



Model used to generate the content: Pro Search

The link of the conversation : <https://www.perplexity.ai/search/make-an-arduino-code-using-the-DzYv7xFAQJ.YrSFSARLpOg#0>

Make an arduino code using the photoresistor sensor. Base the code in the instructions below and I also attached here the lesson regarding the arduino serial connection that will be used in the code.

Objectives:

- Understand and implement Arduino Serial Connection
- Utilize and familiarize students on different basic Serial connection functions
- Create a simple circuit that can be controlled using Serial connection

Instructions:

Using the diagram and code base used for Laboratory Activity 3, fulfill the following:

1. Choose one (1) sensor to be used in this activity
2. If using thermistor, set the temperature threshold to 50 Celsius. If using photoresistor, set the brightness threshold to 220.
3. If the threshold is met, blink the LED in pin 8 (delay = 100) AND SHOULD continue to blinking even after the sensor reading is below threshold.
4. If the word "stop" is entered in the Serial Monitor, the blinking should stop. The word should be case insensitive (can enter stop in different cases)

Here is an **Arduino code** built according to your instructions and based on the *Arduino Serial Connection Guide* you attached. It uses a **photoresistor (LDR)** sensor, employs **serial communication** functions such as `Serial.begin()`, `Serial.print()`, `Serial.available()`, and `Serial.readStringUntil()`, and fulfills the required behavior from your objectives.

Arduino Code: Photoresistor with Serial-Controlled LED

```
// Laboratory Activity: Serial-Controlled LED using Photoresistor
// Objective: Use Serial Monitor commands to control LED blinking based on brightness
// threshold

const int ldrPin = A0;      // Photoresistor connected to analog pin A0
const int ledPin = 8;       // LED connected to digital pin 8
const int threshold = 220; // Brightness threshold

int sensorValue = 0;
bool isBlinking = false;   // Flag to track if LED should continue blinking

void setup() {
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600); // Initialize Serial communication
    Serial.println("Photoresistor Serial Control initialized.");
    Serial.println("Type 'stop' in Serial Monitor to stop blinking.");
}

void loop() {
    sensorValue = analogRead(ldrPin);

    // Display the sensor reading in the Serial Monitor
    Serial.print("Brightness: ");
    Serial.println(sensorValue);
    delay(500);

    // Check if brightness exceeds threshold
    if (sensorValue > threshold && !isBlinking) {
        isBlinking = true; // Enable blinking mode
        Serial.println("Brightness exceeded threshold. LED will start blinking.");
    }

    // Continue blinking if enabled
    if (isBlinking) {
        digitalWrite(ledPin, HIGH);
        delay(100);
    }
}
```

```

    digitalWrite(ledPin, LOW);
    delay(100);
}

// Handle Serial input for stopping the blink
if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim(); // Remove any whitespace/newline characters
    command.toLowerCase();

    if (command == "stop") {
        isBlinking = false;      // Stop blinking
        digitalWrite(ledPin, LOW);
        Serial.println("Blinking stopped.");
    }
}
}

```

❖ Code Explanation

- `Serial.begin(9600)` initializes serial communication at 9600 bps, per the guide.
- The code prints the photoresistor's analog value using `Serial.print()` and `Serial.println()`.
- If brightness exceeds **220**, blinking starts and continues even if it later drops.
- The user can type “**stop**” (any case) in the Serial Monitor to stop LED blinking using `Serial.readStringUntil('\n')`.
- The LED blinks at **100 ms** intervals as specified.

⌚ Suggested Circuit Setup

- **Photoresistor (LDR)** and **10kΩ resistor** form a voltage divider on pin **A0**.
- **LED on pin 8** with a **220Ω resistor** to ground.
- Serial Monitor baud rate: **9600**.

This code is fully compatible with the provided "Arduino Serial Connection Guide" and demonstrates the principles of serial reading, string handling, and loop-controlled LED behavior effectively.

**

please check if this code is correct:

```
// -----
// Serial Controlled Photoresistor System
// -----



// --- Pin Naming ---
#define PHOTORESISTOR A0
#define LED 8

// --- Threshold ---
const int BRIGHT_THRESHOLD = 220;

// --- Control Variables ---
bool isBlinking = false; // Tracks if LED should blink

void setup() {
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
    Serial.println("== SERIAL CONTROLLED PHOTORESISTOR SYSTEM ==");
    Serial.println("Type 'stop' to stop LED blinking.");
    Serial.println("-----");
    delay(1000);
}

// -----
// FUNCTIONS
// -----



// Function to read photoresistor brightness value
int readBrightness() {
```

```

int rawBrightness = analogRead(PHOTORESISTOR);
int brightness = map(rawBrightness, 0, 1023, 0, 255);
return brightness;
}

// -----
// MAIN LOOP
// -----
void loop() {
    // --- Read sensor value ---
    int brightness = readBrightness();
    Serial.print("Current Brightness: ");
    Serial.println(brightness);
    delay(100);

    // --- Check threshold ---
    if (brightness >= BRIGHT_THRESHOLD && !isBlinking) {
        isBlinking = true;
        Serial.println("Threshold reached! LED will start blinking.");
    }

    // --- Handle Serial Input ---
    if (Serial.available()) {
        String inputString = Serial.readStringUntil('\n'); // Read until newline
        inputString.trim(); // Remove spaces/newlines
        inputString.toLowerCase(); // Make input lowercase

        if (inputString == "stop") {
            isBlinking = false;
            digitalWrite(LED, LOW);
            Serial.println("Blinking stopped.");
        }
    }
}

// --- Blink LED if Active ---
if (isBlinking) {
    digitalWrite(LED, HIGH);
}

```

```

delay(100);
digitalWrite(LED, LOW);
delay(100);
} else {
  delay(100);
}
}

```

The code you provided is **correct and well-structured** — it meets all the requirements listed in your activity and adheres to the *Arduino Serial Connection Guide*. Below is a detailed verification of functionality and minor improvement notes:

Code Verification Summary

Requirement	Code Behavior	Status
Use of photoresistor sensor (A0)	Reads analog input from PHOTORESISTOR using <code>analogRead()</code>	<input checked="" type="checkbox"/>
Set brightness threshold = 220	Declared as <code>const int BRIGHT_THRESHOLD = 220;</code>	<input checked="" type="checkbox"/>
If threshold is reached, blink LED in pin 8 (delay = 100) and keep blinking	Implemented in conditional section and controlled by <code>isBlinking</code>	<input checked="" type="checkbox"/>
Blinking continues even if brightness drops	Controlled by persistent <code>isBlinking</code> flag	<input checked="" type="checkbox"/>
Stop blinking if user enters stop (case-insensitive)	Uses <code>Serial.readStringUntil('\n'), trim(), and toLowerCase()</code>	<input checked="" type="checkbox"/>
Serial communication baud rate (9600)	Initialized using <code>Serial.begin(9600)</code>	<input checked="" type="checkbox"/>
Output and communication functions (<code>print</code> , <code>println</code> , <code>available</code> , <code>readStringUntil</code>) follow Arduino standards	Correct usage confirmed per Serial function definitions in guide	<input checked="" type="checkbox"/>

Notes and Minor Improvements

1. Mapping the sensor value

The use of `map(rawBrightness, 0, 1023, 0, 255)` is valid; however, since your threshold is 220, the mapping compresses your range. If your brightness threshold was determined experimentally using raw analog values, you might omit `map()` and directly compare `analogRead()` to 220 for consistency.

Example:

```
int brightness = analogRead(PHOTORESISTOR);
if (brightness >= BRIGHT_THRESHOLD) { ... }
```

2. Serial Input Responsiveness

Because the code uses `Serial.readStringUntil('\n')`, the Serial Monitor must have **Newline** selected as the line ending. To make it more robust, you can document this for students.

3. Loop Efficiency

The small delay (100 ms) is acceptable for visible LED blinking and does not interfere with serial communication.

Final Verdict

The code is **correct, efficient, and meets all objectives** in your activity. It properly uses the Serial interface for communication, processes text commands case-insensitively, and demonstrates steady-state control logic ideal for an educational exercise on Arduino Serial Communication.

**

1. 3_Arduino_Serial_Connection_Guide.pdf
2. 3_Arduino_Serial_Connection_Guide.pdf