

# 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 :  $U = R \times I$  (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).
- 

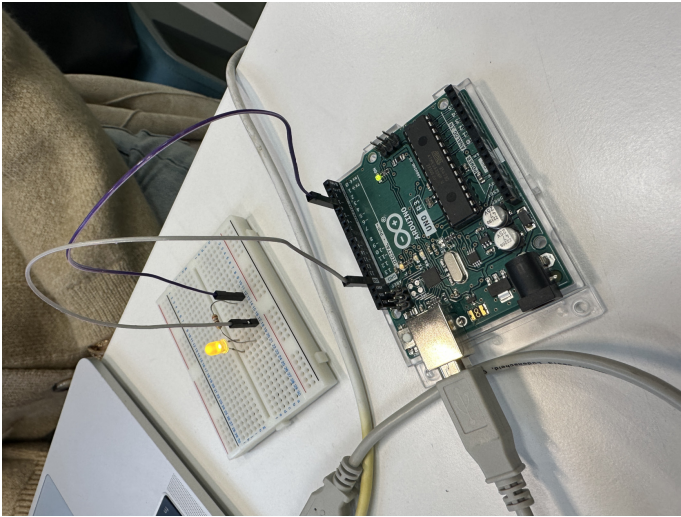
## 3. Réalisation des premiers montages

### 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 maintenant faire le code pour faire clignoter la LED :

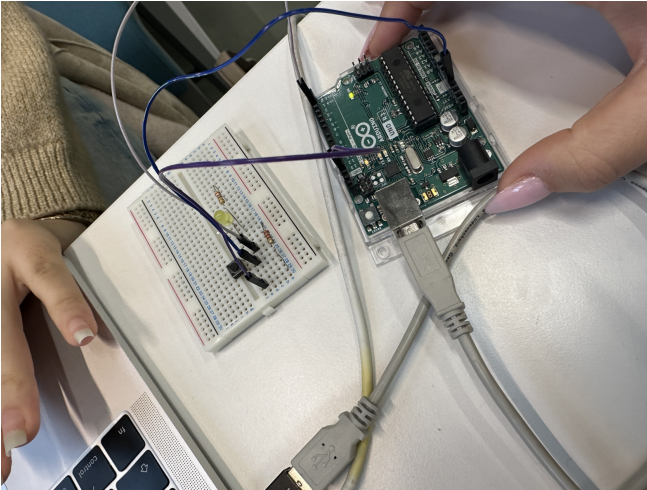
```
Blink.ino
1  #define LED 3
2
3  // the setup function runs once when you press reset or power the board
4  void setup() {
5      // initialize digital pin LED_BUILTIN as an output.
6      pinMode(LED, OUTPUT);
7  }
8
9  // the loop function runs over and over again forever
10 void loop() {
11     digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
12     delay(5000);             // wait for a second
13     digitalWrite(LED, LOW);  // turn the LED off by making the voltage LOW
14     delay(1000);             // wait for a second
15 }
16
```

---

## 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

```
1  #define BOUTON 2
2
3  void setup() {
4      pinMode(BOUTON, INPUT);
5      Serial.begin(9600);
6  }
7
8  void loop() {
9      int etat_bouton = digitalRead(BOUTON);
10
11      if(etat_bouton == 1){
12          Serial.println("Bouton pressé");
13      }
14
15      delay(100);
16  }
17
18
19
```

Output Serial Monitor x

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

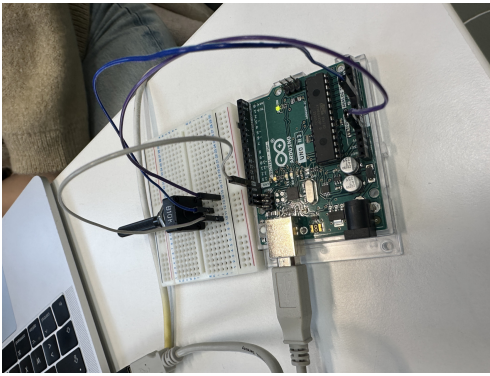
New Line

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.

```
1 #define POT A0
2
3 void setup() {
4   pinMode(POT, INPUT);
5   Serial.begin(9600);
6 }
7
8
9 void loop() {
10  int valeur = analogRead(POT);
11
12  Serial.print("Valeur du potentiomètre : ");
13  Serial.println(valeur);
14
15  delay(200);
16 }
17
```

Output Serial Monitor x

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

Valeur du potentiomètre : 281  
Valeur du potentiomètre : 281  
Valeur du potentiomètre : 281  
Valeur du potentiomètre : 281  
Valeur du potentiomètre : 281  
Valeur du potentiomètre : 281

Ln 1, Col 15 Arduino Uno on /dev/cu.usbmodem14301 2

## 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.

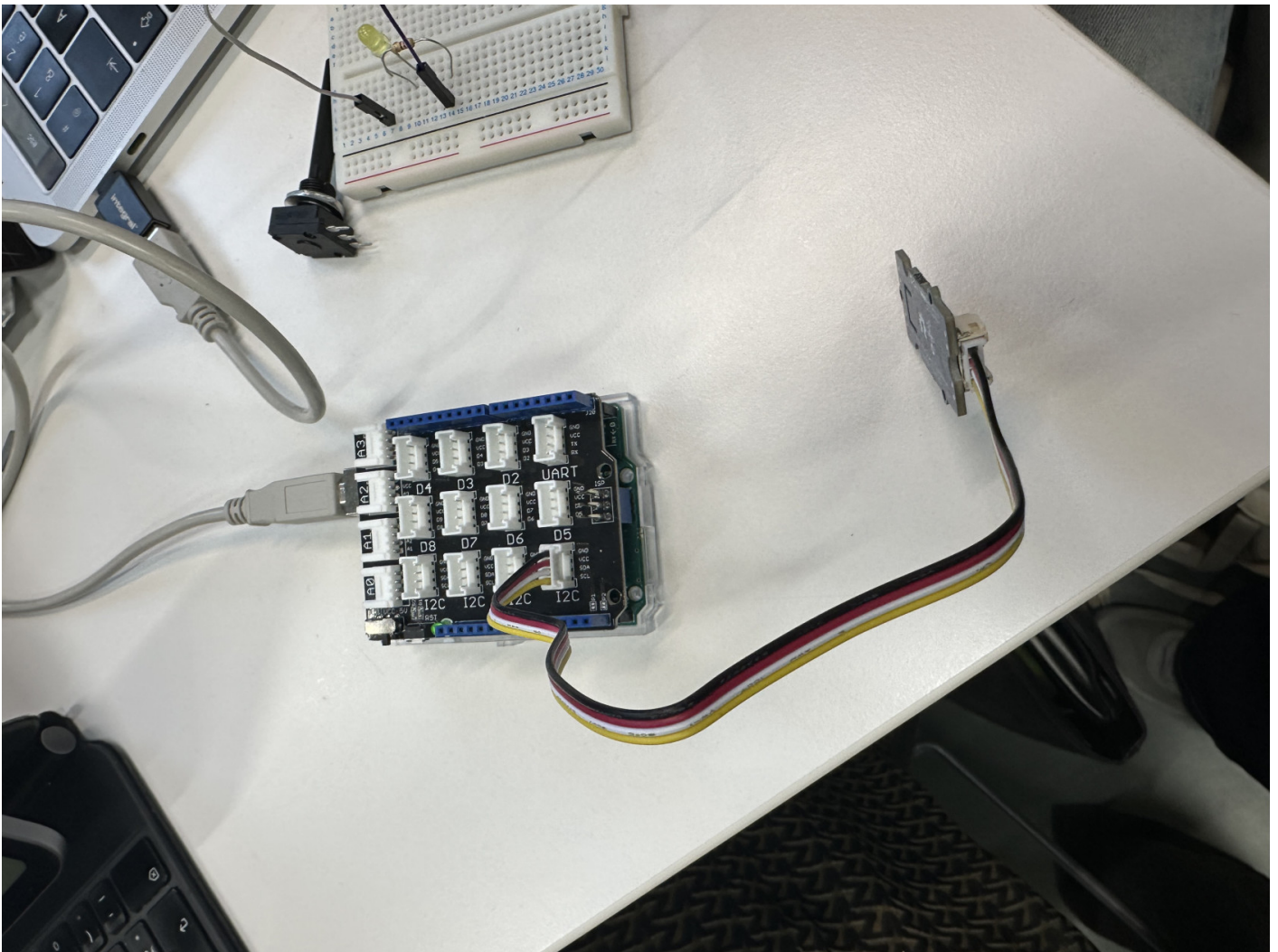


Blink.ino

```
1 #define LED 3
2
3 // the setup function runs once when you press reset or power the board
4 void setup() {
5   // initialize digital pin LED_BUILTIN as an output.
6   pinMode(LED, OUTPUT);
7 }
8
9 // the loop function runs over and over again forever
10 void loop() {
11   digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
12   delay(5000);             // wait for a second
13   digitalWrite(LED, LOW);  // turn the LED off by making the voltage LOW
14   delay(1000);             // wait for a second
15 }
16
```

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

```
1  #include "Seeed_SHT35.h"
2
3
4  /*SAMD core*/
5  #ifdef ARDUINO_SAMD_VARIANT_COMPLIANCE
6      #define SDAPIN  20
7      #define SCLPIN  21
8      #define RSTPIN  7
9      #define SERIAL SerialUSB
10 #else
11     #define SDAPIN  A4
12     #define SCLPIN  A5
13     #define RSTPIN  2
14     #define SERIAL Serial
15 #endif
16
17 SHT35 sensor(SCLPIN);
18
19
20 void setup() {
21     SERIAL.begin(115200);
22     delay(10);
23 }
```

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

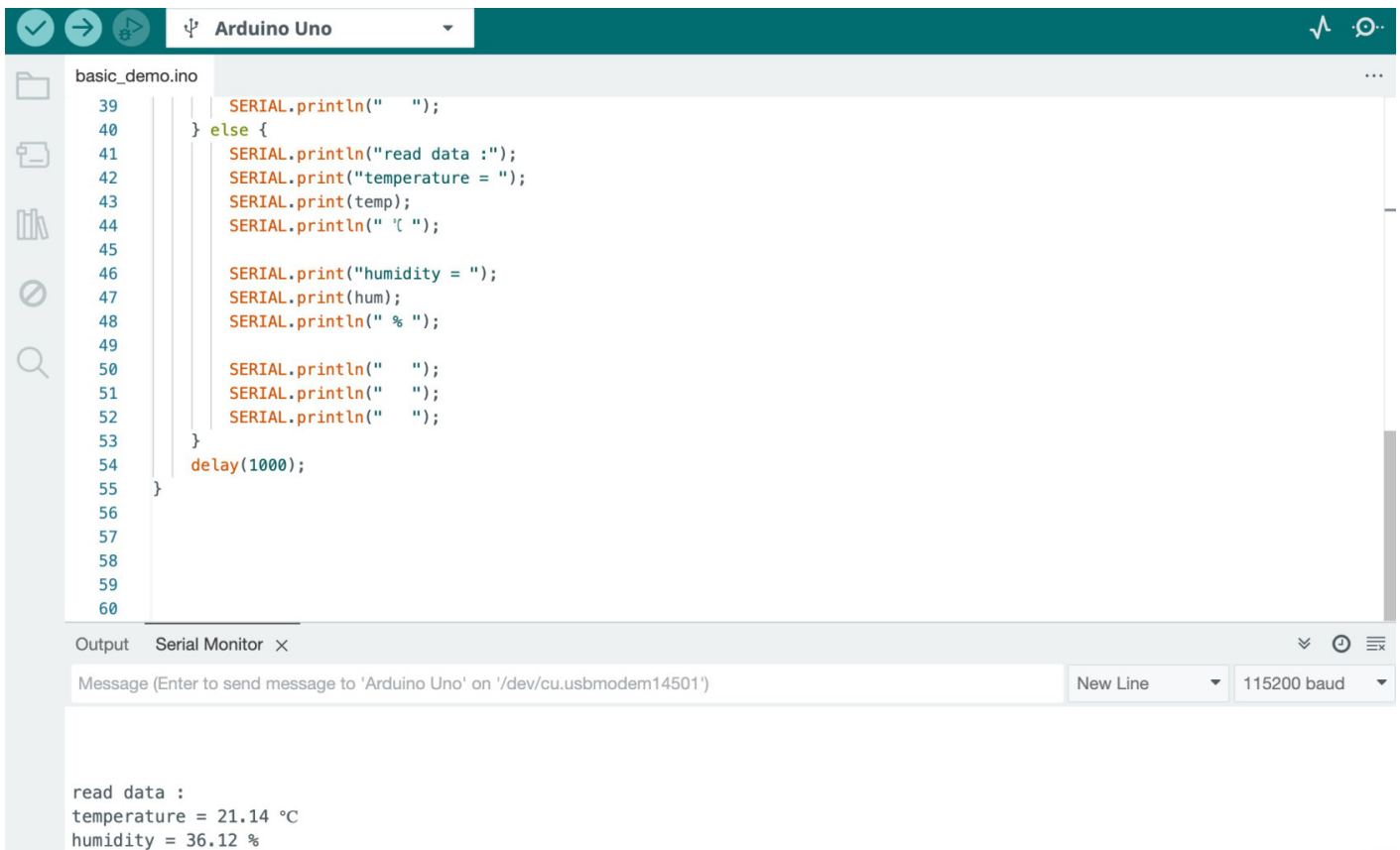
basic\_demo.ino

```
20 void setup() {
21     SERIAL.begin(115200);
22     delay(10);
23     SERIAL.println("serial start!!");
24     if (sensor.init()) {
25         SERIAL.println("sensor init failed!!");
26     }
27     delay(1000);
28 }
29
30
31 void loop() {
32     u16 value = 0;
33     u8 data[6] = {0};
34     float temp, hum;
35     if (NO_ERROR != sensor.read_meas_data_single_shot(HIGH_REP_WITH_STRCH, &temp, &hum)) {
36         SERIAL.println("read temp failed!!");
37         SERIAL.println(" ");
38         SERIAL.println(" ");
39         SERIAL.println(" ");
40     } else {
41         SERIAL.println("read data :");
42     }
43 }
```

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 %



The screenshot shows the Arduino IDE interface. The top toolbar includes icons for checking, running, and uploading code, along with a dropdown menu set to 'Arduino Uno'. The main editor window displays a C++ sketch named 'basic\_demo.ino' with the following code:

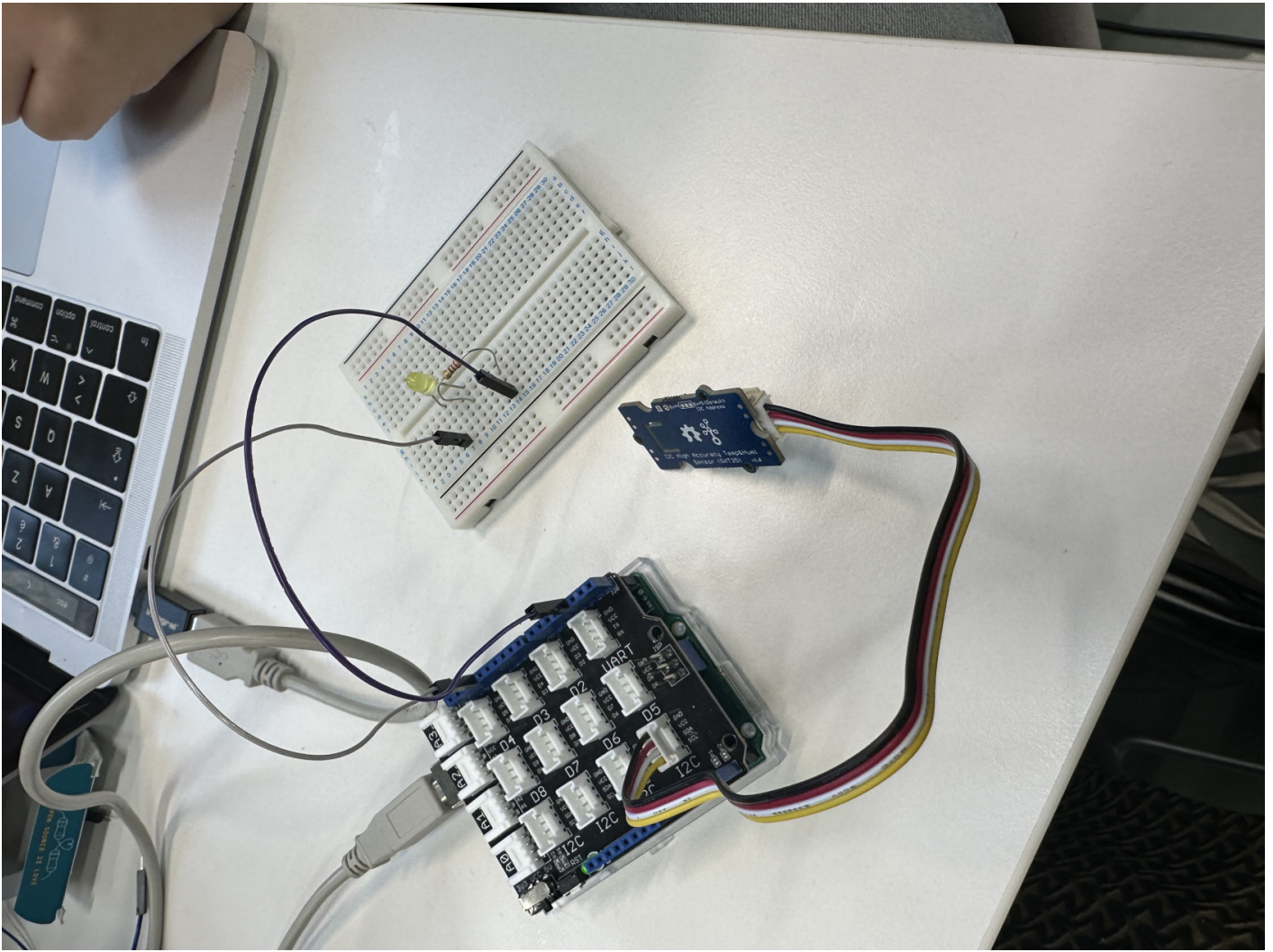
```
39     SERIAL.println(" ");
40   } else {
41     SERIAL.println("read data :");
42     SERIAL.print("temperature = ");
43     SERIAL.print(temp);
44     SERIAL.println(" °C ");
45
46     SERIAL.print("humidity = ");
47     SERIAL.print(hum);
48     SERIAL.println(" % ");
49
50     SERIAL.println(" ");
51     SERIAL.println(" ");
52     SERIAL.println(" ");
53   }
54   delay(1000);
55 }
56
57
58
59
60
```

Below the editor is the 'Serial Monitor' window, which is currently empty. The status bar at the bottom indicates the output device is 'Arduino Uno' on '/dev/cu.usbmodem14501' with a baud rate of 115200.

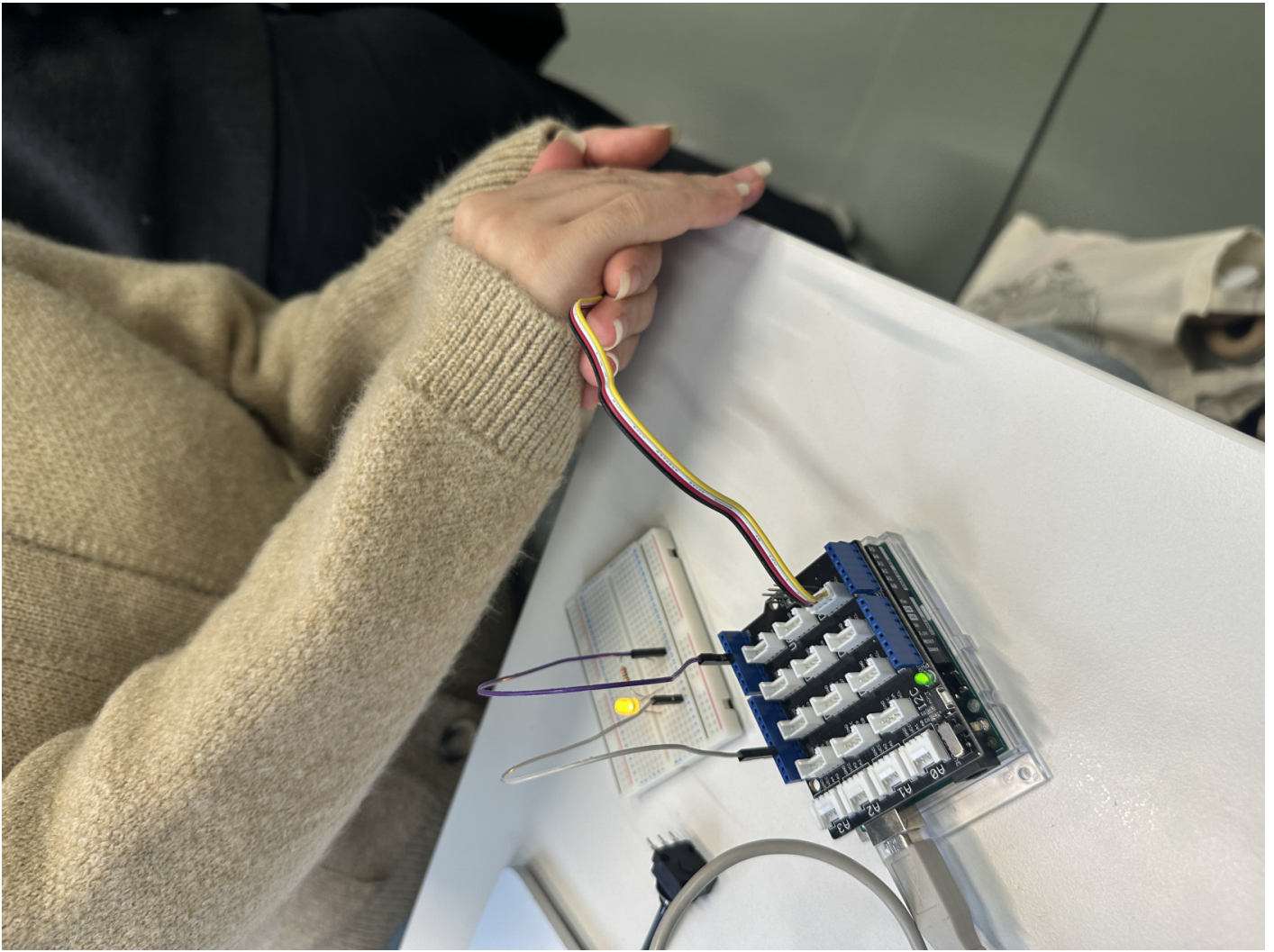
# Cinquième montage:

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.

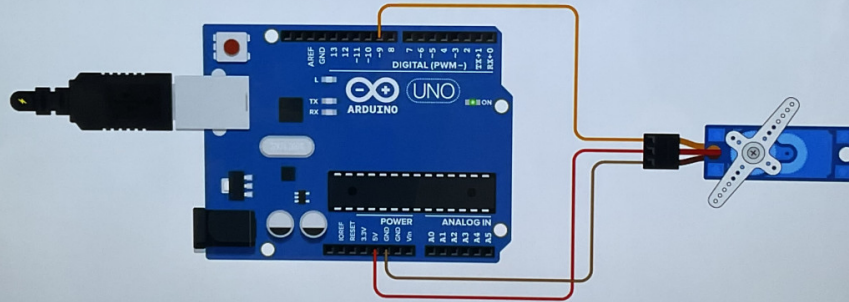


## Sixième montage:

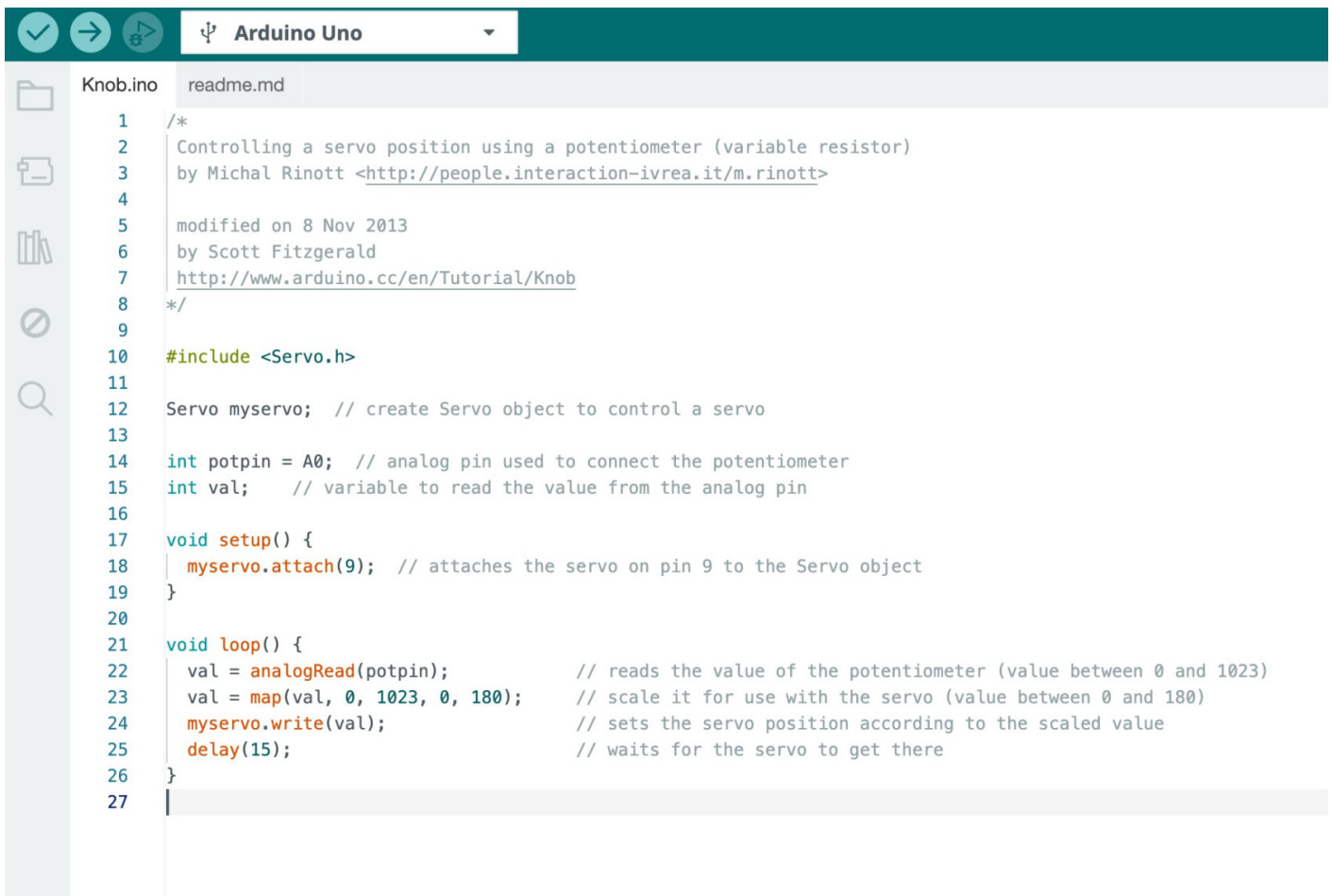
Branchement du cerveau moteur:



# Branchement du servo moteur



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



```
1  /*
2  Controlling a servo position using a potentiometer (variable resistor)
3  by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>
4
5  modified on 8 Nov 2013
6  by Scott Fitzgerald
7  http://www.arduino.cc/en/Tutorial/Knob
8  */
9
10 #include <Servo.h>
11
12 Servo myservo;  // create Servo object to control a servo
13
14 int potpin = A0;  // analog pin used to connect the potentiometer
15 int val;  // variable to read the value from the analog pin
16
17 void setup() {
18   myservo.attach(9);  // attaches the servo on pin 9 to the Servo object
19 }
20
21 void loop() {
22   val = analogRead(potpin);  // reads the value of the potentiometer (value between 0 and 1023)
23   val = map(val, 0, 1023, 0, 180);  // scale it for use with the servo (value between 0 and 180)
24   myservo.write(val);  // sets the servo position according to the scaled value
25   delay(15);  // waits for the servo to get there
26 }
27
```

Code servant à faire bouger le servo moteur seul:



```
1  /* Sweep
2  by BARRAGAN <http://barraganstudio.com>
3  This example code is in the public domain.
4
5  modified 8 Nov 2013
6  by Scott Fitzgerald
7  https://www.arduino.cc/en/Tutorial/LibraryExamples/Sweep
8  */
9
10 #include <Servo.h>
11
12 Servo myservo;  // create Servo object to control a servo
13 // twelve Servo objects can be created on most boards
14
15 int pos = 0;    // variable to store the servo position
16
17 void setup() {
18   myservo.attach(9);  // attaches the servo on pin 9 to the Servo object
19 }
20
21 void loop() {
22   for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
23     // in steps of 1 degree
24     myservo.write(pos);              // tell servo to go to position in variable 'pos'
25     delay(15);                       // waits 15 ms for the servo to reach the position
26   }
27   for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
28     myservo.write(pos);              // tell servo to go to position in variable 'pos'
29     delay(15);                       // waits 15 ms for the servo to reach the position
30   }
31 }
32
```

## **OBJECTIF POUR LA PROCHAINE FOIS :**

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

Ce qu'on veut faire: utiliser le capteur pour pouvoir faire monter la température et donc faire bouger le servo moteur à partir de 23°C.

## **IDÉE DE PROJET FINAL :**

Une version simplifiée du projet de la **\*\*mini serre automatisée\*\*** pourrait être de créer un **\*\*système d'irrigation automatique à base de capteur d'humidité\*\***, sans intégrer d'écran LCD ni autres éléments complexes.

Voici une version plus simple :

**### \*\*Projet : Système d'irrigation automatique simplifié\*\***

- **\*\*Définition du projet : \*\*** Créer un système d'arrosage automatique pour une plante en utilisant un capteur d'humidité du sol. Lorsqu'un seuil d'humidité bas est détecté, une pompe se met en marche pour arroser la plante.

- **Technologies :**

- **Arduino** pour le contrôle du capteur d'humidité et de la pompe.
- **3D printing** pour fabriquer un petit boîtier pour le capteur et la pompe, ou pour une base de support si nécessaire.

- **Fonctionnalité principale :**

- Le capteur mesure l'humidité du sol.
- Si l'humidité est trop basse (sol sec), le système déclenche l'arrosage via une petite pompe.
- Une fois l'humidité du sol suffisante, l'arrosage s'arrête automatiquement.

- **Idée d'amélioration simplifiée :**

- Remplacer l'écran LCD par une simple LED qui s'allume lorsque le système arrose, indiquant que l'irrigation est en cours.

**Plan simplifié de réalisation :**

1. **Capteur d'humidité** : Utilisation d'un capteur d'humidité de sol basique, facile à intégrer avec Arduino.
2. **Pompe à eau** : Une petite pompe submersible alimentée par Arduino ou via une source externe.
3. **Arduino** : Programmation de l'Arduino pour surveiller en continu le niveau d'humidité et contrôler la pompe d'arrosage.
4. **Boîtier imprimé en 3D** : Un boîtier simple pour protéger les composants électroniques et fixer le capteur dans le sol.

**Liste du matériel :**

- Arduino Uno ou Nano
- Capteur d'humidité de sol
- Petite pompe à eau 5V
- Transistor ou relais pour contrôler la pompe
- LED pour signaler l'arrosage
- Tubes pour l'irrigation (si nécessaire)
- Imprimante 3D et filament pour fabriquer un support

**Avantages :**

- Facilité d'assemblage et de programmation pour des débutants.
- Moins de composants à gérer (pas d'écran LCD ou de capteurs supplémentaires).
- Moins de risques d'échec et moins de temps nécessaire.

Qu'en penses-tu ? Cette version te semble-t-elle plus accessible tout en répondant à ton besoin ?