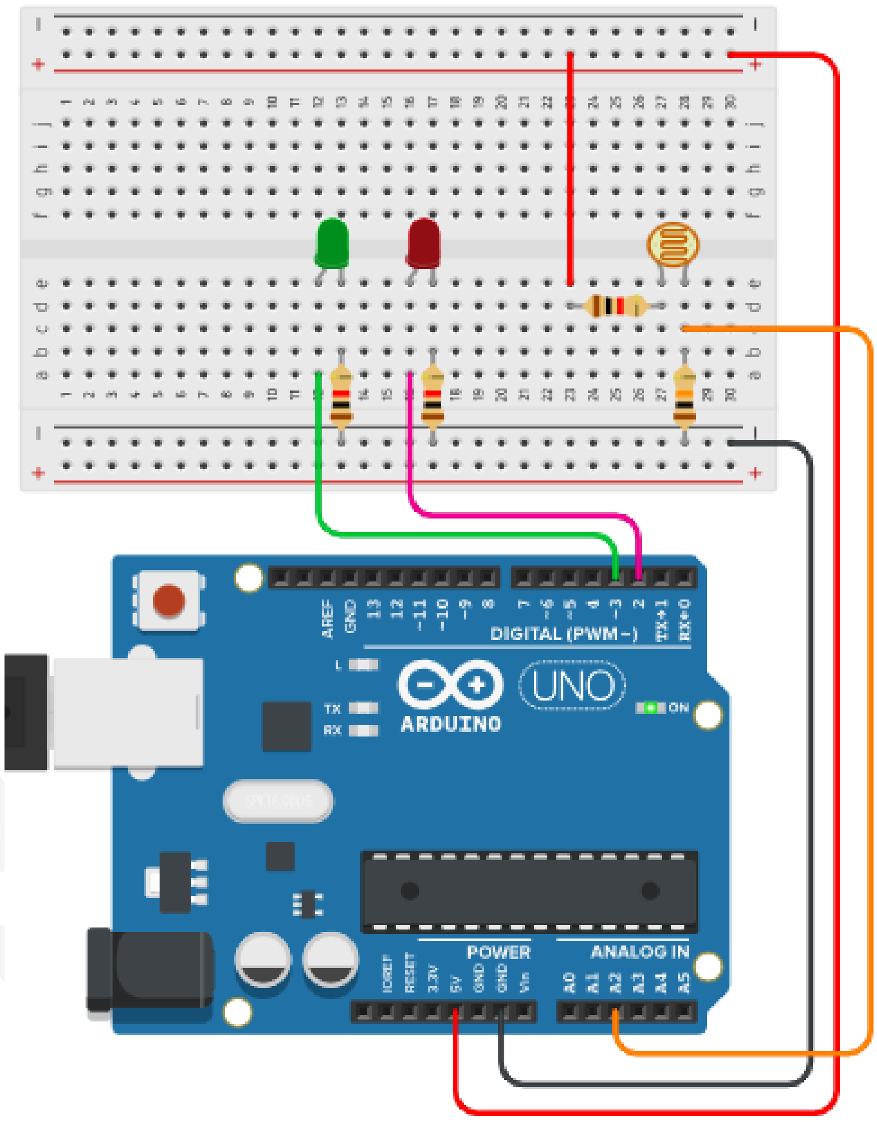
Activity: Interfacing Analog Light Sensor with Arduino

Objective:

We have already seen, learnt and programmed an Arduino for receiving an analog input on any of its 6 analog input pins and then sending the value over to the serial monitor but we have not actually connected any sensing component so far.



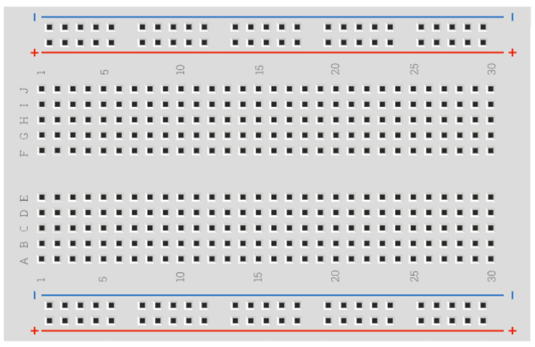
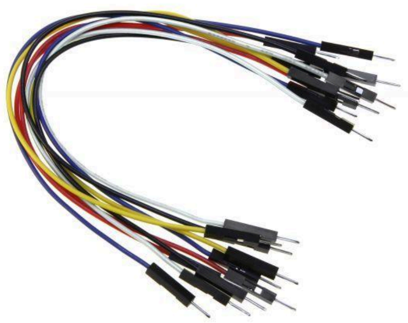
Let's try it out by choosing the most basic analog sensor of all, the LDR. The Light Dependent Resistor (LDR) or a photoresistor is exactly what its name suggests. It is a resistor whose resistance changes according to the amount of Light falling on it. So, working principle of an LDR is simple:

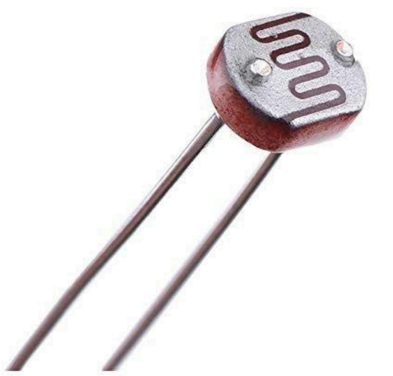
"More Light = Less Resistance and Less Light = More Resistance"

Now clearly, the value of this resistance is continuously changing as the light falling on it changes so it is a continuously varying value and not just 0 and 1. In other words this resistance or in other words the Light is not a Digital but an Analog entity. And so these cannot be read by "digitalRead();". Just as we have 0 and 1 for a digitalRead();, we have values ranging from 0 to 1023 for an "analogRead();"

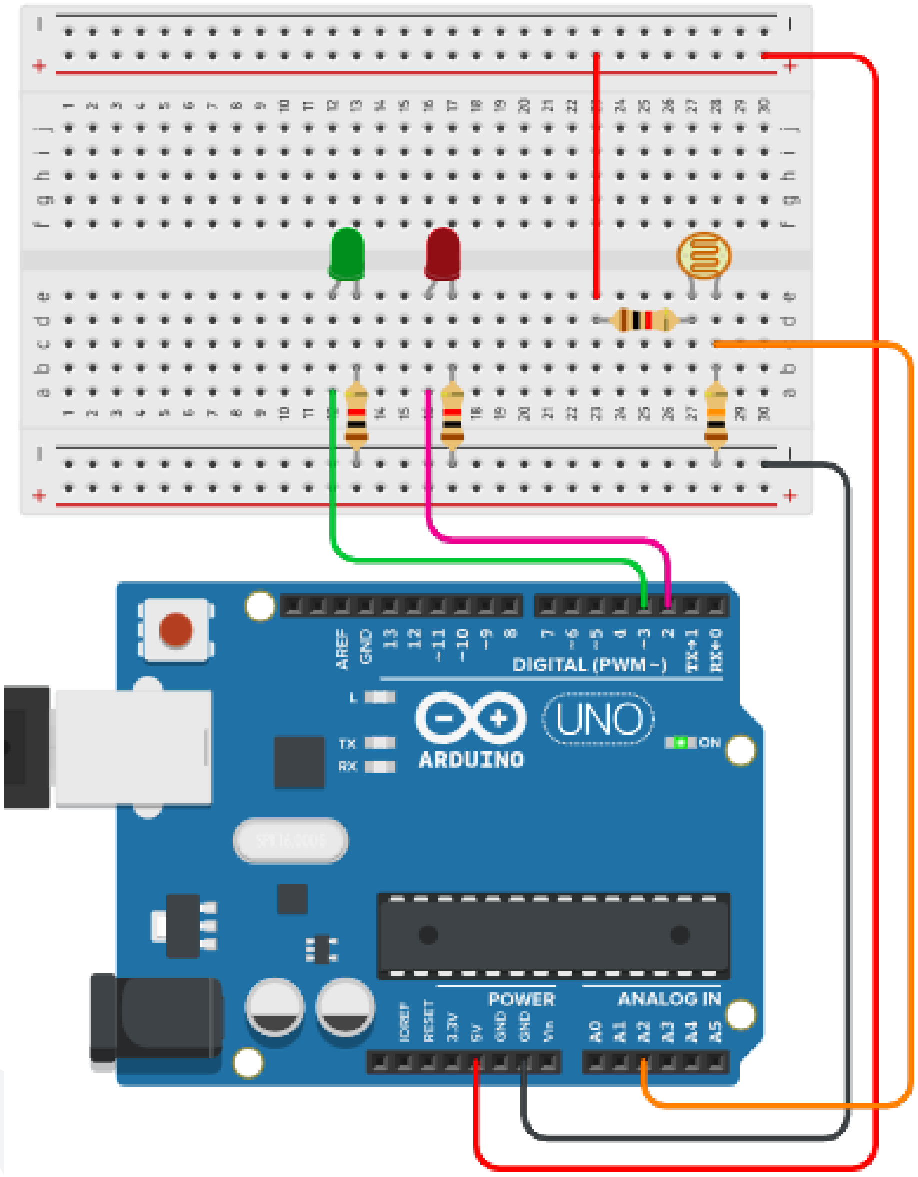
So, in today's activity we will learn how to take Light Intensity as an analog Input using Arduino. These analog values for the changing light intensities, once read, will be sent over to the Computer via the USB using the Arduino's Serial Monitor Utility. And finally two different outputs in the form of Red and Green LEDs are controlled.

Materials Required:

S.no.	Item	Qty	Image
1	Arduino	1	
2	Breadboard	1	
3	M-M Connection Wires	10	
4	5mm Red LED	1	
5	5mm Green LED	1	

6	1k Resistor (Brown Black Red Golden)	3	
7	10k Resistor (Brown Black Orange Golden)	1	
8	LDR	1	
9	USB Cable for Arduino	1	

Connection Diagram:



Explanation:

The connection diagram shown in the image above has the Light sensing component (the LDR) as the Analog Input and two LEDs (Red and Green) with their individual resistors as the Outputs.

The LDR is connected such that its one terminal is connected to the +ve of the breadboard with the 1k series resistor in the middle. The second terminal of the LDR is connected to one of the terminals of the 10k resistor while the other side of the 10k resistor is connected to the -ve of the Breadboard.

The junction, where the LDR meets the 10k resistor, is connected to the Arduino Analog pin 2 (A2) and is shown by the Orange line.

The Red LED is connected such that its +ve terminal is connected to the Arduino's digital Pin 2 (D2) and the -ve terminal is connected to the 1k resistor. The other side of the 1k resistor is connected to the -ve of the Breadboard.

The Green LED is connected such that its +ve terminal is connected to the Arduino's digital Pin 3 (D3) and the -ve terminal is connected to the 1k resistor. The other side of the 1k resistor is connected to the -ve of the Breadboard.

Lastly, the Arduino's 5v pin is connected to the +ve of the breadboard and the Gnd pin is connected to the -ve of the breadboard.

Arduino Code:

The Arduino code is simple but needs to be done in 2 parts. Let's see why. We have 1 analog input whose value can range from some minimum value to some maximum value. And we have 2 outputs, one of which we want to turn On (and the other remains Off) when the present value of the sensor is below a given **threshold** (just a fancy word for limit). We want the second Output to turn On (and the first remains Off) when the present value of the sensor is above the given threshold.

If we don't know where to place this **threshold**, we won't be able to set when the Red LED goes On or Off and when the Green LED goes On or Off. So first, we need to find out the value of this **threshold**. For this, we simply send the value of the sensor continuously to the Computer using "Serial.println();".

Here is that initial code:

```
#define LDR   A2                   //pin A2 named as LDR

int light_value;                   // variable for light intensity

void setup()
{
  pinMode(LDR,INPUT);              // LDR pin as Input

  Serial.begin(9600);              // begin serial comm at 9600 bps
}

void loop()
{
  light_value = analogRead(LDR);   //read LDR pin and store in variable

  Serial.println(light_value);     // send stored value to computer
  delay(200);                      // wait for 200ms
}
```

Explanation:

```
#define LDR   A2                   //pin A2 named as LDR

int light_value;                   // variable for light intensity
```

Here, we are telling the Arduino that from now on, we will call pin **A2** as **LDR** and then take an Integer type variable “**light_value**” for storing the analog value.

```
void setup()
{
  pinMode(LDR,INPUT);              // LDR pin as Input

  Serial.begin(9600);              // begin serial comm at 9600 bps
}
```

Here, inside the setup function, we are declaring that we are going to use LDR (pin 2) as Input and that we are beginning a serial communication with the Computer at a speed of 9600 bits per second (bps).

```
void loop()
{
  light_value = analogRead(LDR);   //read LDR pin and store in variable

  Serial.println(light_value);     // send stored value to computer
  delay(200);                      // wait for 200ms
}
```

Here, inside the loop function, we are continuously:

1. Reading the LDR pin and storing its value in the “light_value” variable.
2. Printing the value of this variable on the serial monitor (computer’s USB).
3. Waiting for a short time to make sure the sent value is printed for sure.

Once this code is compiled and uploaded, we will see a continuous stream of live values printed on the serial monitor terminal. These values will change suddenly when we place our hand above the LDR, casting a shadow on it. These values will again change suddenly when we move our hand away from the LDR.

We are looking for this approximate value on one side of which we will call it dark and on the other side of it we will call it bright. This value need not necessarily be the same for everyone. It will vary from sensor to sensor and place to place depending on how much light or shadow is at a given place.

Let’s say we found this threshold value to be 200. So now, we will turn On Red on one side (greater than) of this value and turn On Green on the other side (less than) of this value.

So, we will use two sets of “if” conditions to compare the present value to the threshold. Then, depending on whether the value is less than or greater than 200 we will “digitalWrite();” the Red and the Green LEDs accordingly. Given below is the complete Arduino code for doing the process that was just described.

Here is the complete Arduino code for part 1.

```
#define LDR   A2     //pin A2 named as LDR
#define Red   2      //pin 2 named as Red
#define Green 3      //pin 3 named as Green

int light_value;     // variable for light intensity
```

```
void setup()
{
  pinMode(LDR,INPUT);     // LDR pin as Input
  pinMode(Red,OUTPUT);    // Red pin as Output
  pinMode(Green,OUTPUT); // Green pin as Output

  Serial.begin(9600);     // begin serial comm at 9600 bps
}



void loop()
{
  light_value = analogRead(LDR);  //read LDR pin and store in variable

  Serial.println(light_value);    // send stored value to computer
  delay(200);                     // wait for 200ms

  if(light_value > 200)           // if value more than threshold
    {
      digitalWrite(Red,HIGH);     // turn Red LED On
      digitalWrite(Green,LOW);    // turn Green LED Off
    }

  if(light_value < 200)           // if value less than threshold
    {
      digitalWrite(Red,LOW);      // turn Red LED Off
      digitalWrite(Green,HIGH);   // turn Green LED On
    }
}
```

Here, we added the comparison part using the “if”. One “if” compares if the value is Greater than 200 while the other “if” compares if the value is Less than 200.

Consequently, the code does the following:

1. If the present measure analog value is greater than 200, the Red LED is turned On and the Green LED is turned Off.

2. If the present measure analog value is less than 200, the Red LED is turned Off and the Green LED is turned On.

Outcome and Observations:

1. Once the Arduino code is compiled and uploaded, start the serial monitor by selecting the proper baud rate number. You will see values being displayed continuously one below another.

2. If we move our hand such that it casts a shadow on the LDR light sensor, the value of the analog readings decreases and as we keep moving our hand closer and closer to the LDR light sensor, the value keeps decreasing and finally when the LDR light sensor is made completely dark, the value drops to minimum.

3. Now, if we keep moving our hand farther and farther away from the LDR light sensor, the value of the analog readings keeps increasing and finally there comes a point where the value no longer changes even if we move our hand farther.

4. If, however, we take the setup to a brighter room or outdoors, the total range of analog values will vary more as the amount of light available is more.

5. If we change the 100k resistor with a 10k resistor, the entire analog value range shifts down.

6. If we want the setup to behave in the opposite way i.e. moving the hand closer should reduce the analog readings and moving the hand farther should increase the readings, then we need to take the inverse of the “light_value” variable.

   So, after the following line

   ```
   light_value = analogRead(ldr);
   ```

   We will add the following line

   ```
   light_value = 1023 - ldr_value;
   ```

   Adding this line will reverse the behavior of the setup. Now, more light will make lower readings and less light will make higher readings.