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 SIQ G02, 2CS SIQ2

LAB report n°02

Title:

Arduino Communications
GPIO-Sensors-Actuators)

Studied by:

First Name: Nour el Imane
Last Name: HEDDADJI
E-mail: jn_heddadji@esi.dz

A. Theory

1. Pushbutton

1.1. The use and connection of a pushbutton

In Arduino, a pushbutton is an input component that initiates actions when pressed and returns to its initial state when released. To connect a pushbutton with a digital input :

1. **Step 1:** Insert the pushbutton into the breadboard,Ensure that the button's legs are inserted into different rows, with two legs on each side of the gap.
2. **Step 2:** Use a jumper wire to connect one leg of the push button to the GND (ground) pin on the Arduino board.
3. **Step 3:** Use a wire to link the other leg of the pushbutton to a digital input pin on the Arduino board.

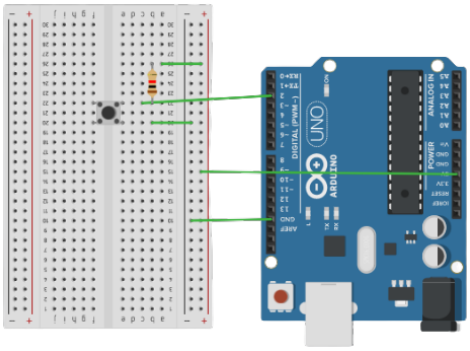


Figure 1: Connecting push-button

1.2. De-bounce circuit

Bouncing is a noise or oscillations that can occur in a pushbutton switch when it is pressed or released. A de-bounce circuit, consisting of a capacitor and resistor connected in parallel with a push button, filters out rapid signal changes caused by contact bouncing. This provides a more stable and accurate signal to the Arduino's digital input pin.

1.3. Pull-Up and Pull-Down resistors

To ensure a stable voltage level at a digital pin input, you can use pull-up and pull-down resistors.

1. **Pull-up :** A pull-up resistor is connected between the digital pin and a higher voltage level (such as Vcc).
2. **Pull-down Resistor :** A pull-down resistor is connected between the input pin and ground.

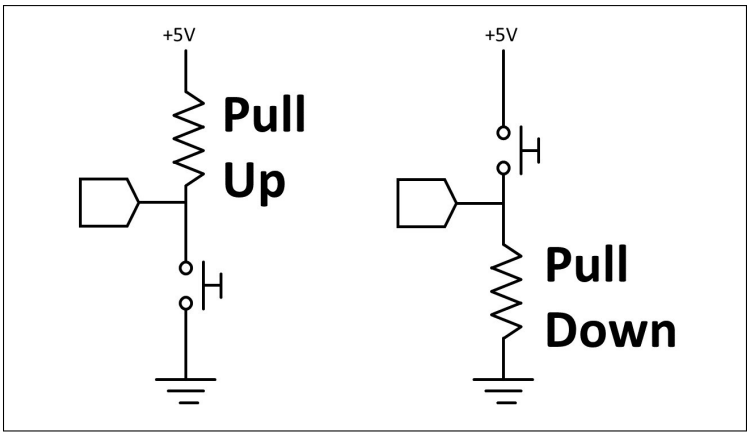


Figure 2: Pull-Up vs Pull-Down resistors

2. Connection of a light sensor (Analog GPIO)

2.1. Introduction of the light sensor

A light sensor is a device that detects the presence, absence, or intensity of light in a given environment.

2.2. Using Arduino serial console as a display

1. In the setup function, use **Serial.begin(9600)** to set up the serial connection.
2. In the loop function, use **Serial.print()** to send data to the serial console.
3. After uploading your sketch, open the serial console by selecting«Serial Monitor» from the «Tools» menu.

2.3. Configuring GPIO line as an analog output

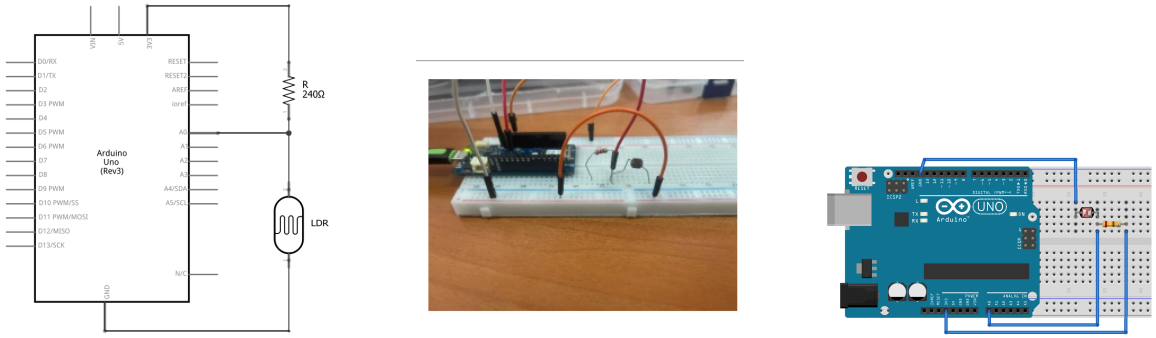
1. In the setup function, use the **pinMode()** function to configure the GPIO pin as an output.
2. In the loop function, use the **analogWrite()** function to set the output voltage.

B. Activity

1. Connection of a light sensor (analog pin)

Hardware

An arduino MKR1010 board is connected to a light sensor (LDR) and a 240Ω resistor:



(a) Schematic view (b) electronics mounting (c) Circuit view

Figure 3: LDR connection electronics views

Software

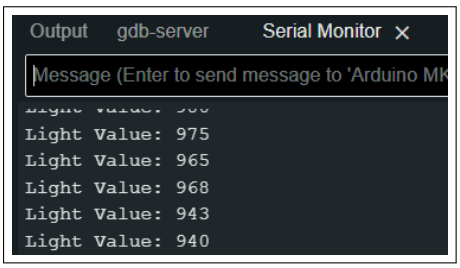
In the setup() function, we configure a serial port for communication and we set the analog pin sensorPin as an input pin by using the pinMode() function. In the loop() function every 1 second:

```
int ldr = A3;   //assigning ldr pin A3.
int out = 7;   //assigning out pin 7.
 void setup()
 {
   pinMode(out, OUTPUT);  //setting pinmode as output
   pinMode(ldr, INPUT);  //setting pinmode as input
   Serial.begin(9600); //initialize serial connection
 }
 void loop()
 {
   int sensorValue = analogRead(ldr);
   Serial.println(sensorValue); // print out the value you read
   delay(1000);  // delay in between reads for stability
 }
```

Listing 1: LDR connection Arduino program

Analysis

The serial console display values (between 0 and 1023) which are proportional to the light amount absorbed by the LDR sensor.



(d) console output

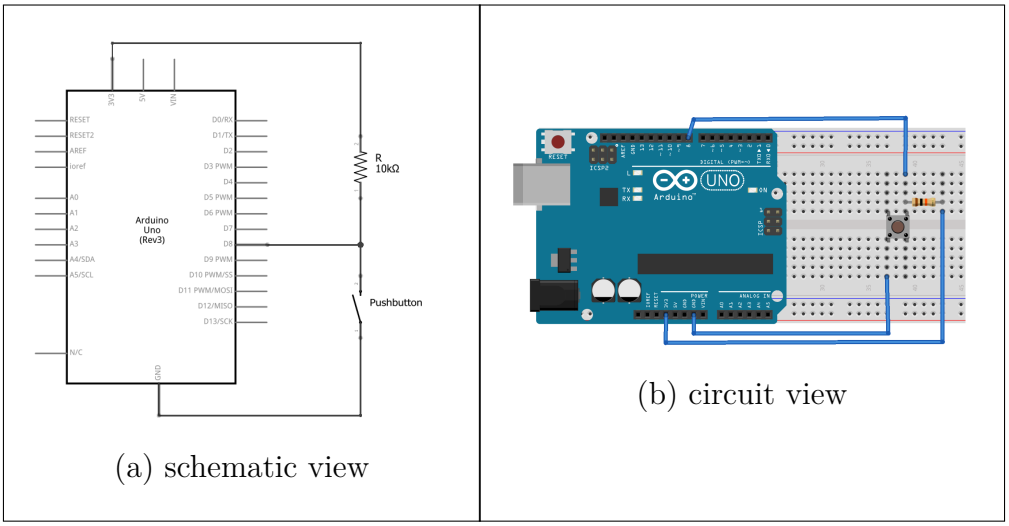
Figure 4: LDR connection electronics experience

2. Connection of a pushbutton (digital pin)

2.1. Pull-Up resistor

Hardware

An arduino MKR1010 board is connected to a pushbutton and a 10kΩ pull-up resistor:



(a) schematic view

(b) circuit view

Figure 5: Pull-Up resistor electronics views

Software

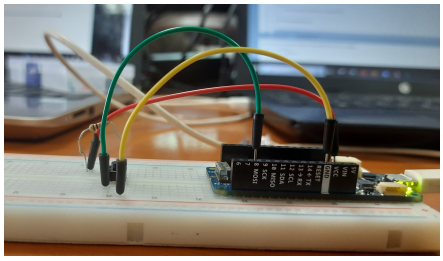
In the *setup()* function, we configure a serial port for communication and set the 'BUTTON_PIN'as an input using *pinMode()*.
In the *loop()* function, we check if the button state has changed, and if so, we print whether the button is pressed or not.

```
1 // initialize a constant for the digital pin input
2 #define BUTTON_PIN 8
3  int previous=0;
```

```arduino
void setup()
{
  Serial.begin(9600);  // initialize the serial connection
  pinMode(BUTTON_PIN, INPUT);  // configure the digital pin as input
}
void loop()
{
  // read the binary logic voltage between the pin and the ground
  byte buttonState = digitalRead(BUTTON_PIN);
  //  display the button state only if it had changed
  if(previous!=buttonState)
  {
    // when the button is pressed, the state is LOW
    if (buttonState == LOW)
      {Serial.println("Button is pressed");}
    // when the button is not pressed, the state is HIGH
    else
      {Serial.println("Button is not pressed");}
    previous=buttonState; // change the state
  }
}
```

Listing 2: Pull-Up resistor Arduino program

Analysis



(c) electronics mounting

```
Button released
Button pressed
Button pressed
Button pressed
```

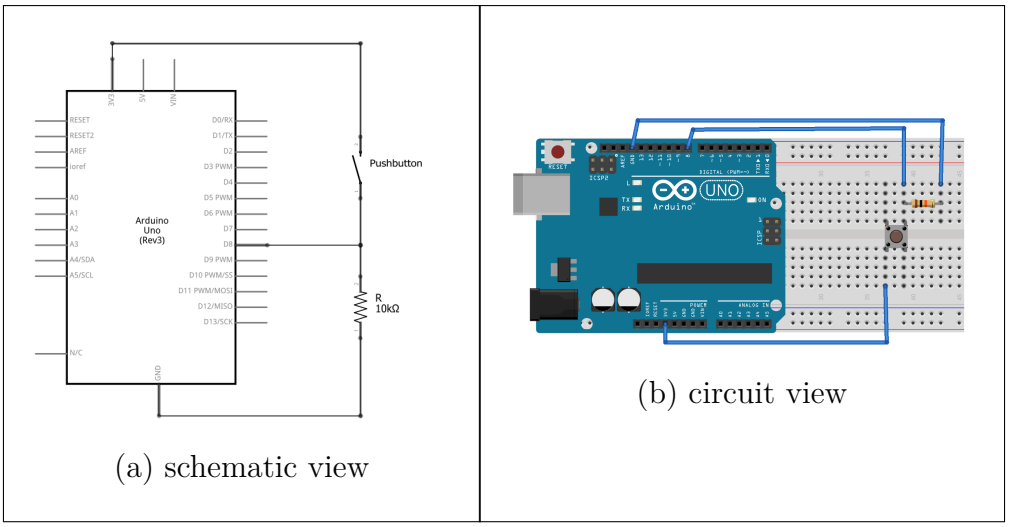
(d) console output

Figure 6: Pull-Up resistor electronics experience

2.2. Pull-Down resistor

Hardware

An arduino MKR1010 board is connected to a pushbutton and a 10kΩ pull-down resistor:



(a) schematic view

(b) circuit view

Figure 7: Pull-Down resistor electronics views

Software

The code below shows only the loop() function:

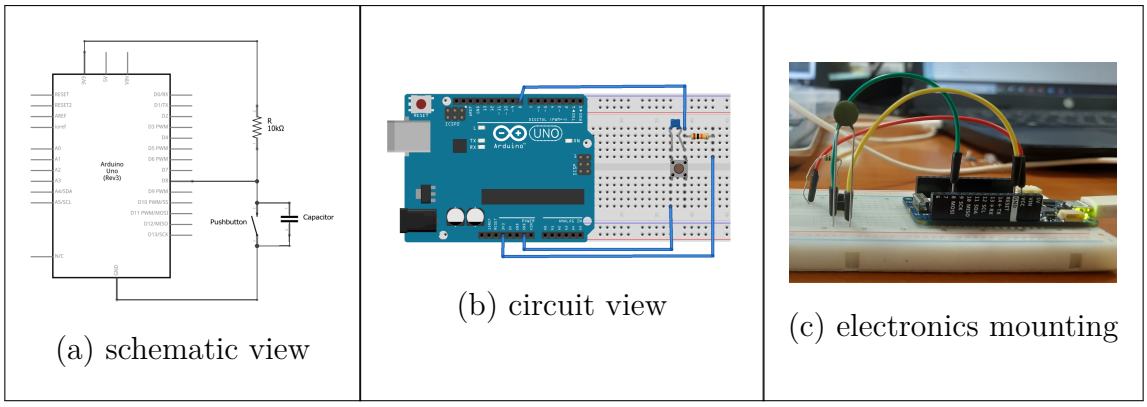
```
const int BUTTON_PIN = 2; // Change to the desired pin number
byte previous = LOW; // Initialize the previous state
void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Configure the pin as an input with
    a pull-up resistor
  Serial.begin(9600); // Initialize the serial communication
}
void loop() {
  byte buttonState = digitalRead(BUTTON_PIN); // Read the button state
  if (previous != buttonState) {
    if (buttonState == HIGH) {
      Serial.println("Button is pressed");
    } else {
      Serial.println("Button is not pressed");
    }
    previous = buttonState; // Change the state
  }
}
```

Listing 3: Pull-Down resistor Arduino program

2.3. De-bounce circuit

Hardware

A capacitor is connected between the pushbutton in the previous circuit:



(a) schematic view

(b) circuit view

(c) electronics mounting

Figure 8: De-bounce circuit electronics views

Software

The program used here is the same that the one used with the Pull-Up resistor configuration.

Analysis

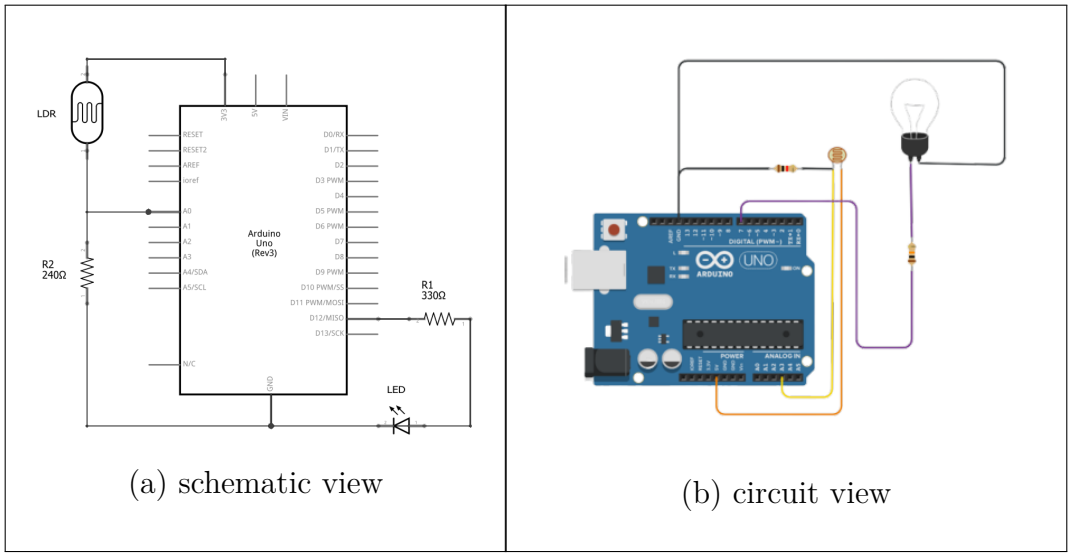
When the de-bounce circuit is used, the signal read from the digital input is clean and consistent with the real pushbutton state.

3. Using an LED with LDR and Pushbutton

3.1. From a light threshold

Hardware

Two circuits are connected with the Arduino board, one for the LED and another . The figure below shows the electronics details:



(a) schematic view

(b) circuit view

Figure 9: LED with LDR electronics views

Software

```
int sensorPin = A0; // select the analog input pin for LDR
int LEDpin = 12; // select the digital output pin for LED
void setup()
{
  Serial.begin(9600);  // initialize the serial connection
  pinMode(LEDpin,OUTPUT);  // set the digital pin for the LED as output
  pinMode(sensorPin,INPUT);  // set the digital pin for the LED as input
}
void loop()
{
  int valeur = analogRead(sensorPin);  // read tha LDR value
  Serial.println(valeur); //prints the LDR values on the screen
  delay(1); // 1 second wait
  // turn on the LED if the sensor value is <= 30
  if (valeur <= 30 ) {
    digitalWrite(12, HIGH);
    Serial.println("ON");
  }
  // turn of the LED if the sensor value is > 30
  else {
    digitalWrite(12, LOW);
    Serial.println("OFF");
  }
}
```

Listing 4: LED with LDR Arduino program

Analysis

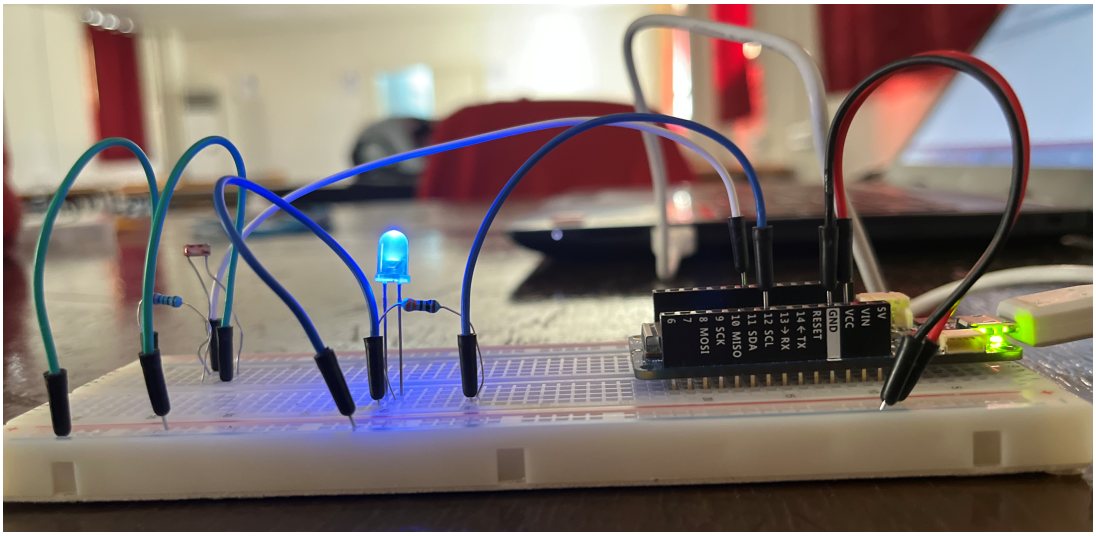
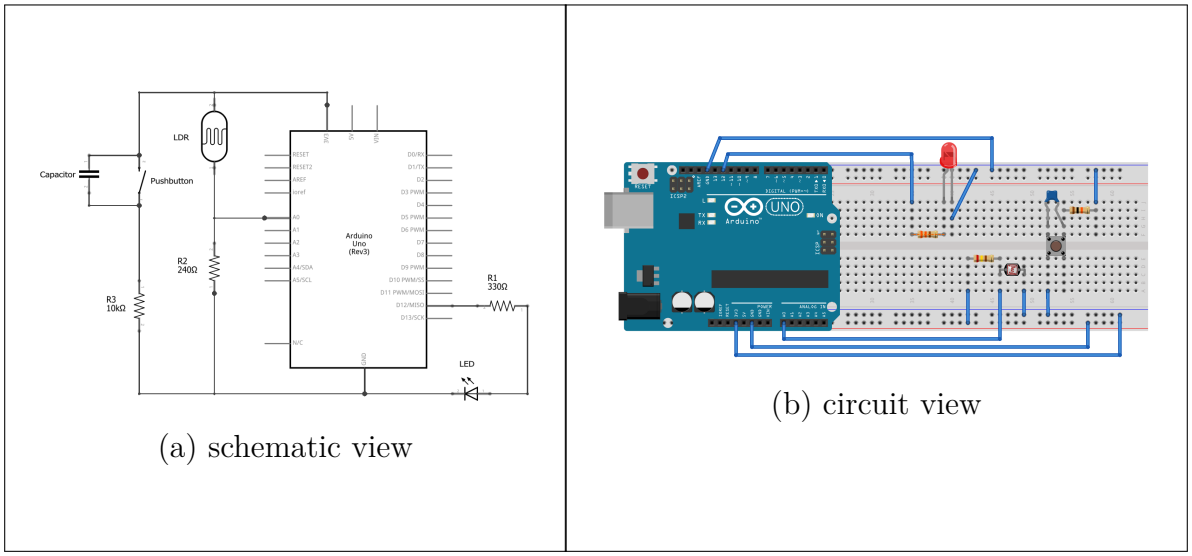


Figure 10: LED with LDR electronics experience

When the light intensity is low, the LED truns on. When the light intensity is high, the LED turns off.

3.2. By pressing the button

Hardware



(a) schematic view

(b) circuit view

Figure 11: LED with LDR and pushbutton electronics views

Software

```
int sensorPin = A0; // select the analog input pin for LDR
int LEDpin = 12; // select the digital output pin for LED
int buttonPin = 8; // select the digital input pin for the pushbutton
// the system state indicates whether the LED is auto-controled or not
int state = 0;
void setup()
{
  Serial.begin(9600);  // initialize the serial connection
  pinMode(LEDpin,OUTPUT);  // set the digital pin for the LED as output
  pinMode(sensorPin,INPUT);  // set the digital pin for the LED as input
  pinMode(buttonPin,INPUT);  // set the digital pin for pushbutton as input
}
void loop()
{
  //if the button is pressed, then change the system state
  if(digitalRead(8) == HIGH){
    Serial.println(state); //prints the system state value
    // change the state
    if(state==1)
      state=0;
    else
      state=1;
  }
  // if the light control is activated, then call the function
  if (state == 1)
      {lightControl();}
  delay(50);  // short delay
}
// the function that control the LED based on the light intensity
void lightControl(){
  int valeur = analogRead(A6); // read tha analog value from the sensor
```

```
32   // turn on the LED if the sensor value is <= 30
33   if (valeur <= 30 ) {
34     digitalWrite(12, HIGH);
35     Serial.println("ON");}
36   // turn of the LED if the sensor value is > 30
37   else {
38     digitalWrite(12, LOW);
39     Serial.println("OFF"); }
40 }
```

Listing 5: LED with LDR and pushbutton Arduino program

We configure BUTTONPIN and sensorPin as inputs and LEDpin as the only output. In the loop(), we check the push button state. If it's pressed, we toggle the system state and call lightControl.

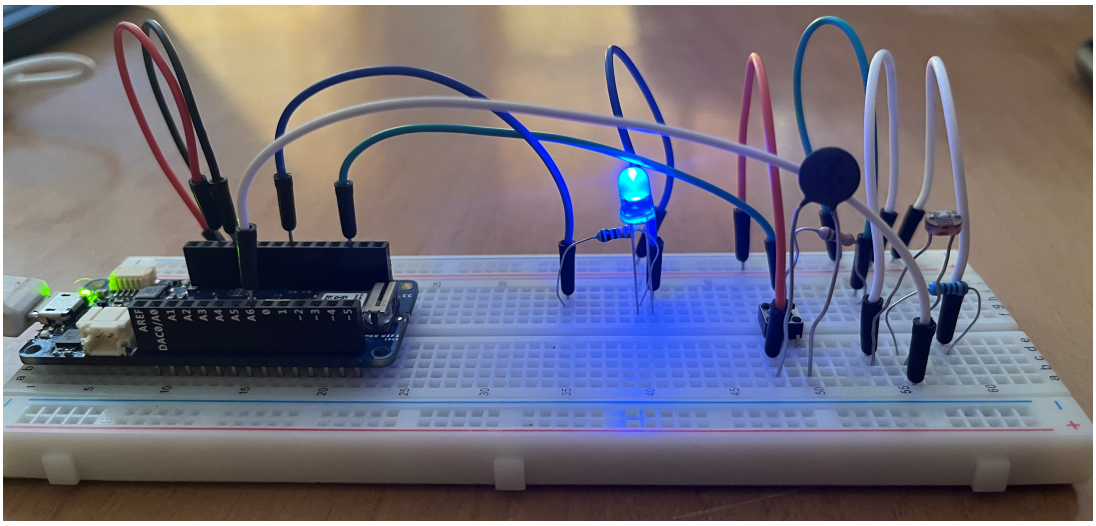
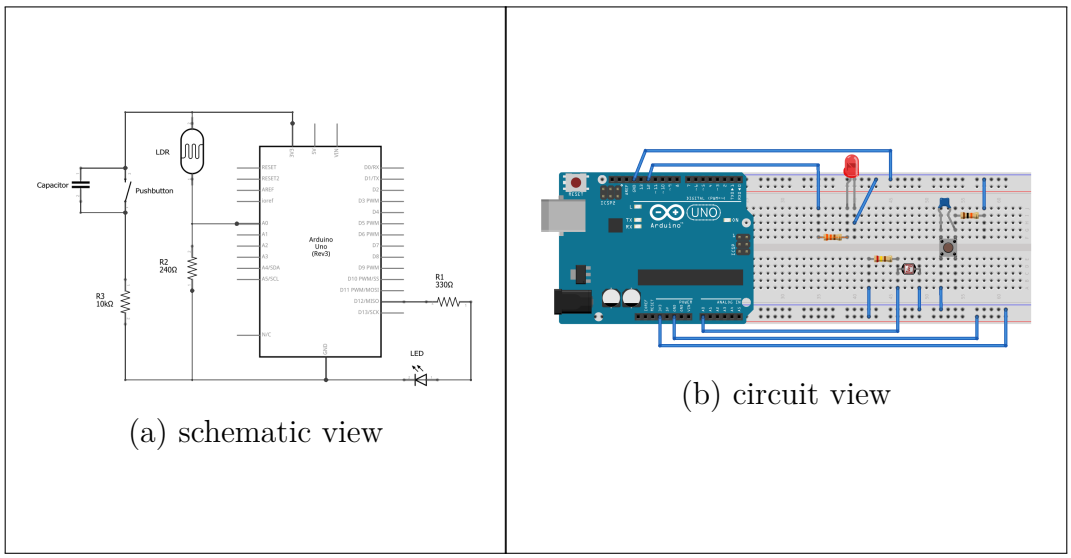


Figure 12: LED with LDR and pushbutton electronics experience

3.3. A scheme for a smart control of streetlights for public use.



(a) schematic view

(b) circuit view

Figure 13: LED with LDR electronics views

4. Conclusion

In conclusion, push buttons are common components in Arduino circuits. To prevent issues, we can use a de-bounce circuit with the button or address the problem through software. Additionally, we use resistors, such as pull-up and pull-down resistors, to set default states for signals. The key difference lies in the default state.

Light-dependent resistors (LDRs) are passive light sensors widely employed in various applications. By integrating LDR sensors with pull-up or pull-down resistors and LEDs, we can create a smart streetlight control system for public use.