ECOLE NATIONALE
SUPÉRIEURE
D'INFORMATIQUE

الجـــمهوريـــة الجزائريـــة الديمـــقراطيـــة الشـعبيـة
وزارة التعليـــم العـــالي والبـــحث العلمـــي
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

**Intelligent and Communicating Systems, ICS**

$2^{nd}$ Year Specialty SIL G1

# Lab Report n°02

**Title:**

# Arduino Communications GPIO-Sensors-Actuators

**Studied by:**

**First Name:** Yasmine

**Last Name:** DINARI

**E_mail:** ly_dinari@esi.dz

# A. Theory

## 1. Pushbutton (digital input)

A pushbutton is a simple mechanical switch used to connect or disconnect an electrical circuit. When pressed, it allows current to flow through the circuit, and when released, it interrupts the flow. In microcontroller circuits like Arduino, pushbuttons are connected to GPIO pins configured as digital inputs.

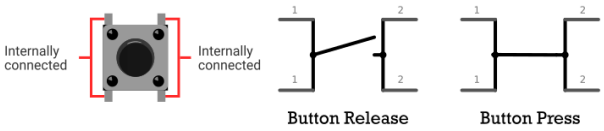


Figure 1: Pushbutton Functionning

## 2. De-bounce circuit

When a pushbutton is pressed or released, mechanical vibrations cause rapid fluctuations in the signal (bouncing). This can lead to multiple detections of a single press. To eliminate this effect: Hardware solution: Use an RC filter (resistor-capacitor circuit) to smooth out the signal. Software solution: Implement a delay or debounce logic in the code.

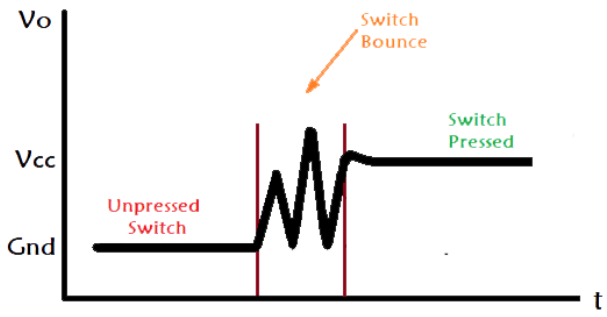


Figure 2: Bouncing Diagram

## 3. Pull-Up and Pull-Down Resistors

- A pull-up resistor connects the input pin to a high voltage (Vcc) to ensure a default HIGH state.

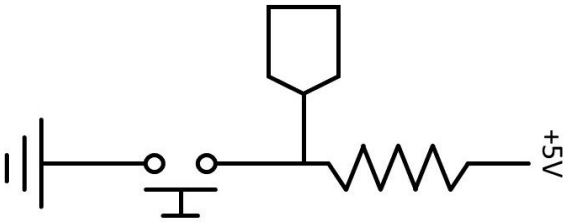


Figure 3: Pull-Up resistance circuit

- A pull-down resistor connects the input pin to ground (GND) to ensure a default LOW state.

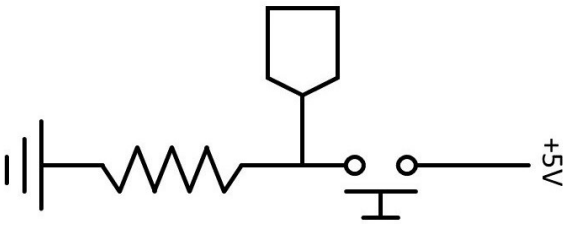


Figure 4: Pull-Down resistance circuit

- These resistors prevent the pin from being in a floating state (undefined voltage).

# 4. Introduction of the light sensor

A Light Dependent Resistor (LDR) is a passive sensor whose resistance decreases as light intensity increases. It is commonly used in applications like automatic lighting systems and exposure meters. A photodiode, in contrast, is an active sensor that generates a current when exposed to light (like a solar cell).

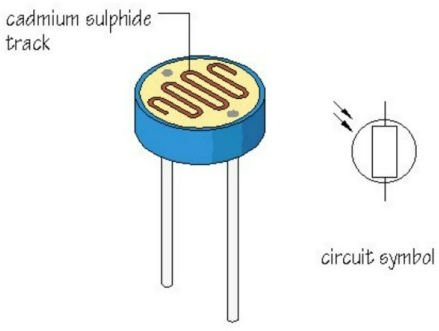


Figure 5: ldr

Figure 6: photodiode

# 5. How to use the Arduino serial console as a display

The Arduino Serial Monitor allows real-time communication between your Arduino and computer. It helps debug code and visualize sensor data using ‘Serial.print()‘ and ‘Serial.println()‘.

### Initialize Serial Communication

- In ‘setup()‘, start with:

```
1     Serial.begin(9600);
```

Listing 1: Set up communication

### Send Data to Serial Monitor

```
1     Serial.print("Value: ");
2     Serial.println(25);
```

Listing 2: Sending data

### Open Serial Monitor

- Click the Serial Monitor icon in the Arduino IDE.
- Set baud rate to match ‘Serial.begin()‘ (e.g., **9600**).

### View Real-Time Data

Monitor prints values continuously. Adjust baud rate settings if needed.

# 6. Arduino Analog GPIO Input Output

Analog GPIO pins on Arduino enable interaction with sensors that produce varying voltage levels. Unlike digital pins that recognize only HIGH and LOW states, analog inputs measure a range of values (0–1023).

```
1 const int pin = A0;
2 void setup()
3 {
4   Serial.begin(9600);
5 }
6 void loop()
7 {
```

```
8   lightVal = analogRead(pin);
9 }
```

Listing 3: Read Analog pin value

# B. ACTIVITY:

## 1. Connection of a light sensor (Analog)

This circuit demonstrates how to use an LDR (Light Dependent Resistor) with Arduino to measure light intensity by creating a voltage divider.

- Place the LDR on the breadboard.
- Connect one leg of the LDR to 5V on the Arduino.
- Connect the other leg of the LDR to one end of the 10 kΩ resistor.
- Connect the other end of the resistor to GND on the Arduino.
- Connect the junction between the LDR and the resistor to A0 on the Arduino (this is your analog input).]
- Use this code in your Arduino IDE:

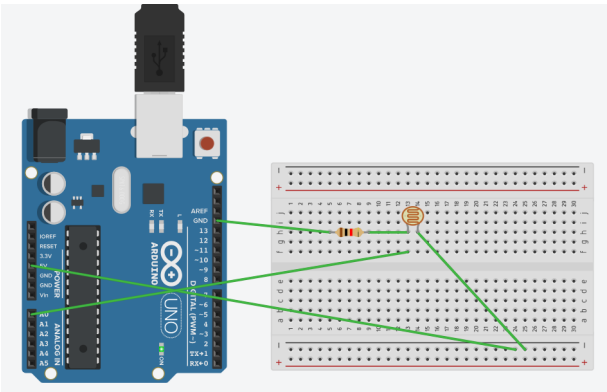

Figure 7: LDR Cicruit Simulation

```
const int LDR_pin = A0;
int lightVal ;
void setup()
{ Serial.begin(9600);}
void loop()
{
  lightVal = analogRead(LDR_pin);
  Serial.print("Light value is :");
  Serial.println(lightVal);
  delay(1000);

}
```

Listing 4: Simulation of LDR Circuit Code

- Open the Serial Monitor in the Arduino IDE or TinkerCad (Tools > Serial Monitor).
- Observe how the light values change as you vary light intensity over the LDR (e.g., by covering it with your hand or shining a flashlight).


Figure 8: LDR Cicruit on Arduino IDE

## 2. Connection of a Push Button (Digital)

This circuit demonstrates how to use a push button with Arduino to detect button presses and control an LED.

- Place the push button on the breadboard.
- Connect one leg of the push button to digital pin 8 on the Arduino.
- Connect the opposite leg of the push button to GND on the Arduino.
- Add a pull-up resistor by enabling the internal pull-up in the Arduino code.

```
void setup()
{
    pinMode(8, INPUT);
    pinMode(13, OUTPUT);
    Serial.begin(9600);
}
void loop()
{
    int value = digitalRead(8);
    Serial.println(value);
    if (value == 0)
    {
        Serial.println("pressed");
```

```
    }
    else
    {
        Serial.println("released");
    }
    delay(100);
    digitalWrite(13, value);
}
```

Listing 5: Simulation of Push Button Circuit Code

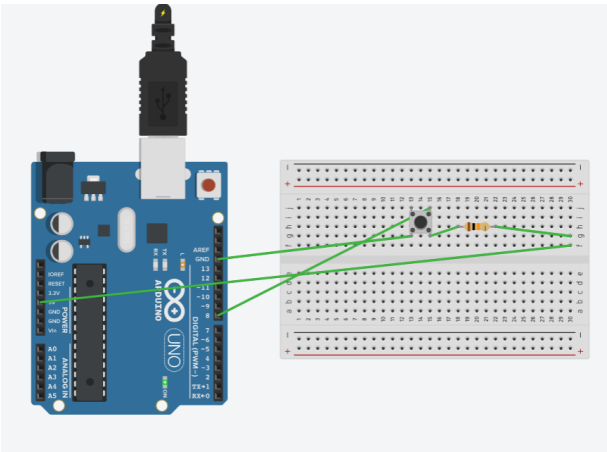

Figure 9: Push Button Circuit Simulation

- Open the Serial Monitor in the Arduino IDE (Tools > Serial Monitor).
- Observe how the serial output changes between "pressed" and "released" as you press and release the button.

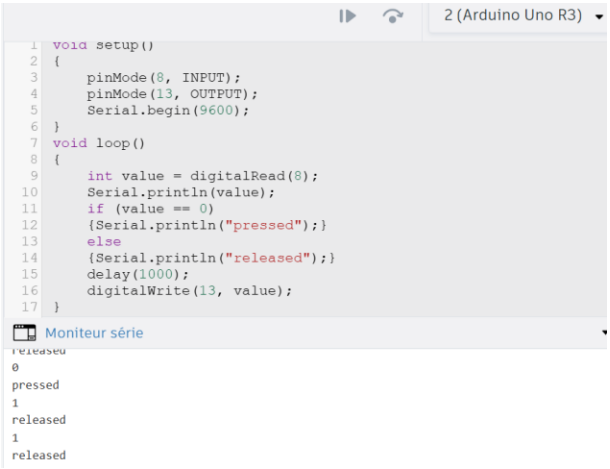


Figure 10: Push Button Circuit Simulation Output

# 3. Connection of a De-bounce Circuit (Capacitor)

This section demonstrates how to connect a de-bounce circuit using a capacitor to stabilize push button signals and eliminate noise caused by rapid switching.

- Add a capacitor across the two legs of the push button.
- Place the push button on the breadboard and connect it as before:
  - One leg to digital pin 8 on the Arduino.
  - The opposite leg to GND on the Arduino.
- Test the circuit again using the same code provided for the push button.

## 3.1. Is the De-bounce Circuit Necessary?

A de-bounce circuit is necessary to avoid false readings caused by mechanical noise when pressing or releasing the button. Without a capacitor, rapid toggling may result in inconsistent or multiple readings for a single press.

## 3.2. Example Showing the Problem

- Without a capacitor: The Serial Monitor may show multiple ”pressed” or ”released” messages for a single button press due to bouncing.
- With a capacitor: The signal becomes stable, and each press results in only one ”pressed” or ”released” message.

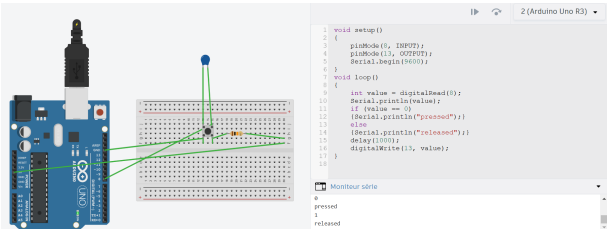


Figure 11: Push Button Behavior with De-bounce Circuit

# 4. Using an LED with LDR and Push Button

This section demonstrates how to combine an LDR and a push button with an LED in an Arduino circuit to control the LED based on light intensity and button presses.

- Connect the LDR as described in the ”Connection of a Light Sensor” section.

- Connect the push button as described in the "Connection of a Push Button" section.
- Connect an LED to digital pin 13 on the Arduino, with its cathode connected to GND (via a resistor if necessary).
- Modify the code to include logic for controlling the LED based on both light intensity and button state.

```
const int LDR_pin = A0;
const int button_pin = 8;
const int LED_pin = 13;

void setup() {
    pinMode(button_pin, INPUT);
    pinMode(LED_pin, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    int lightVal = analogRead(LDR_pin);
    int buttonState = digitalRead(button_pin);

    Serial.print("Light value: ");
    Serial.println(lightVal);

    if (buttonState == LOW && lightVal < 500) {
        digitalWrite(LED_pin, HIGH); // Turn on LED if button is pressed
   and light is dim
        Serial.println("LED ON");
    } else {
        digitalWrite(LED_pin, LOW); // Turn off LED otherwise
        Serial.println("LED OFF");
    }

    delay(100);
}
```

Listing 6: Simulation Code for LED Control Using LDR and Push Button

- Open the Serial Monitor in the Arduino IDE (Tools > Serial Monitor).
- Observe how the LED behavior changes based on light levels detected by the LDR and button presses.

# 5. Imagine a scheme for a smart control of streetlights for public use

A smart streetlight system could use LDR sensors and Arduino microcontrollers to optimize urban lighting. LDRs would detect ambient light, automating on/off cycles. Motion sensors could adjust brightness based on pedestrian or vehicle presence. Network

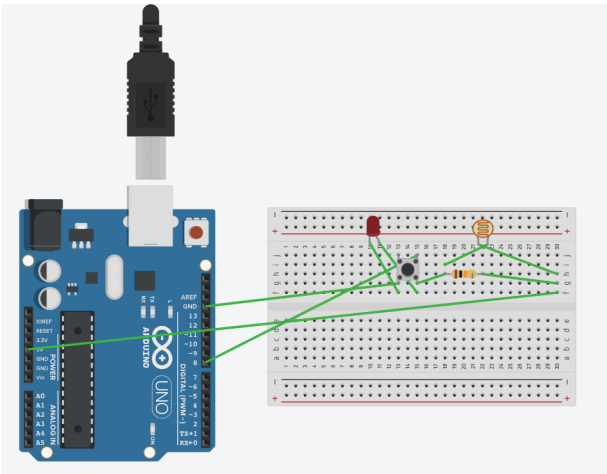

Figure 12: Circuit Behavior with LDR, Push Button, and LED

connectivity would enable remote control and monitoring, allowing real-time adjustments for events or weather conditions. This system would reduce energy consumption while enhancing public safety and urban responsiveness.

# 6. CONCLUSION

These Arduino experiments have provided a solid foundation in electronics and embedded programming. We've learned to work with analog and digital inputs/outputs, essential skills for creating interactive systems. The practical applications, like the smart streetlight concept, demonstrate how these basic components can solve real-world problems. This hands-on experience bridges theory and practice, preparing us for more complex projects in computer science and electronics. As we move forward, these skills will be invaluable in developing innovative solutions and tackling future technological challenges.