

# TP MOSH - BENAZZOUZ - HEBRAS

## TP1A : My first program : Blink

Un script nous est fourni :

```
int ledPin = 13;      // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup() {
    // initialize the digital pin as an output:
    pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
    digitalWrite(ledPin, HIGH);    // set the LED on
    delay(1000);                 // wait for a second
    digitalWrite(ledPin, LOW);     // set the LED off
    delay(1000);                 // wait for a second
}
```

Il nous permet de faire clignoter la LED pin 13, cependant on utilise un delay qui est très contraignant dans un code plus complexe, car il bloque l'exécution, cad que rien d'autre ne s'effectue pendant ce temps là

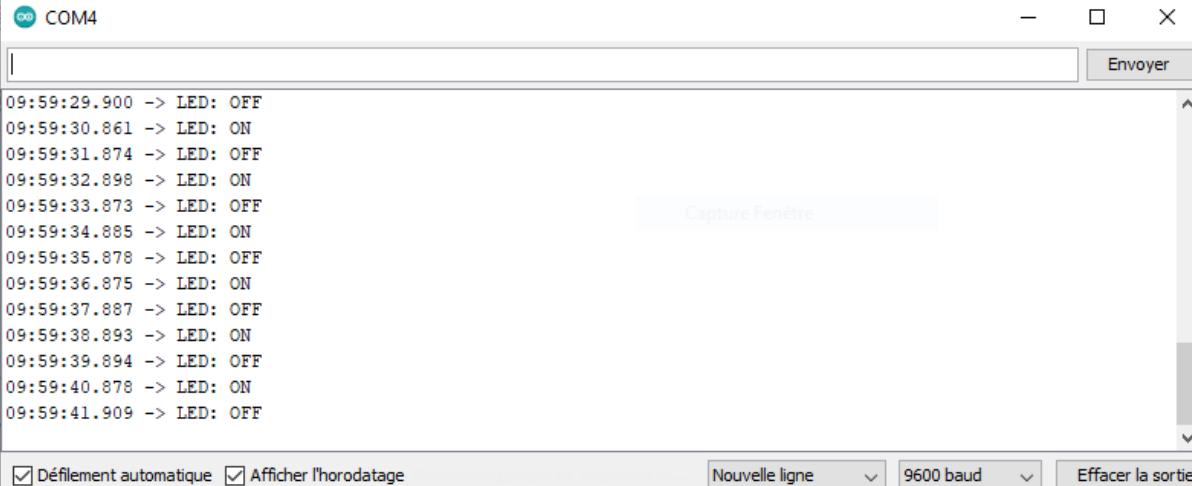
## TP1B: Blink without delay

une autre méthode pour faire le blinking est en mesurant le temps passé entre deux changements d'état, pour cela on initialise deux variables currentMillis et previousMillis et on calcule la différence entre les deux. le currentMillis est déterminé grâce à la fonction "Millis()" qui renvoie le temps écoulé depuis que la carte a commencé à fonctionner.

```
if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    if (ledState == LOW) {
        ledState = HIGH;
    } else {
        ledState = LOW;
    }
}
```

# TP1C : Serial Monitor (Arduino => PC)

On observe :

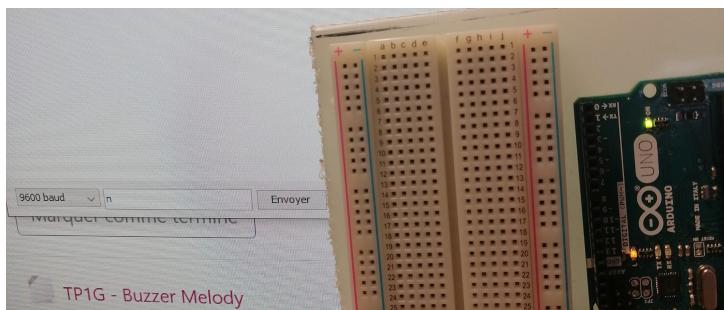


The screenshot shows the Arduino Serial Monitor window titled "COM4". The text area displays a log of LED state changes:

```
09:59:29.900 -> LED: OFF
09:59:30.861 -> LED: ON
09:59:31.874 -> LED: OFF
09:59:32.898 -> LED: ON
09:59:33.873 -> LED: OFF
09:59:34.885 -> LED: ON
09:59:35.878 -> LED: OFF
09:59:36.875 -> LED: ON
09:59:37.887 -> LED: OFF
09:59:38.893 -> LED: ON
09:59:39.894 -> LED: OFF
09:59:40.878 -> LED: ON
09:59:41.909 -> LED: OFF
```

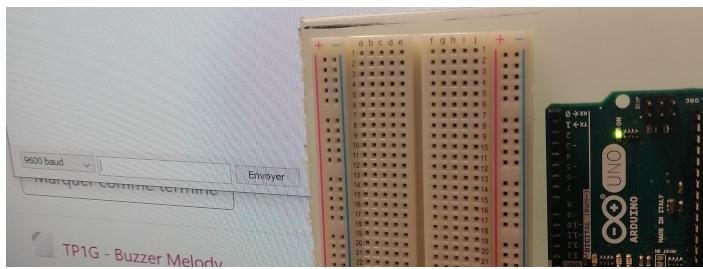
At the bottom, there are checkboxes for "Défilement automatique" and "Afficher l'horodatage", and buttons for "Nouvelle ligne", "9600 baud", and "Effacer la sortie".

# TP1D : Serial Monitor (PC => Arduino)



étape 1 : la led est allumée, on tape “n”

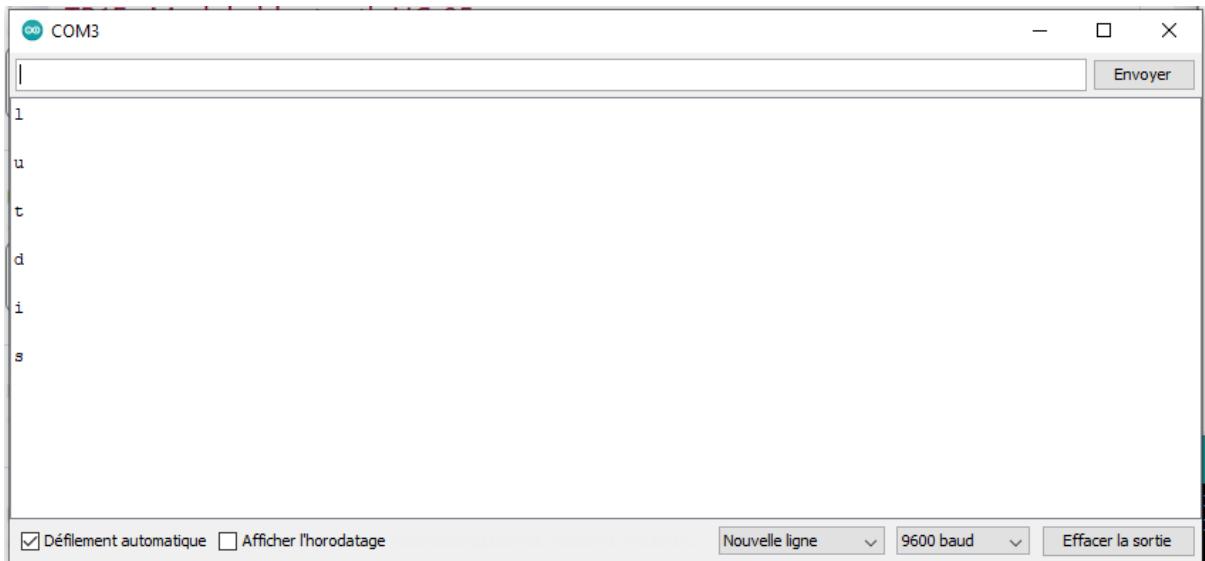
étape 2 : on tape “entrer”

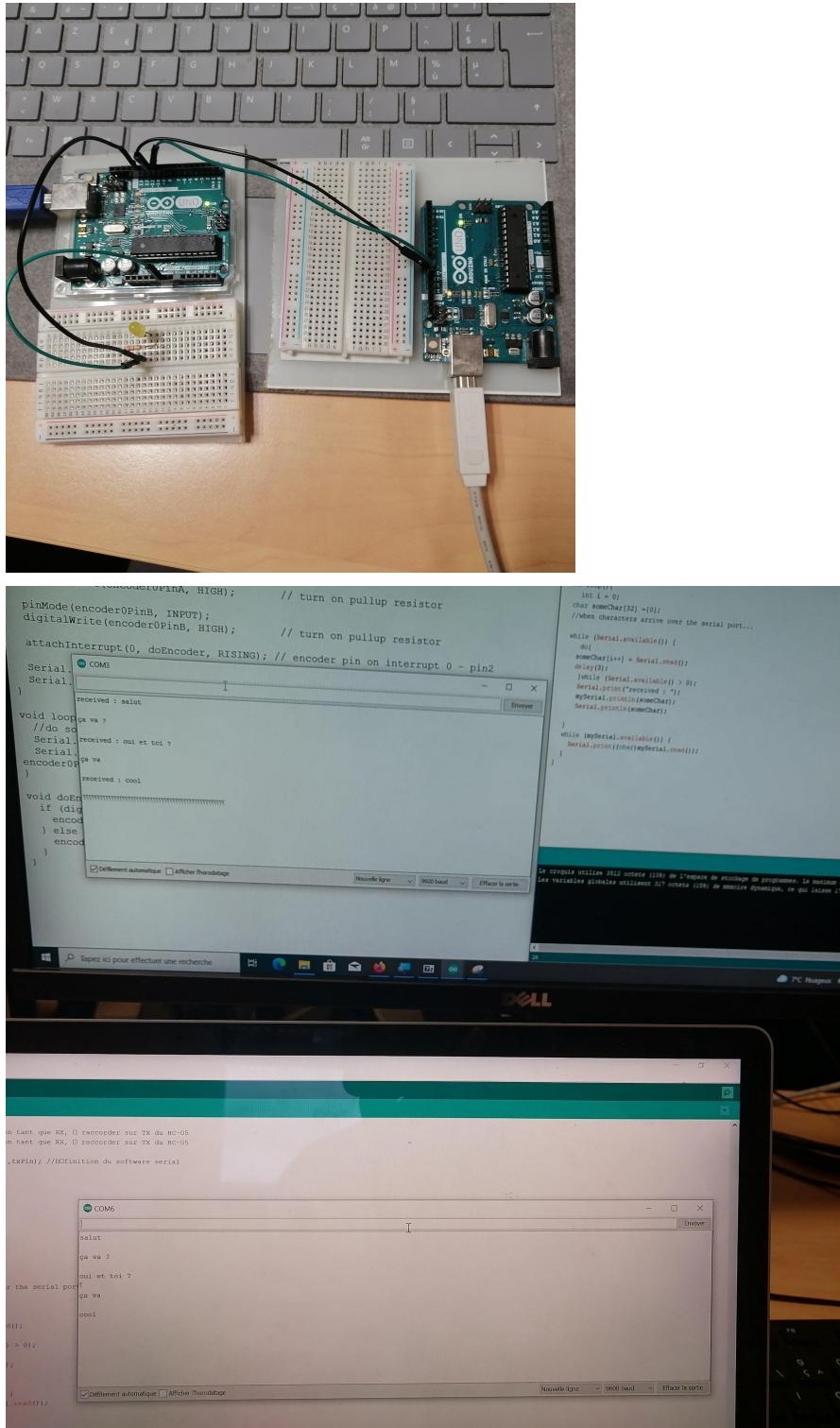


étape 3 : la led s'éteint

Remarque: les led RX et TX s'allument respectivement à l'envoi du PC vers la carte et de la carte vers le PC.

# TP1E : Serial Monitor





## TP1F: Encodeur rotatoire

En faisant tourner l'encodeur, on observe la variation de notre variable. Cependant, des rebonds ajoutent du bruit au signal. Pour les enlever, nous pourrions ajouter un filtre passe-bas hardware.



## TP1G: Buzzer melody

Here we send a periodic signal to a buzzer, by varying the frequency we make different notes.

note: it resembles a PWM with a 50% ratio but with a varying frequency, whereas in a PWM the ratio is varying and the frequency is constant.

## TP1H : Temperature Sensor

The `getVoltage` function returns a float, that is then stored in the `temp` variable

```
int tempPin = 0;
void setup() {
    Serial.begin(9600);
}

void loop() {
    float temp = getVoltage(tempPin);
    temp = (temp-0.5)*100;
    Serial.println(temp);
```

```

    delay(500);
}

float getVoltage(int pin) {
    return(analogRead(pin) *.004882814);
}

```

en posant le doigt sur le capteur, on observe l'évolution sur le moniteur série

```

11:42:45.059 -> 24.22
11:42:45.574 -> 23.73
11:42:46.090 -> 24.22
11:42:46.559 -> 24.71
11:42:47.074 -> 25.20
11:42:47.590 -> 25.20
11:42:48.053 -> 25.68
11:42:48.561 -> 25.20
11:42:49.077 -> 26.17
11:42:49.592 -> 25.68
11:42:50.061 -> 25.68
11:42:50.576 -> 25.20
11:42:51.092 -> 25.68

```

## TP2A : Touchpad DS

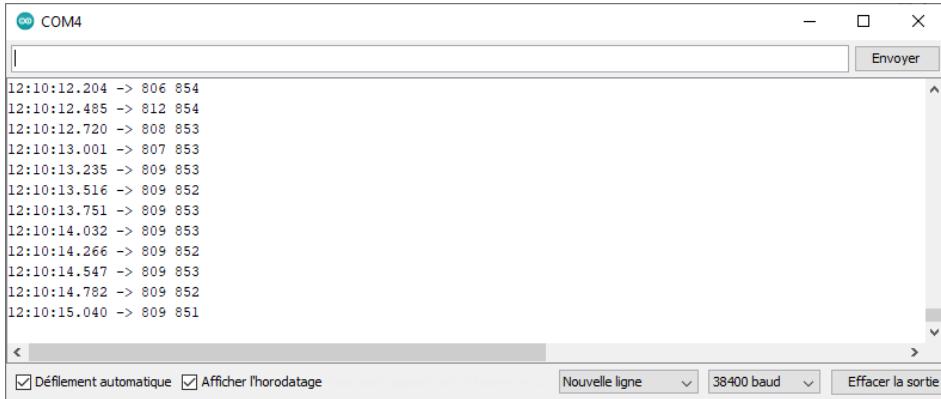
Here we code the reading of a touchpad, this is done by reading two tension divider, the two values are read on the analog ports and are scaled between 0 and 1000 (from 0 to 5V). in practice those values are never reach, but we see very well the variation from a corner to another.

for instance, this is the value when nothing is touched, and when we touch the bottom left corner:

```

12:09:45.498 -> 82 62
12:09:45.732 -> 82 68
12:09:46.013 -> 81 68
12:09:46.248 -> 80 67
12:09:46.521 -> 79 67
12:09:46.756 -> 79 66
12:09:47.037 -> 82 65
12:09:47.308 -> 82 66
12:09:47.522 -> 84 67
12:09:47.790 -> 80 68
12:09:48.071 -> 79 68
12:09:48.305 -> 80 67

```

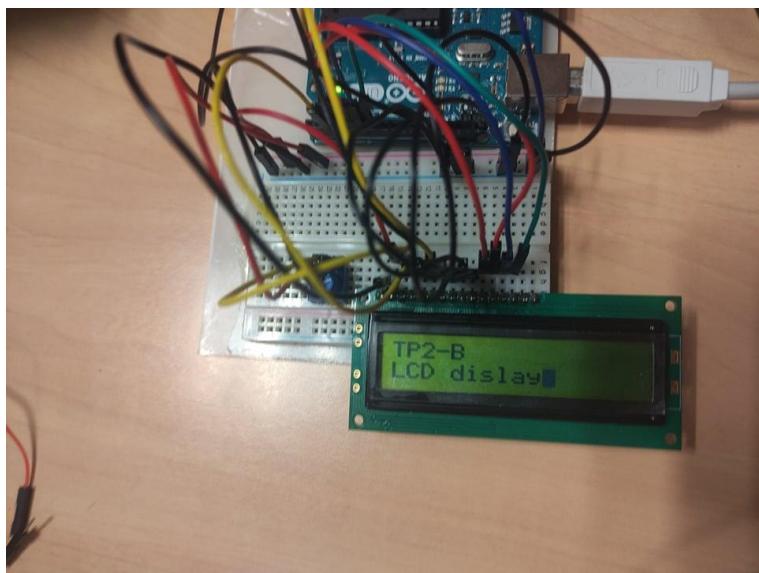


A screenshot of a terminal window titled "COM4". The window displays a series of timestamped messages in a monospaced font. The messages show time intervals and corresponding values. At the bottom of the window, there are several control buttons: "Défilement automatique" (checked), "Afficher l'horodatage" (checked), "Nouvelle ligne" (dropdown), "38400 baud" (dropdown), and "Effacer la sortie".

```
12:10:12.204 -> 806 854
12:10:12.485 -> 812 854
12:10:12.720 -> 808 853
12:10:13.001 -> 807 853
12:10:13.235 -> 809 853
12:10:13.511 -> 809 852
12:10:13.751 -> 809 853
12:10:14.032 -> 809 853
12:10:14.266 -> 809 852
12:10:14.547 -> 809 853
12:10:14.782 -> 809 852
12:10:15.040 -> 809 851
```

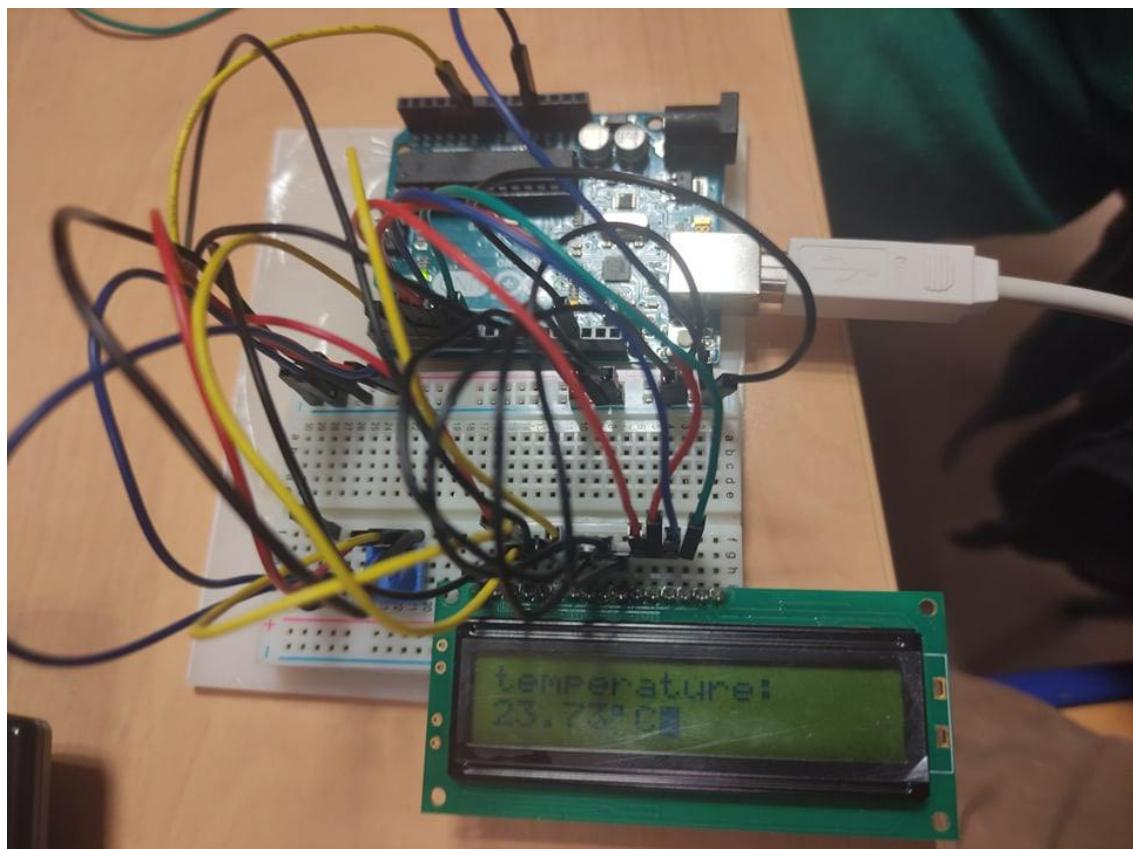
note, when we don't touch the screen a remaining tension is detected, we could correct that by fixing a threshold above the remaining value.

## TP2B,C,D : LCD



La fonction `blink` permet de faire clignoter le curseur, utile si on souhaite que l'utilisateur puisse taper des choses (pour se s'en servir comme terminal par exemple)

## TP2-B: Icd et température



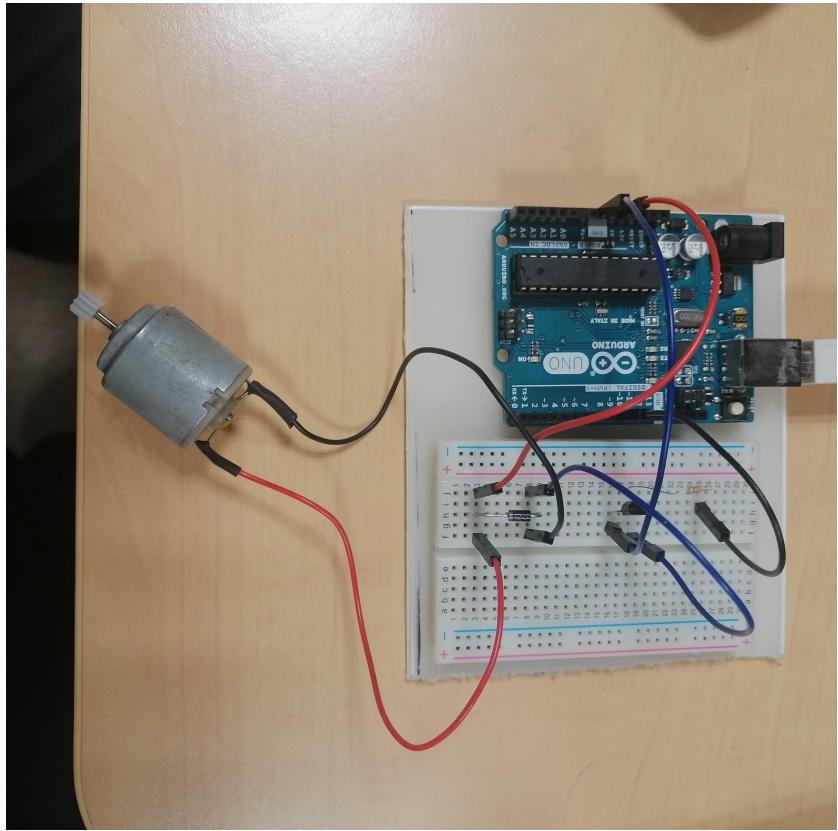
(NB : pour afficher le caractère “degré” : stocker dans une variable de type char la valeur en binaire du caractère et printer cette variable)

## TP2-C: lcd et touchpad

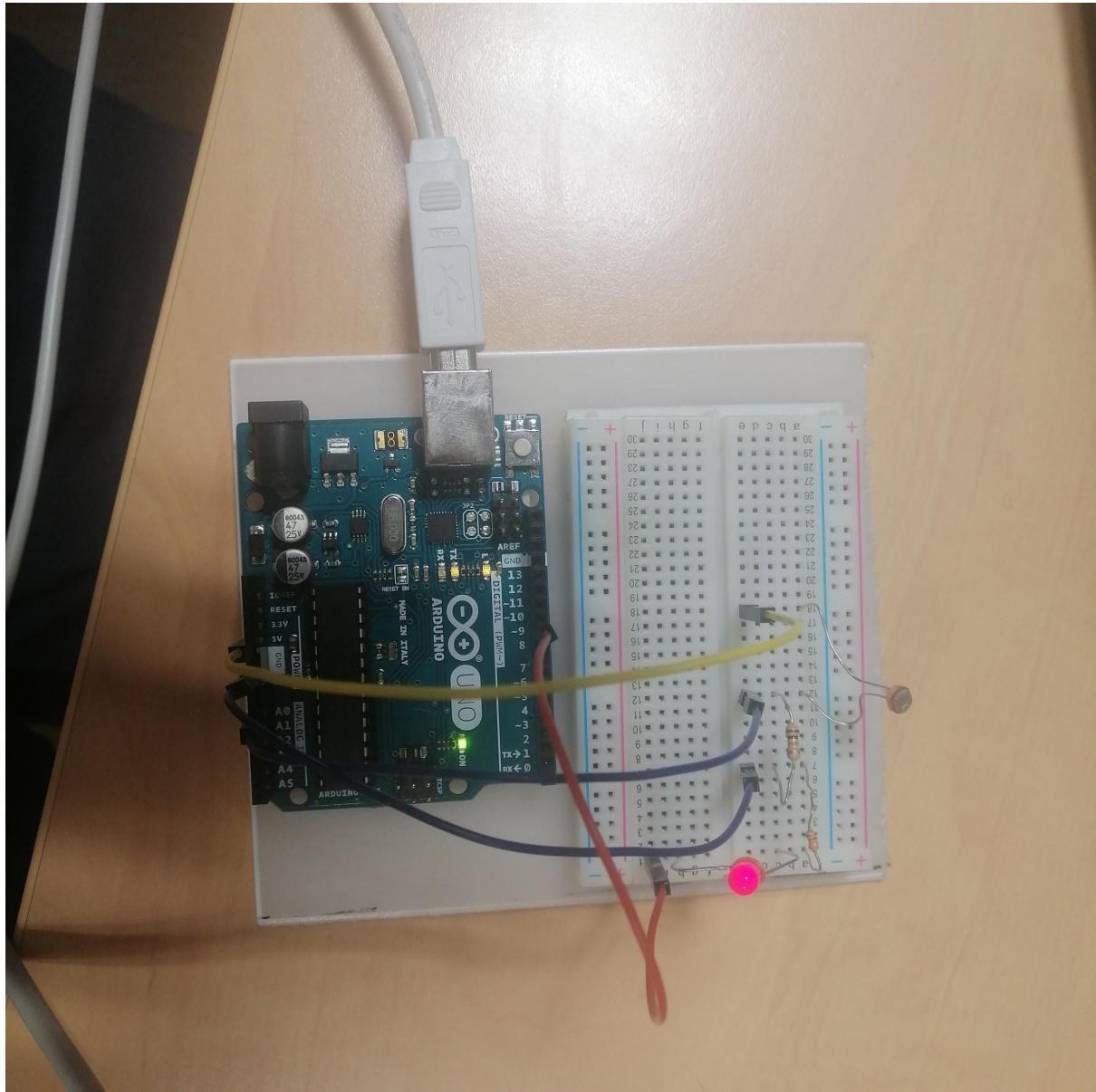


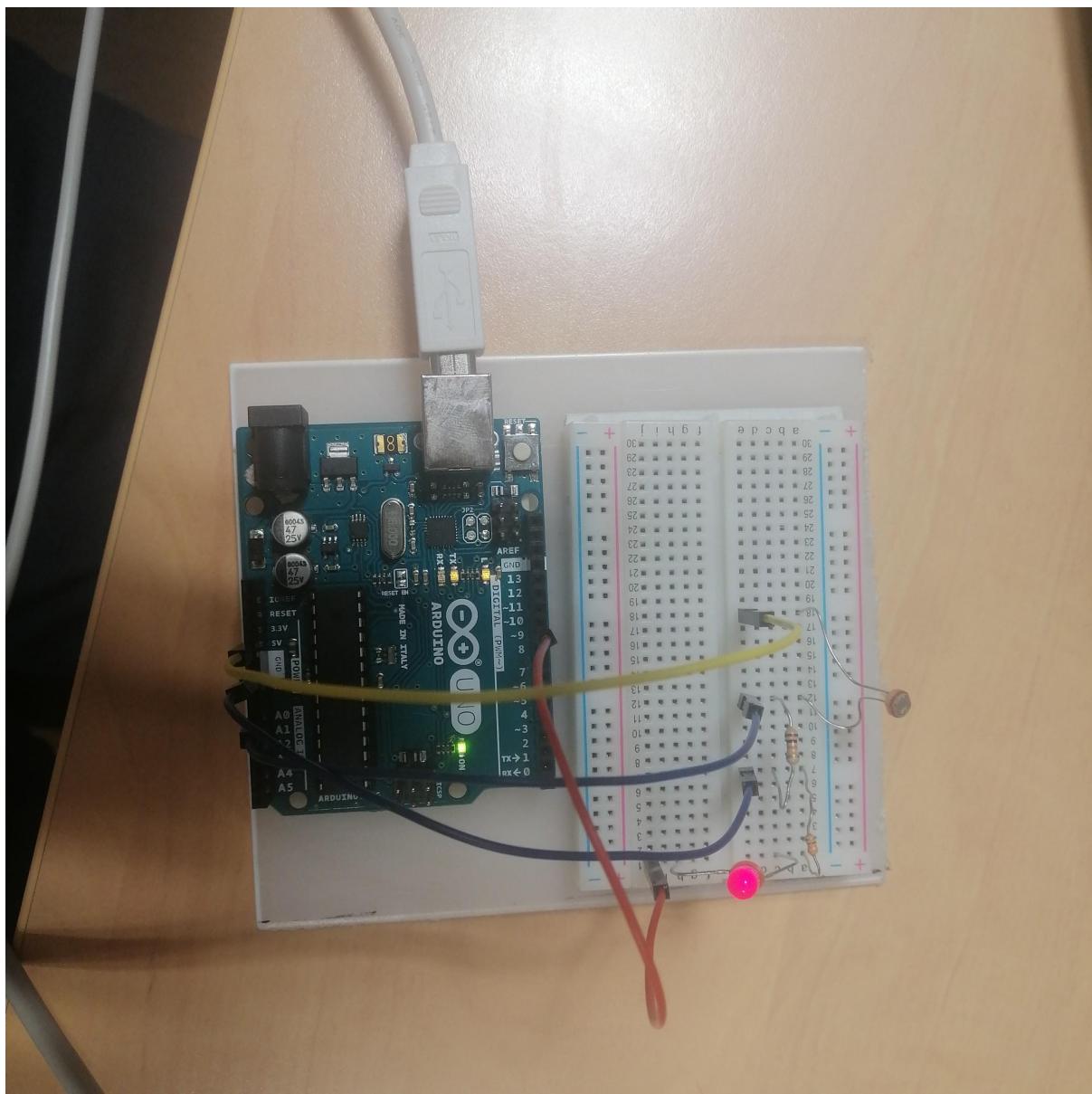
In this example we notice the use of the function `Lcd.scrollDisplayLeft`, which allow us to display scrolling text, useful when the message is longer than what the lcd screen can display at once.

## TP3A : Moteur DC

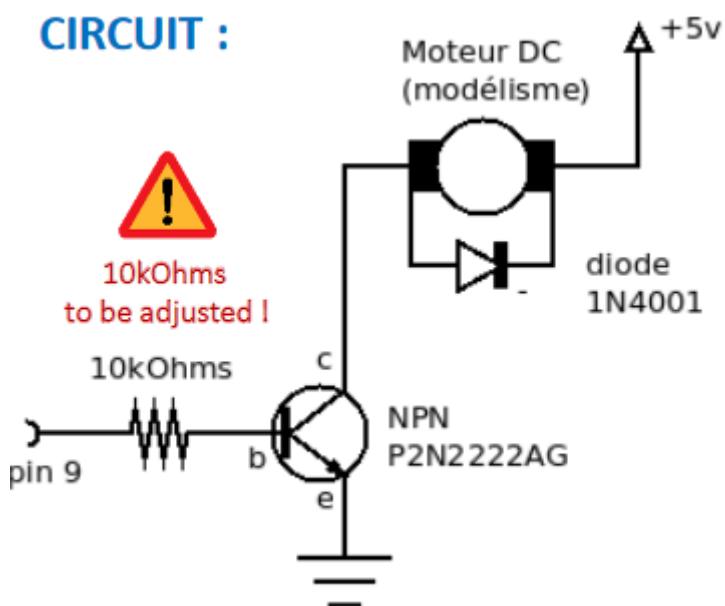


Lorsqu'aucun courant ne passe dans le moteur, pour que la tension n'explose pas, une diode est ajoutée au circuit.

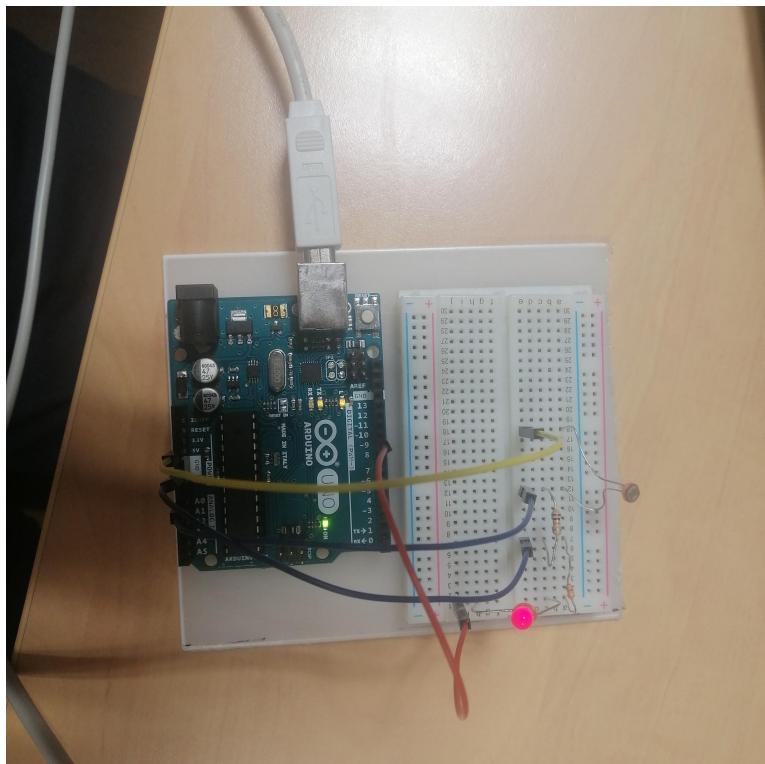
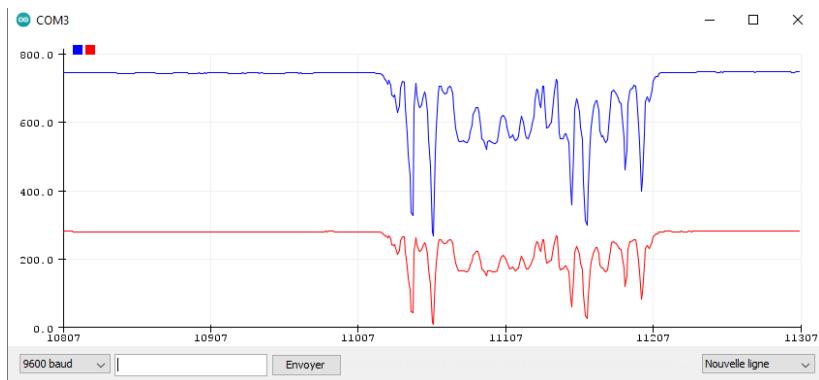




## CIRCUIT :



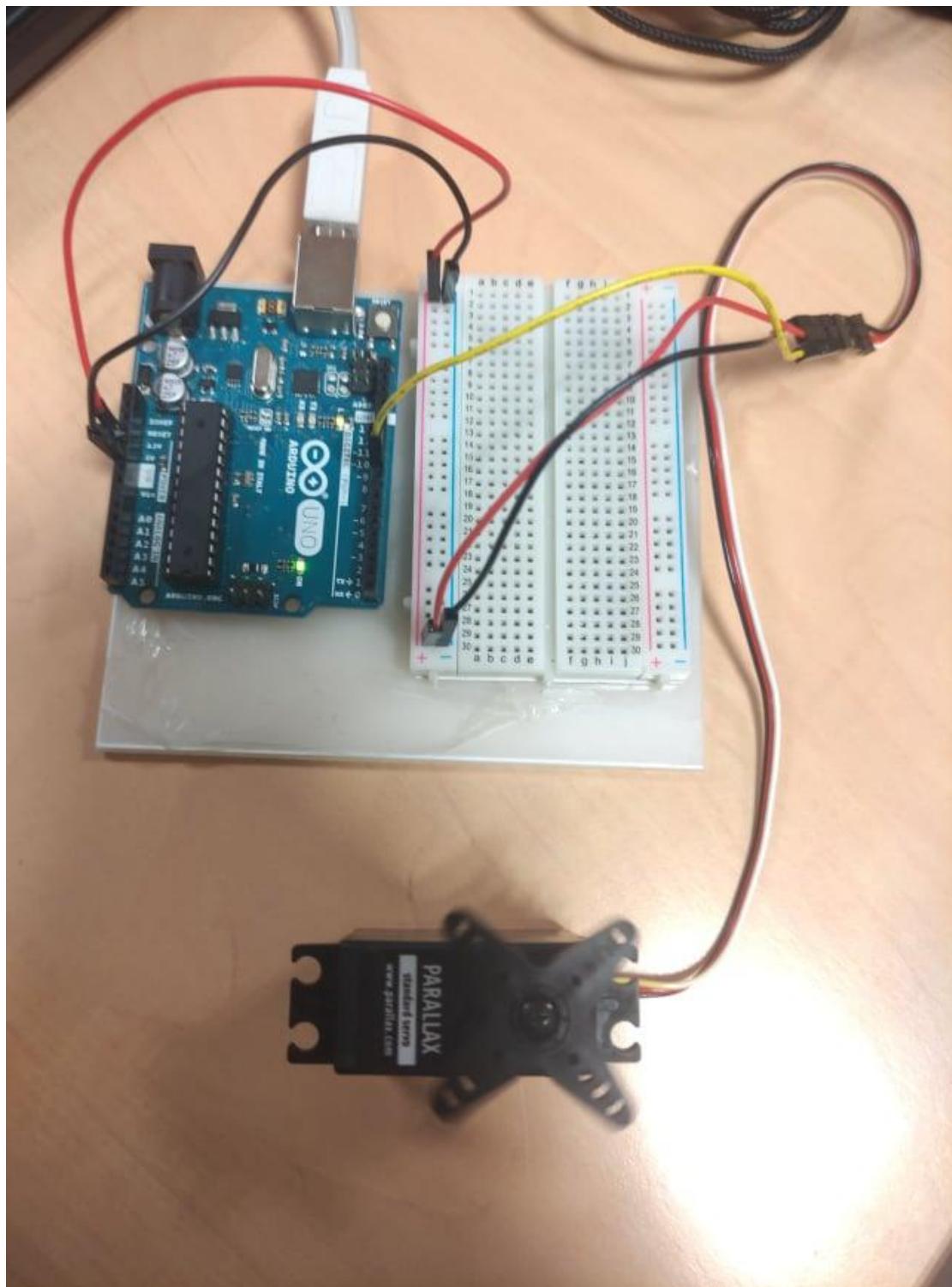
## TP3B : Photorésistance



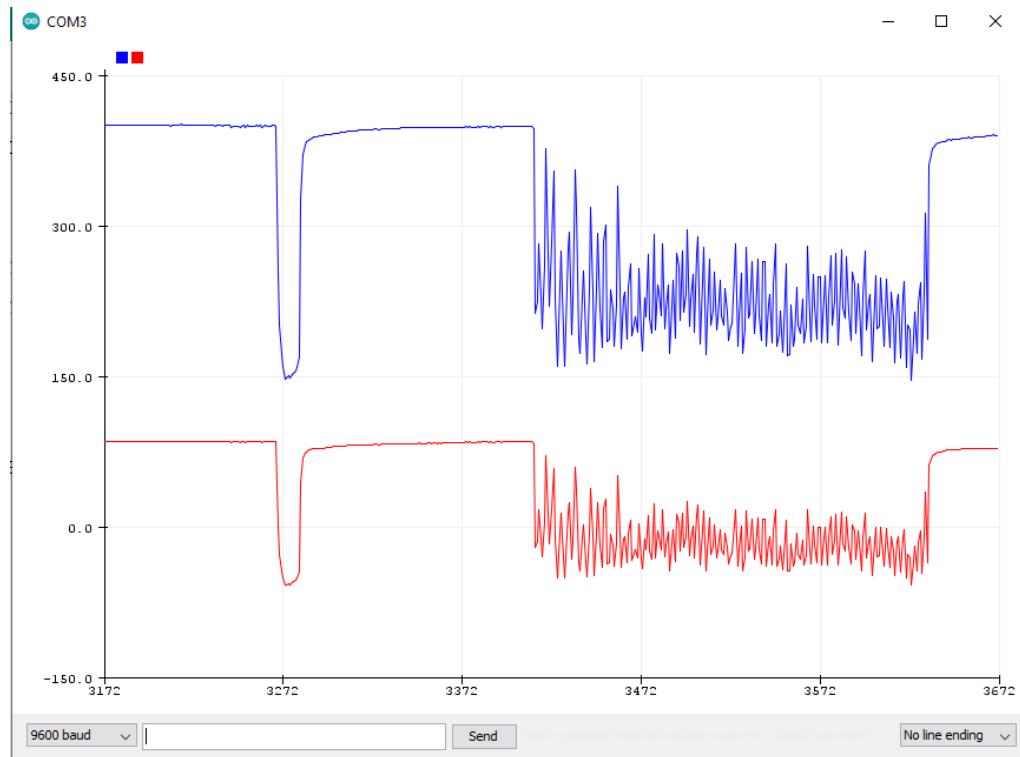
En approchant et en éloignant le doigt de la photorésistance, on peut moduler la tension reçue et la valeur de luminosité observée. Lorsque la luminosité mesurée baisse, la luminosité sortant de la diode baisse aussi.

## TP3C : servo-motor

here we use the Servo library, which is super easy and intuitive to use to use.



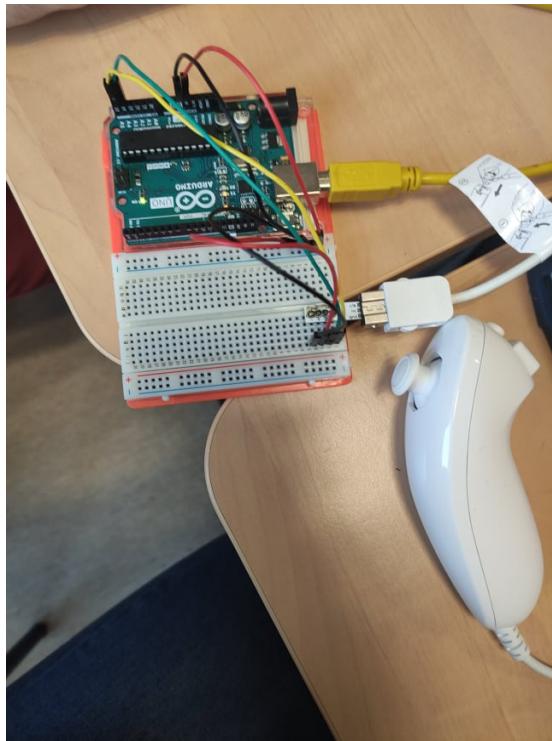
## TP3D : FLEX sensor



Un pont diviseur de tension permet de récupérer la tension aux bornes du capteur du flex sensor

## TP3F: Interruptions

# TP Nunchuck



```
COM6
Envoyer
08:51:27.503 -> 131 126 304 555 571 1 1
08:51:27.597 -> 131 126 304 555 571 1 1
08:51:27.689 -> 131 126 303 555 571 1 1
08:51:27.782 -> 131 126 303 555 572 1 1
08:51:27.922 -> 131 126 304 555 572 1 1
08:51:28.015 -> 131 126 304 555 571 1 1
08:51:28.108 -> 131 126 304 555 571 1 1
08:51:28.203 -> 131 126 303 555 571 1 1
08:51:28.296 -> 131 126 303 553 572 1 1
08:51:28.434 -> 131 126 304 555 571 1 1
08:51:28.525 -> 131 126 304 555 571 1 1
08:51:28.620 -> 130 126 303 555 571 1 1
08:51:28.713 -> 131 126 303 555 571 1 1
Nouvelle ligne 19200 baud Effacer la sortie
 Défilement automatique  Afficher l'horodatage
```

en raison de manque de temps, nous passons directement au TP7, node-red est un outils d'intérêt pour nous puisque nous l'utilisons dans d'autres projets

→ test capteur gaz

```
08:39:50.812 -> {"Capteur1":250}
08:39:51.829 -> {"Capteur1":251}
08:39:52.802 -> {"Capteur1":257}
08:39:53.821 -> {"Capteur1":245}
08:39:54.792 -> {"Capteur1":298}
08:39:55.807 -> {"Capteur1":782}
08:39:56.826 -> {"Capteur1":987}
08:39:57.793 -> {"Capteur1":977}
08:39:58.808 -> {"Capteur1":983}
08:39:59.825 -> {"Capteur1":892}
08:40:00.793 -> {"Capteur1":927}
08:40:01.804 -> {"Capteur1":850}
08:40:02.820 -> {"Capteur1":313}
08:40:03.828 -> {"Capteur1":272}
```

Nous constatons qu'effectivement la valeur mesurée par le capteur de gaz augmente quand on met le capteur en présence d'une vanne de briquet.