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

Report

Title:

SIMULATION :
LED-PushButton-debounce-Pullup

Studied by:

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

1. Introduction

In this tutorial, we will explore the simulation of two Arduino circuits. The first circuit controls the brightness of a LED based on ambient light using an LDR (Light Dependent Resistor). The second circuit enables manual control of the LED with a push button, incorporating a debounce circuit for stability.

2. Tools

1. Tinkercad: A user-friendly and interactive simulation platform.
2. Proteus: Offering a detailed and realistic simulation environment.
3. Arduino IDE: Required for writing and uploading Arduino code.

3. Components

1. LED
2. LDR (Light Dependent Resistor)
3. PushButton
4. 220 and 10k ohm resistors
5. Arduino Uno
6. Breadboard
7. Wires

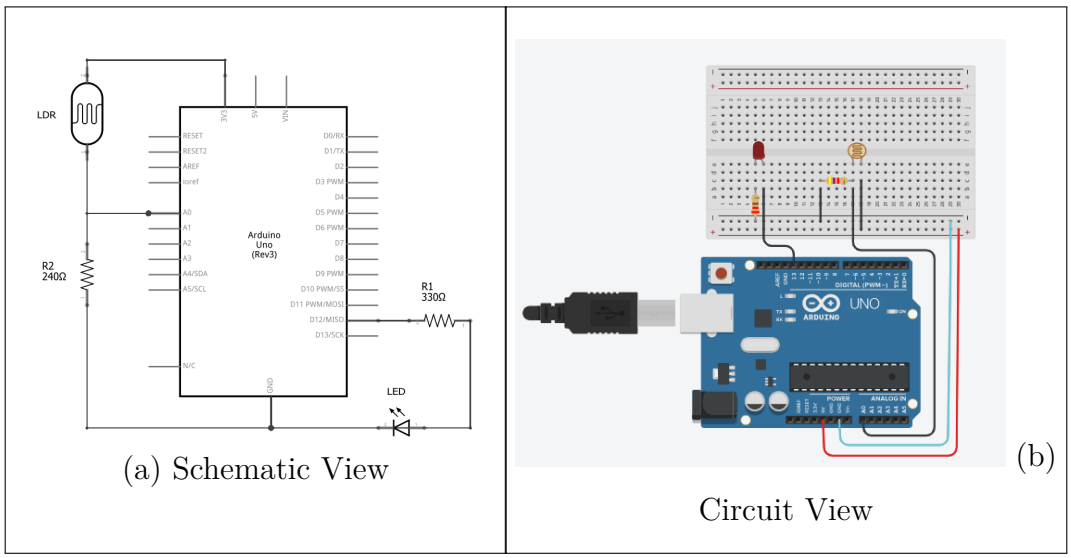
LED Control with LDR

1. Tinkercad Simulation

1. **Step 1: Log in to Tinkercad**
   - i Go to Tinkercad.
   - ii Log in to your Tinkercad account or create one if you don't have an account.
2. **Step 2: Create a New Project** After logging in, click on "Create New Circuit."
3. **Step 3: Add Components** In the Tinkercad workspace, open the "Components" panel. Search for and add the following components to your project:
   - a Arduino Uno.
   - b Breadboard.
   - c LDR (Light Dependent Resistor).
   - d LED.
   - e Wires.
4. **Step 4: Build the Circuit** Position the components on the breadboard as follows:
   1. Connect one leg of the LDR to one end of the breadboard (e.g., Row 1).
   2. Connect the other leg of the LDR connected to a 10k ohm resistor to a nearby row on the breadboard.
   3. Insert the LED into the breadboard with its longer leg (anode) connected to a 220 ohm resistor.
   4. Connect the other leg of the resistor to a row on the breadboard.
   5. Connect the short leg (cathode) of the LED to the ground (GND) on the Arduino.
   6. Connect one end of a jumper wire to the same row as the LDR's other leg.
   7. Connect the other end of the jumper wire to an analog pin on the Arduino (e.g., A0).
   8. Connect another jumper wire from the same row as the resistor to one of the digital pins (e.g., D9) on the Arduino.

1. **Step 5: Write and Upload Arduino Code** Write the Arduino code (see Listing 1.1) in the in the Tinkercad workspace.
2. **Step 6: Simulate** Run the simulation in Tinkercad and observe how the LED responds to changes in light conditions detected by the LDR.

Hardware and Software



(a) Schematic View (b) Circuit View

Figure 1.1: LED with LDR Electronics Views

```arduino
// Define the pins for the light sensor and LED
const int lightSensorPin = A0;
const int ledPin = 12;
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int lightValue = analogRead(lightSensorPin);
  Serial.println(lightValue);
  int threshold = 100;
  if (lightValue < threshold) {
    digitalWrite(ledPin, HIGH); // Turn on the LED
  } else {
    digitalWrite(ledPin, LOW);  // Turn off the LED
  }
}
```

Listing 1.1: LED with LDR Arduino Program

Code Explanation

i *const int lightSensorPin = A0* and *const int ledPin = 12*: Declare variables for the LDR pin and LED pin.

ii *int lightValue = analogRead(lightSensorPin)*: Read the LDR value and store it in the *lightValue* variable.

iii *Serial.println(lightValue)*: Print the LDR value to the serial monitor.

iv *int threshold = 100*: Set a threshold value.

v *if (lightValue < threshold)*: Check if the LDR value is lower than the threshold.

vi *digitalWrite(ledPin, HIGH)*: Turn on the LED.

vii *digitalWrite(ledPin, LOW)*: Turn off the LED.

Analysis

When the light detected by the LDR falls below a certain threshold (100 in this case), the LED turns on. The code provides a basic example of using analog input, digital output, and conditional statements to control an LED based on environmental conditions.

2. Proteus Simulation

1. **Step 1: Launch Proteus**

   Open the Proteus Design Suite software.

2. **Step 2: Create a New Project**

   i After launching Proteus, click on "File" > "New Project."

   ii Name your project and choose a directory to save it.

1. **Step 3: Add Components** In the Proteus workspace, click on "Pick from Libraries." Search for and add the following components to your project:

   1. Arduino Uno.
   2. LDR (Light Dependent Resistor).
   3. LED.
   4. Resistors.
   5. Ground symbol (for connecting to the Arduino ground).
   6. Power supply (for providing voltage).
   7. Virtual terminal.

1. **Step 4: Build the Circuit**

   1. Position the components in your Proteus workspace.
   2. Connect the components following the same connections as in your Tinkercad simulation, ensuring the LDR is connected to an analog pin on the Arduino and the LED to a digital pin.
   3. From the sidebar, find and add a "Virtual Terminal" component to your project. Place it in a convenient location.
   4. Connect the TXD pin of the Arduino Uno with the RXD pin of the Virtual Terminal and the TXD with the RXD.

2. **Step 5: Write and Upload Arduino Code**
    1. Open the Arduino IDE on your computer. Write the same code provided in the Tinkercad simulation section for Arduino code that controls the LDR and LED.
    2. Once you've written the code, compile and upload it to the Arduino Uno in the Proteus workspace.
3. **Step 6: Simulate** Click "Run Simulation" to execute your project. Observe how the LED responds to changes in light conditions detected by the LDR.

Hardware and Software

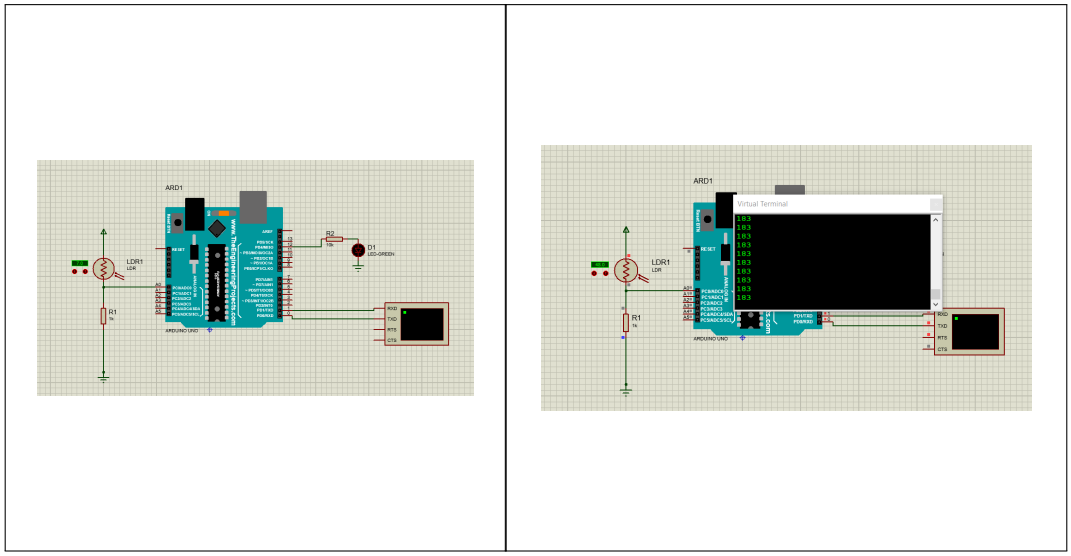


Figure 1.2: LED with LDR Electronics Views

Debounce Circuit

1. Tinkercad Simulation

1. **Step 1: Log in to Tinkercad**
   i Go to Tinkercad.
   ii Log in to your Tinkercad account or create one if you don't have an account.
2. **Step 2: Create a New Project** After logging in, click on "Create New Circuit."
3. **Step 3: Add Components** In the Tinkercad workspace, open the "Components" panel. Search for and add the following components to your project:
   a Arduino Uno.
   b Breadboard.
   c Capacitor (10uF).
   d LED.
   e Wires.
   f 220 and 10k ohm resistors.
4. **Step 4: Build the Circuit** Position the components on the breadboard as follows:
   1. Connect one leg of the push button to digital pin 7 on the Arduino.
   2. Connect the other leg of the push button to the 5V pin on the Arduino.
   3. Connect a 10kΩ resistor between digital pin 7 and the ground (GND) on the Arduino.
   4. Insert the LED into the breadboard with its longer leg (anode) connected to a 220 ohm resistor.
   5. Connect a 10uF capacitor between the two legs of the push button.
   6. Connect the other leg of the resistor to a row on the breadboard.
   7. Connect the short leg (cathode) of the LED to the ground (GND) on the Arduino.
   8. Connect one end of a jumper wire to the same row as the LDR's other leg.
   9. Connect the other end of the jumper wire to an analog pin on the Arduino (e.g., A0).
   10. Connect another jumper wire from the same row as the resistor to one of the digital pins (e.g., D9) on the Arduino.

1. **Step 5: Write and Upload Arduino Code** Write the Arduino code (see Fig 2.1) in the in the Tinkercad workspace.
2. **Step 6: Simulate** Run the simulation in Tinkercad and observe how the LED responds to changes in light conditions detected by the LDR.

Hardware and Software

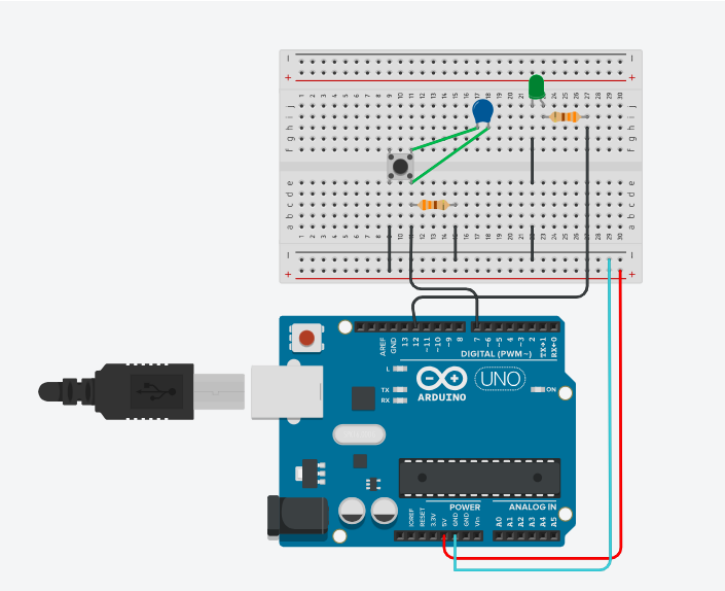


Figure 2.1: Debounce circuit simulation in Tinkercad

```arduino
const int LED = 12;
const int BUTTON = 7;
int val = 0;
int old_val = 0;
int state = 0;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT);
}

void loop() {
  val = digitalRead(BUTTON);
  if ((val == HIGH) && (old_val == LOW)) {
    state = 1 - state;
    delay(10);
  }
  old_val = val;
  if (state == 1) {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

Listing 2.1: LED with LDR Arduino Program

Code Explanation

1. *const int LED = 12* and *const int BUTTON = 7*:

Define pin constants for the LED and push button.

2. `pinMode(LED, OUTPUT)`: Set LED pin as output.
3. `pinMode(BUTTON, INPUT)`: Set push button pin as input.
4. *if ((val == HIGH) (old_val == LOW))*: Check for button press.
5. `state = 1 - state`: Toggle LED state with a delay.

Code Analysis

Running the simulation highlights the code's effective approach to addressing button debouncing, a common issue in electronics.

Button debouncing is crucial for ensuring that a single physical button press results in a single change in the LED's state.

2. Proteus Simulation

1. **Step 1: Open Proteus**
    1. Launch the Proteus software.
    2. Create a new project or open an existing one.

1. **Step 2: Add Components**
    i In the Proteus workspace, open the "Library" panel.
    ii Search for and add the following components to your project:
    - Arduino Uno.
    - Push button.
    - Capacitor (10uF).
    - Resistors (10kΩ and 220Ω).
    - LED.
    - Wires.
    - Oscilloscope.

2. **Step 3: Build the Circuit** Ensure that the components are connected as follows:
    1. Connect one leg of the push button to digital pin 7 on the Arduino.
    2. Connect the other leg of the push button to the 5V pin on the Arduino.
    3. Insert a 10kΩ resistor between digital pin 7 and the ground (GND) on the Arduino.
    4. Attach a 10uF capacitor between the two legs of the push button.
    5. Connect the anode (longer leg) of the LED to a 220Ω resistor.

6. Connect the other end of the resistor to a row on the breadboard.
7. Connect the short leg (cathode) of the LED to the ground (GND) on the Arduino.
8. Use wires to connect one end of a jumper wire to the same row as the push button's other leg.
9. Connect the other end of the jumper wire to an analog pin on the Arduino (e.g., A0).
10. Connect another jumper wire from the same row as the resistor to one of the digital pins (e.g., D9) on the Arduino.
11. Add an oscilloscope and connect it to the digital pin where the button signal is observed to monitor the signal changes.

1. **Step 4: Write and Upload Arduino Code** Write the same Arduino code (see Listing 1.1) used in the Tinkercad workspace. Upload this code to the Arduino in the Proteus project.

2. **Step 5: Simulate and Observe** Run the simulation in Proteus and observe how the LED responds. Additionally, use the oscilloscope to monitor the effect of the debounce circuit on the button signal. This allows you to visualize the noise-filtering capability of the debounce circuit.

Hardware

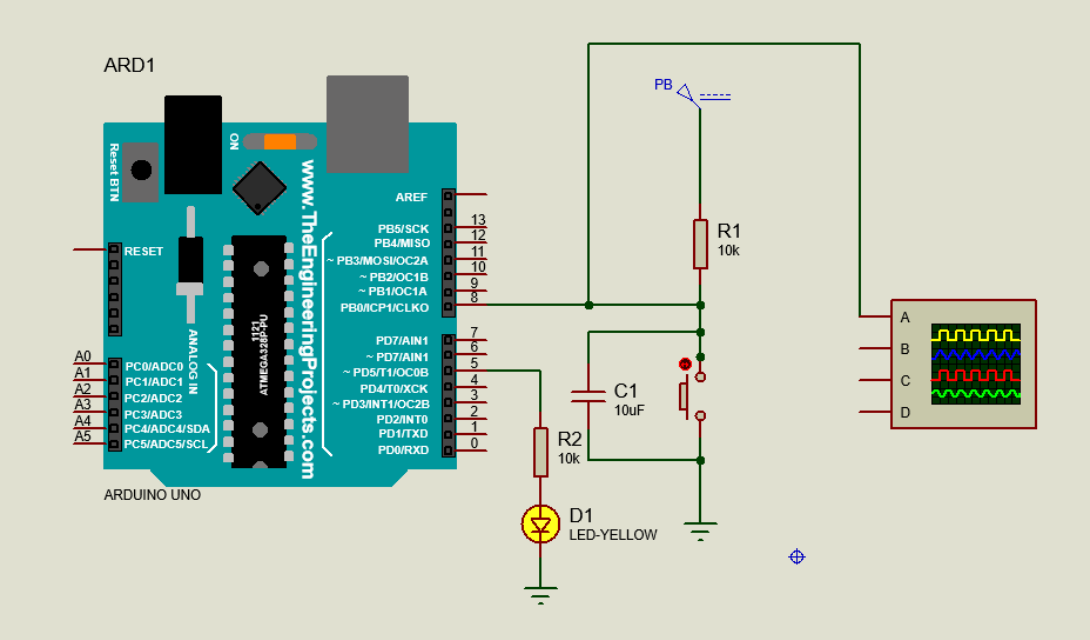
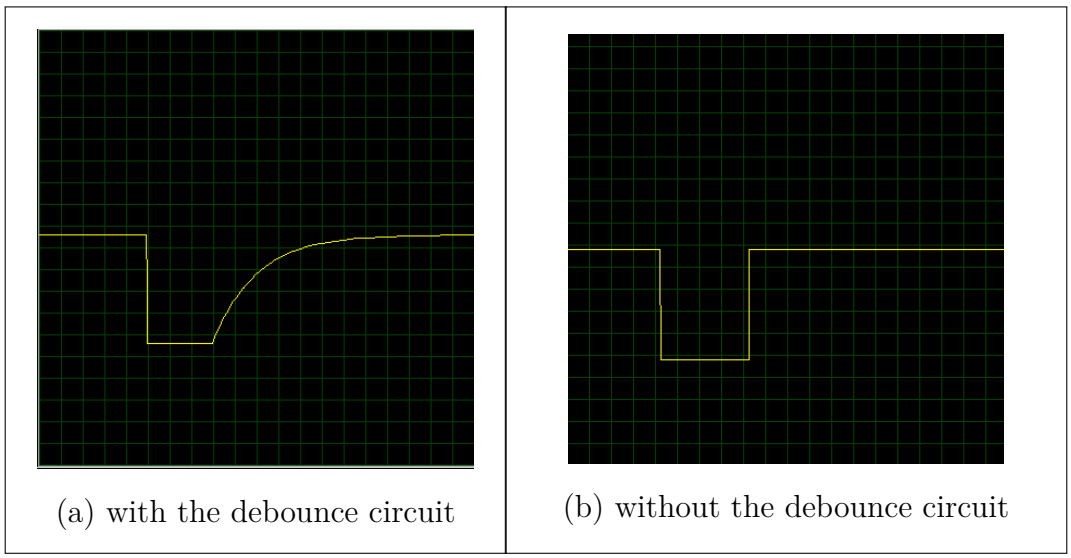


Figure 2.2: Debounce circuit simulation in Proteus

2.1. Observations



(a) with the debounce circuit

(b) without the debounce circuit

Figure 2.3: Simulation results

Image (a) on the oscilloscope displays voltage changes with a debounce circuit. When the button is pressed, the voltage drops to LOW and gradually rises to HIGH as the capacitor recharges.

In contrast, Image (b), without debounce, shows rapid voltage fluctuations immediately after the button press, resulting in a noisy signal that can lead to multiple erroneous states. A debounce circuit is essential to mitigate this issue.