ino readme.md

```
/*
 Controlling a servo position using a potentiometer (variable resistor)
 by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

 modified on 8 Nov 2013
 by Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/Knob
*/

#include <Servo.h>

Servo myservo;  // create Servo object to control a servo

int potpin = A0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the Servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it for use with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

Code servant à faire bouger le servo moteur seul:

Arduino Uno

Sweep.ino readme.md

```
/* Sweep
 by BARRAGAN <http://barraganstudio.com>
 This example code is in the public domain.

 modified 8 Nov 2013
 by Scott Fitzgerald
 https://www.arduino.cc/en/Tutorial/LibraryExamples/Sweep
*/

#include <Servo.h>

Servo myservo;  // create Servo object to control a servo
// twelve Servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the Servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
}
```

OBJECTIF POUR LE 03/03/25 :

=> Créer un code qui combine ce qu’on a vu aujourd’hui.

Ce qu’on veut faire: utiliser un capteur de température pour allumer une LED quand <23° et eteindre la LED puis activer le cerveau moteur quand >23°.

Voici ci-dessous le code à tester pour le premer devoir du 03/03/2025

```
#include "Seeed_SHT35.h"
#include <Servo.h>

#define LED 3  // Définition de la broche pour la LED
#define SERVO_PIN 9  // Définition de la broche pour le servomoteur

/*SAMD core*/
```

```
#ifdef ARDUINO_SAMD_VARIANT_COMPLIANCE
    #define SDAPIN  20
    #define SCLPIN  21
    #define RSTPIN  7
    #define SERIAL SerialUSB
#else
    #define SDAPIN  A4
    #define SCLPIN  A5
    #define RSTPIN  2
    #define SERIAL Serial
#endif

SHT35 sensor(SCLPIN);
Servo myServo;

void setup() {
    SERIAL.begin(115200);
    delay(10);
    SERIAL.println("Serial start!!");
    if (sensor.init()) {
        SERIAL.println("Sensor init failed!!");
    }
    pinMode(LED, OUTPUT); // Configuration de la LED en sortie
    myServo.attach(SERVO_PIN); // Attachement du servomoteur à sa broche
    delay(1000);
}

void loop() {
    float temp, hum;
    if (NO_ERROR != sensor.read_meas_data_single_shot(HIGH_REP_WITH_STRCH, &temp, &hum)) {
        SERIAL.println("Read temp failed!!");
    } else {
        SERIAL.println("Read data :");
        SERIAL.print("Temperature = ");
        SERIAL.print(temp);
        SERIAL.println(" °C ");

        SERIAL.print("Humidity = ");
        SERIAL.print(hum);
        SERIAL.println(" % ");
```

```
        // Allume la LED si la température est inférieure à 23°C, sinon active le servomoteur
        if (temp < 23.0) {
            digitalWrite(LED, HIGH);
            myServo.write(0); // Position neutre du servomoteur
            SERIAL.println("LED ON (Temp too low)");
        } else {
            digitalWrite(LED, LOW);
            myServo.write(90); // Active le servomoteur à 90°
            SERIAL.println("LED OFF, Servo Activated (Temp high)");
        }
    }
    delay(1000);
}
```

Voici les détails et explication du code ci dessus :

"

Ce code est écrit en **C++ pour Arduino** et utilise un **capteur de température et d'humidité SHT35**, un **servomoteur** et une **LED**. Voici une explication détaillée :

---

1. Inclusion des bibliothèques

```
#include "Seeed_SHT35.h"
#include <Servo.h>
```

- `Seeed_SHT35.h` : Bibliothèque pour gérer le capteur **SHT35**.
- `Servo.h` : Bibliothèque Arduino pour contrôler un **servomoteur**.

---

2. Définition des broches

```
#define LED 3        // La LED est connectée à la broche 3
#define SERVO_PIN 9  // Le servomoteur est connecté à la broche 9
```

- **LED** connectée à la broche **3**.
- **Servomoteur** connecté à la broche **9**.

---

3. Définition des broches I2C pour le capteur SHT35

```
#ifdef ARDUINO_SAMD_VARIANT_COMPLIANCE
  #define SDAPIN  20
  #define SCLPIN  21
  #define RSTPIN  7
  #define SERIAL SerialUSB
#else
  #define SDAPIN  A4
  #define SCLPIN  A5
  #define RSTPIN  2
  #define SERIAL Serial
#endif
```

- Ces lignes définissent les broches **SDA (données)** et **SCL (horloge)** pour la communication I2C.
- `ARDUINO_SAMD_VARIANT_COMPLIANCE` est utilisé pour vérifier si la carte est **un microcontrôleur SAMD**.
  - **Si c'est une carte SAMD** → SDA = **20**, SCL = **21**, RST = **7**, communication via `SerialUSB`.
  - **Sinon (Arduino classique)** → SDA = **A4**, SCL = **A5**, RST = **2**, communication via `Serial`.

---

4. Création des objets

```
SHT35 sensor(SCLPIN);
Servo myServo;
```

- `SHT35 sensor(SCLPIN);` : Crée un objet pour le capteur **SHT35**, avec la broche `SCLPIN`.
- `Servo myServo;` : Crée un objet pour le **servomoteur**.

---

5. Configuration initiale (`setup`)

```
void setup() {
  SERIAL.begin(115200);  // Initialise la communication série
  delay(10);
  SERIAL.println("Serial start!!");

  if (sensor.init()) {  // Initialise le capteur
    SERIAL.println("Sensor init failed!!");
  }
```

```
  pinMode(LED, OUTPUT);      // Configure la LED en sortie
  myServo.attach(SERVO_PIN); // Attache le servomoteur à la broche 9
  delay(1000);
}
```

- Initialise **la communication série** (`Serial` ou `SerialUSB`).
- Initialise **le capteur SHT35** (affiche un message si l'initialisation échoue).
- Configure la **LED comme sortie**.
- Attache le **servomoteur** à la broche définie (`SERVO_PIN`).

---

6. Boucle principale (`loop`)

```
void loop() {
  float temp, hum;
  if (NO_ERROR != sensor.read_meas_data_single_shot(HIGH_REP_WITH_STRCH, &temp, &hum)) {
    SERIAL.println("Read temp failed!!");
  } else {
    SERIAL.println("Read data :");
    SERIAL.print("Temperature = ");
    SERIAL.print(temp);
    SERIAL.println(" °C ");

    SERIAL.print("Humidity = ");
    SERIAL.print(hum);
    SERIAL.println(" % ");
```

- Déclare les variables `temp` (température) et `hum` (humidité).
- Lit les **données du capteur** `SHT35`.
  - Si la lecture échoue → Affiche `"Read temp failed!!"`.
  - Sinon → Affiche la **température et l'humidité** dans le **moniteur série**.

---

7. Contrôle de la LED et du servomoteur

```
    if (temp < 23.0) {
      digitalWrite(LED, HIGH);   // Allume la LED
      myServo.write(0);          // Place le servomoteur à 0°
      SERIAL.println("LED ON (Temp too low)");
    } else {
      digitalWrite(LED, LOW);    // Éteint la LED
      myServo.write(90);         // Fait tourner le servomoteur à 90°
```

```
        SERIAL.println("LED OFF, Servo Activated (Temp high)");
    }
```

- **Si la température est inférieure à 23°C** :
  - **Allume la LED**.
  - **Place le servomoteur à 0°**.
  - Affiche `"LED ON (Temp too low)"` dans le moniteur série.
- **Sinon (température ≥ 23°C)** :
  - **Éteint la LED**.
  - **Tourne le servomoteur à 90°**.
  - Affiche `"LED OFF, Servo Activated (Temp high)"`.

---

8. Pause avant la prochaine mesure

```
    }
    delay(1000);
}
```

- Attend **1 seconde** avant de recommencer la boucle.

Résumé du fonctionnement

1. **Initialisation** :
   - Initialise la communication série.
   - Initialise le capteur SHT35.
   - Configure la LED et le servomoteur.
2. **Lecture des données** :
   - Récupère la température et l'humidité.
3. **Action selon la température** :
   - **Si température < 23°C** : Allume la **LED**, place le **servomoteur à 0°**.
   - **Si température ≥ 23°C** : Éteint la **LED**, met le **servomoteur à 90°**.
4. **Attente de 1 seconde** avant de recommencer.

Application possible

Ce programme peut être utilisé pour :

- **Une serre automatique** (ouverture d'une trappe en fonction de la température).
- **Un système de ventilation**.
- **Un mécanisme d’alerte** avec une LED et un servomoteur."

Revision #7
Created 7 February 2025 15:01:33 by Matar Rita
Updated 3 March 2025 10:11:06 by Hamed Yasmine