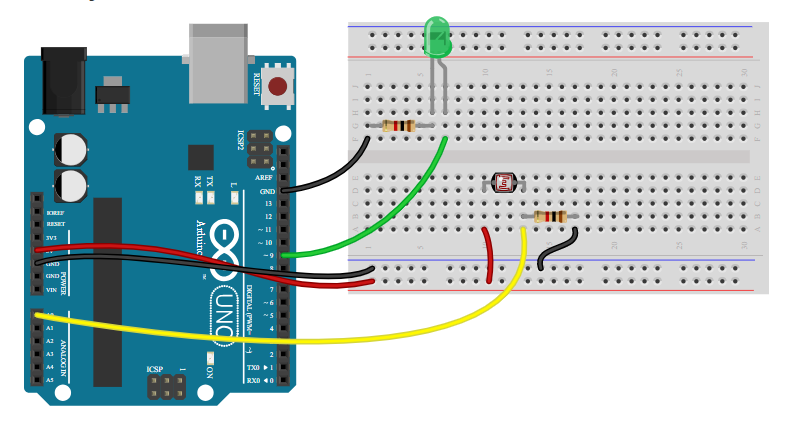
**Lab 6.1: Working with analog sensor**

In this tutorial, we will also see how Arduino can interact with your computer through the serial monitor.

**Analog In-Out Serial**

The following lesson outlines some of the principles in more detail:

1. Assembly



 To create this, you will need:

* An Arduino board
* A USB cable
* Two 1kΩ resistors
* Prototyping wires / jump leads
* A breadboard
* A photoresistor (light-dependent resistor / light sensor)
* An LED of your favorite color

### Code

The following code will light an LED connected to pin 9. We want the brightness of the LED to be proportional to the light intensity observed by the photoresistor connected to pin A0.

(Note that in "A0", the A which precedes the 0 means that it's an Analog input - the analog pins A0-A5 are located on the other side of the Arduino Uno board from the digital pins).

In this code, we take an analog input from A0, which converts the value of the voltage at the photoresistor into a number between 0 and 1023 (i.e. there are 1024 possible values). We then use a special function ('map') to scale this value onto a different range, to send it to the LED. We'll learn about this below...

**Notes:**

* This code comes pre-loaded into the Arduino IDE, and can be found by clicking on

File→ Examples→ 03.Analog→ AnalogInOutSerial

If you are using a real Arduino board, you will need to upload the code and wait until the upload is complete before opening the serial monitor discussed below.

* Did you know that you can access the documentation of a function in the Arduino IDE by right-clicking on it and clicking *Find in reference*?
* For a simulator (eg. Tinkercad), you can copy and paste the code below into the simulator window so you don't have to retype everything.

 /\*  
Analog input and output

This program will light an LED on pin 9.  
The brightness of the LED will be proportional to the brightness of the light   
falling on the photoresistor, which is attached to analog pin A0.

 \*/

// Set up constants:

const int analogInPin = A0; // Number of the pin connected to the photoresistor

const int analogOutPin = 9; // Number of the pin connected to the LED

int sensorValue = 0; // Value read by the photoresistor

int outputValue = 0; // Value sent to the LED

void **setup**() {

  // Initialise communication with the computer

**Serial**.begin(9600);

  // Indicate that the pin analogOutPin is an output:

  pinMode(analogOutPin, OUTPUT);

  // Indicate that analogInPin is an input:

  pinMode(analogInPin, INPUT);

}

void **loop**() {

  // Read the value of the photoresistor and

  // store that value in sensorValue:

  sensorValue = analogRead(analogInPin);

  // scale sensorValue to a value between 0 and 255

  // and store the value in outputValue:

  outputValue = map(sensorValue, 0, 1023, 0, 255);

  // send this new outputValue to the LED

  analogWrite(analogOutPin, outputValue);

  // sending the information to the computer

**Serial**.print("sensor = " );

**Serial**.print(sensorValue);

**Serial**.print("\t output = ");

**Serial**.println(outputValue);

}

* **New functions and keywords**:
* [analogRead](https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/) allows you to read the status of an analog pin and return a numerical value proportional to the voltage received. The Arduino board has 6 channels, A0 to A5, connected to a 10-bit analog-to-digital converter. This means that it transforms an input voltage between 0 and 5V into an integer numerical value between 0 and 1023.

analogRead(analogInPin);

The value returned by [analogRead](https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/" \t "_blank) can be stored in an integer variable which can be declared using the keyword [int](https://www.arduino.cc/reference/en/language/variables/data-types/int/" \o "digitalWrite" \t "_blank)), as follows:

sensorValue = analogRead(analogInPin);

* [**Serial**](https://www.arduino.cc/reference/en/language/functions/communication/serial/) is a library (a set of functions) used for serial port communications between the Arduino board and a computer or other components. This serial port allows you to send and receive character strings on the pins 0 (RX) and 1 (TX) with the computer via the USB port. Therefore, if you use this feature, you cannot use pins 0 and 1 as digital inputs or outputs. If you want to view the text sent from the Arduino to your computer, you can use the serial terminal built into the Arduino environment. Just click on the serial monitor button in the toolbar (**Remember**: wait until the upload is complete before opening the serial monitor):
* In the window that then opens, make sure that you have selected the right communication rate ('baud rate' - explained below) which can be found in the bottom right corner. The Arduino board gets told baud rate to use by the function **[Serial](https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/" \t "_blank)**[.begin](https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/" \t "_blank) in your code. By default the rate is 9600:
* The functions in the [**Serial**](https://www.arduino.cc/reference/en/language/functions/communication/serial/) library are:
  + [**Serial**.begin](https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/) allows initialization of the communication between Arduino and your computer. This function must be placed in the [**setup**](https://www.arduino.cc/reference/en/language/structure/sketch/setup/) block, and it must be followed by a single parameter which corresponds to the communication rate for the serial communication, in terms of the the number of characters exchanged per second (the unit is the baud) . Typically, we pick 9600 baud. To learn more about using the serial port and possible rates, click [here](https://www.arduino.cc/en/serial/begin).  Without the **[Serial](https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/" \t "_blank)**[.begin](https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/" \t "_blank) in the [**setup**](https://www.arduino.cc/reference/en/language/structure/sketch/setup/) block one cannot use the other functions, **[Serial](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Serialbegin" \t "_blank)**[.print](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Serialbegin" \t "_blank) and **[Serial](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Serialprintln" \t "_blank)**[.println](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Serialprintln" \t "_blank).

**Serial**.begin(9600);

* [**Serial**.print](https://www.arduino.cc/en/Serial/Print) this function allows you to send a set of characters indicated in parentheses to the serial port. In the same way,  **[Serial](https://www.arduino.cc/en/Serial/Println" \t "_blank)**[.println](https://www.arduino.cc/en/Serial/Println" \t "_blank) sends a set of characters and follows them with a line break:

**Serial**.print("Text sent to the computer"); // without line break  
**Serial**.println("Text with a line break"); // with line break

* [**Serial**.print](https://www.arduino.cc/reference/en/language/functions/communication/serial/print/) will be particularly useful for a task you have already done without realizing it: **debugging**. It is not uncommon to find that while you can upload code to an Arduino without any problem, the behavior of your program is not what you expected. We all make mistakes, but thanks to [**Serial**](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Serial), we will be able to indicate when we turn on an LED or when we do a test ... This way it's possible for us to follow the progress of our program! But that's not all - **[Serial](https://www.arduino.cc/reference/en/language/functions/communication/serial/print/" \t "_blank)**[.print](https://www.arduino.cc/reference/en/language/functions/communication/serial/print/" \t "_blank) and **[Serial](https://www.arduino.cc/reference/en/language/functions/communication/serial/println/" \t "_blank)**[.println](https://www.arduino.cc/reference/en/language/functions/communication/serial/println/" \t "_blank) can also display the value of variables if we indicate a variable name in parentheses:

**Serial**.print("sensor = " );

**Serial**.print(sensorValue);

* [map](https://www.arduino.cc/reference/en/language/functions/math/map/) allows you to scale a value from one interval to another. The parameters of these functions are as follows:
  + - * initial variable (that is within the initial interval)
      * **lowest** possible value of the **initial** interval
      * **highest** possible value of the **initial** interval
      * **lowest** possible value of the **target** interval
      * **highest** possible value of the **target** interval

Thanks to this function, we will find ourselves with a proportionately scaled value!

In other words - since sensorValue can be anywhere between 0 and 1023, but we want to scale sensorValue down so it always fits within the output interval of 0 to 255. So we use the [map](https://www.arduino.cc/reference/en/language/functions/math/map/) function as follows:

map(sensorValue, 0, 1023, 0, 255);

* [analogWrite](https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/) will be used in this program to modulate the light intensity of a LED connected to the pin specified with the first parameter. In the past weeks, our LEDs were on or off. The advantage of [analogWrite](https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/" \t "_blank) is to be able to adjust the luminous intensity by specifying a number (between 0 and 255) in the second parameter of the function:  
  :

analogWrite(11, 0); // completely turns off the LED on pin 11  
analogWrite(11, 90); // lights up the LED on pin 11 a little bit  
analogWrite(11, 255); // puts the LED on pin 11 to full brightness