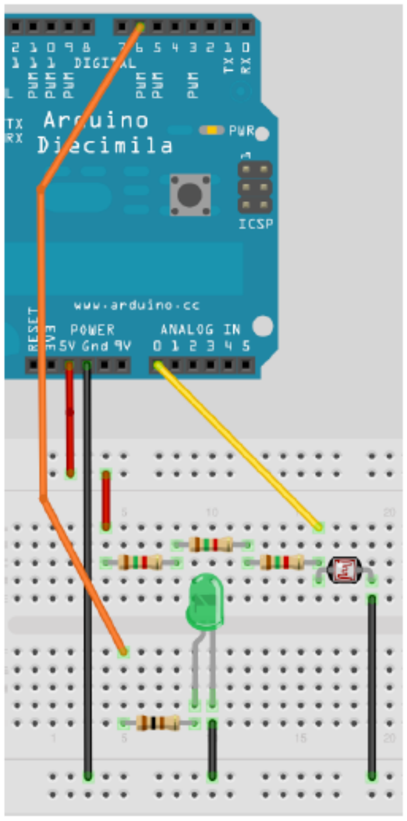
Project 14 - Light Sensor

In this project we are going to use the Light Dependent Resistor in our kit to read values from it and output them to the Serial Monitor.

What you will need

Light Dependent Resistor	
100Ω Resistor	
3 x 1K5Ω Resistors	
Green LED	

Connect it up



Enter the Code

Enter the code, then upload it to your Arduino. You will see the LED flashing on and off. If you cover the LDR (Light Dependent Resistor) you will see the LED flash slower. Now shine a bright light onto the LDR and you will see it flash faster.

```
//Project 14 - Light Sensor

// Pin we will connect to LED
int ledPin = 6;
// Pin connected to LDR
int ldrPin = 0;
// Value read from LDR
int lightVal = 0;

void setup()
{
       // Set both pins as outputs
       pinMode(ledPin, OUTPUT);
}

void loop()
{
       // Read in value from LDR
       lightVal = analogRead(ldrPin);
       // Turn LED on
       digitalWrite(ledPin, HIGH);
       // Delay of length lightVal
       delay(lightVal);
       // Turn LED off
       digitalWrite(ledPin, LOW);
       // Delay again
       delay(lightVal);
}
```

Project 14 - Code Overview

This code is pretty simple and you should be able to work out what it does yourself by now.

The code starts off by initialising variables related to Digital Pin 6, which the LED is connected to and Analogue Pin 0, which the LDR is connect to. We also initialise a variable called lightVal which will store the values red in from the LDR.

```
int ledPin = 6;
// Pin connected to LDR
int ldrPin = 0;
// Value read from LDR
int lightVal = 0;
```

The setup function sets the pinmode of the LED pin to output.

```
      pinMode(ledPin, OUTPUT);
```

In the main loop of the program we read in analog value from Analog Pin 0 and store it in the 'lightVal' variable.

```
      lightVal = analogRead(ldrPin);
```

Then the LED is turned on and off, with a delay equal to the value read in from the analog pin.

```
      digitalWrite(ledPin, HIGH);
      delay(lightVal);
      digitalWrite(ledPin, LOW);
      delay(lightVal);
```

As more light falls on the LDR the value read in from Analog Pin 0 decreases and the LED flashes faster.

Let's find out how this circuit works.

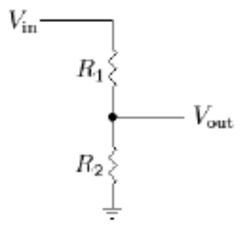
Project 14 - Hardware Overview

The only additional component used in this circuit is the LDR or Light Dependent Resistor (sometimes called a photoresistor). An LDR initially has a very high resistance. But, as light falls on it, the resistance will drop, allowing more current through.

Our LDR is connected in series with 3 x 1.5KΩ Resistors and the input into Analog Pin 0 is between these 2. This is what is known as a voltage divider. We will explain this in a second.

The 3 x 1.5K give a total resistance of 4500Ω (4.5KΩ). Resistors in series have a resistance equal to the sum of their individual resistances. In this case the value is 3 x 1500 = 4500.

A voltage divider is a circuit consisting of two resistances across a voltage supply. An output between the two resistances will give a lower voltage depending on the values of the two resistors.



The diagram on the left shows a voltage divider made up of two resistors. The value of Vout will be lower than the value of Vin.

To work out the value of Vout we use the following calculation:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

We are providing 5 volts into the circuit so let's work out what values we will get out. Using a multimeter I have measured the resistance from the LDR in different conditions.

Conditions	Resistance
LDR Covered by Finger	8KΩ
Light in room (overcast day)	1KΩ
Held under a bright light	150Ω

So using these values of resistance, the input voltage and the calculation we listed above, the approx. output voltage can be calculated thus:

V_{in}	R_1	R_2	V_{out}
5v	4500Ω	8000Ω	3.2v
5v	4500Ω	1000Ω	0.9v
5v	4500Ω	150Ω	0.16v

As you can see, as the resistance of the LDR (R_2) decreases, the voltage out of the voltage divider decreases also, making the value read in from the Analog Pin lower and therefore decreasing the delay making the LED flash faster.

A voltage divider circuit could also be used for decreasing a voltage to a lower one if you used 2 standard resistors, rather than a resistor and an LDR (which is a variable resistor). Alternatively, you could use a potentiometer so you can adjust the voltage out by turning the knob.

Note: you don't have to use the exact same resistors as in the manual. Just use ones with comparable values from your kit. You may use a flashlight from a smart phone for illumination.

- You are now going to modify the code to output the numerical values of light intensity via the serial port. You may remove the LED part of the circuit from your breadboard and delete the associated lines in your code.

You will use `` `analogRead` `` function to read analog input voltages using Arduino:

https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

And use Serial Monitor in Arduino IDE and associated Arduino functions (e.g., `Serial.begin`, `Serial.print`, `Serial.println`, etc) as discussed in consulting Arduino reference

https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/

and

https://www.youtube.com/watch?v=vDaEPJszaXk

- Write an Arduino code to measure the resistance of the LDR approximately every 0.1 second (that is, a sampling rate of 10 Hz) using `delay` function. You will learn better ways to control sampling rates later. For each measurement, your code should output the time of measurement (in milliseconds) and the value of R_2 calculated from the measured analog voltage to the serial port.

Use `millis`() function in your code (consult Arduino reference manual) to get the time of each measurement. Do NOT use time stamps from Arduino IDE Serial Monitor.

Pay attention to the format and put a tab to separate the time value from the resistance value:

```
12345         123.7
12445         235.3
12544         895.1
```

Some examples on how to format the number outputs

```
Serial.print(x, 4);   // print the value of x using 4 decimal places
                      // (the second number specifies precision)
Serial.print("\t");  // print a tab
Serial.println(y);  // print the value of y and a line break
```

- Perform experiments. Display the values on your laptop screen using Serial Monitor. Make sure you are getting reasonable resistance values (10~50,000 Ω) as you change the light intensity. Debug as necessary!

Take a screen shot of your Serial Monitor (Figure 1).

If your Serial Monitor shows strange characters or nothing, check to make sure that the baud rate in your code (e.g., Serial.begin(250000)) match the baud rate selected in Serial Monitor (on the bottom right of the screen).

- Copy the outputs on the Serial Monitor to a spreadsheet and plot the data (Figure 2). Your plot should illustrate that the resistance of the photoresistor changes as a function of time (in millisecond) as you change the illumination intensity.

For debugging purposes, you may use Serial Plotter (under Tools in Arduino IDE) for "real-time" plotting of the data outputted to a serial port

https://www.instructables.com/id/Ultimate-Guide-to-Adruino-Serial-Plotter/

Thermistor

In the previous practice, we used a photoresistor, which is a light dependent resistor. A thermistor is a resistor whose electrical resistance is a function of temperature. We can measure the resistance and use a look up table or calibration function to calculate the corresponding temperature.

A common mathematical function we use to represent the relation between the electrical resistance of a thermistor and the temperature is the Steinhart and Hart equation (consult Wikipedia):

$$T = 1/\left[A + B \ln R + C (\ln R)^3\right]$$

T is in Kelvin and R is in Ohm **(pay close attention to the units!!!).** From a fit to the manufacturer's data, we have:

```
const double A = 0.00116741481796827;
const double B  = 0.000227371846448138;
const double C = 0.000000119663032602201;
```

- Modify your Arduino code such that it now takes a temperature reading approximately every 0.1 second (i.e., a sampling frequency of 10 Hz) and outputs the time of measurement (**in seconds and NOT in milliseconds**) and measured temperature value through the serial port.

Make sure your codes are all well organized, well formatted, and well-documented with comments!

Perform measurements after you replace the photoresistor from Project 14 with your thermistor and the reference resistor (R_1) with a 10 kOhm resistor.

Confirm that you are getting the reasonable room temperature (say, around 20 °C). Debug your circuit and code as necessary.

- Next, record the temperature as you immerse the thermistor in an ice water. You should mix ice cubes with tap water (and stir very well) at least a few minutes before you perform this experiment. Plot the temperature (°C) as a function of time (**in seconds**) (Figure 3).

- Nondimensionalize the temperature profile using the initial temperature T_i and the "final" temperature T_∞:

$$\theta = \frac{T - T_\infty}{T_i - T_\infty}$$

Trim the nondimensionalized data θ to show ONLY the **decaying portion** and then fit it using an exponential function to determine the time constant τ.

$$\theta = e^{-t/\tau}$$

Your next plot (Figure 4) should report the time constant in the caption and show the nondimensionalized temperature data and this exponential fit for easy comparison.

Tip 1:

```
//    Use the 3.3V source, which is cleaner and more stable than the 5V source,
// to bias the voltage divider circuit.

//    Connect the 3.3V source also to pin AREF and include inside setup() in your code

              analogReference(EXTERNAL);
```

Tip 2:

```
// Find the resistance of the thermistor using the voltage divider equation and
// calculate its logarithm.  V_ref is the voltage divider bias voltage and R_ref is the resistance of
// the reference resistor.

double V_ref = 3.3V;
double R_ref = 10000.0;
double log_Resistance = log( (VA2 / (V_ref - VA2)) * R_ref );


// Use Steinhart and Hart equation to calculate temperature

double Temperature = 1.0/( A + B * log_Resistance + C * pow(log_Resistance,3.0) );
```