Exercise 1.1: Blink

This example shows the simplest thing you can do with an Arduino or Genuino to see physical output: blinking the on-board LED light

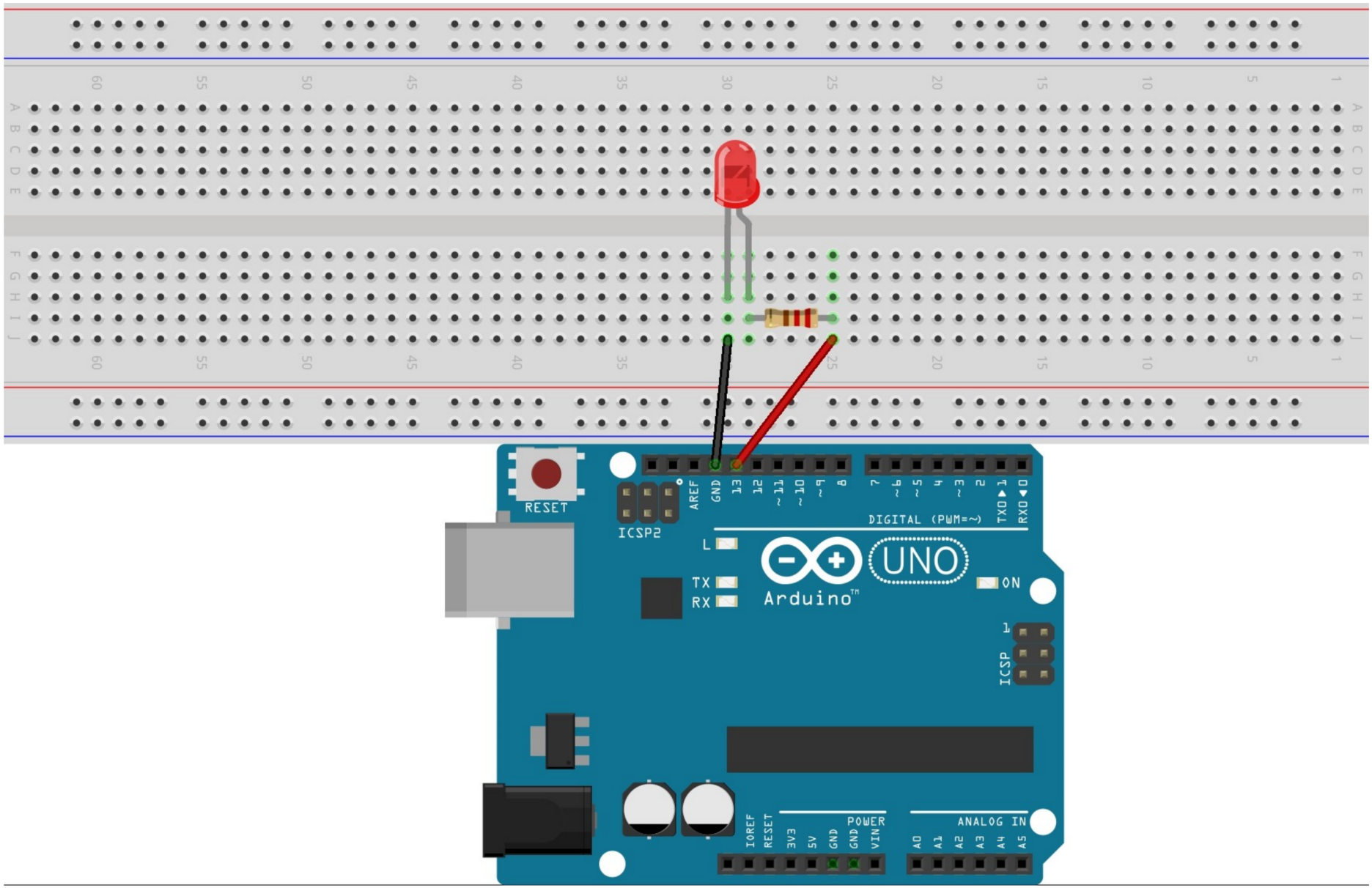
Hardware Required

- Arduino UNO R3
- LED
- 330 ohm resistor

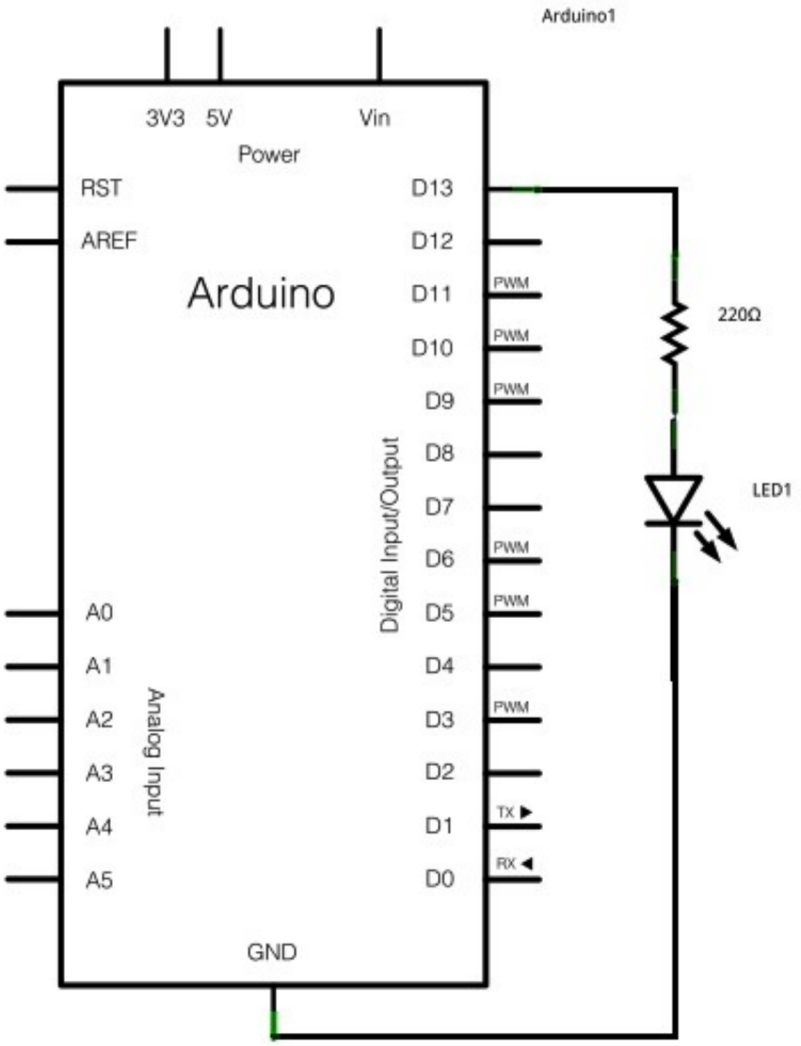
Circuit

This exercise uses the built-in LED that most Arduino and Genuino boards have. This LED is connected to a digital pin and its number may vary from board to board. To make your life easier, we have a constant that is specified in every board named *LED_BUILTIN*.

If you want to light an external LED, you need to build a circuit, where you connect one end of the resistor to the digital pin corresponding to the *LED_BUILTIN* constant. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the GND. What leg you connect the resistor to does not matter, but polarity of the LED will matter. This is because it limits current in one direction but not in the other direction. In the diagram below we show an UNO board that has D13 as the LED_BUILTIN value. The value of the resistor in series with the LED may be of a different value than 330 ohm; the LED will light up with values up to 1K ohm.


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

Schematic



Code

After you've built the circuit plug your Arduino board into your computer. Start the Arduino Software (IDE) and load the code from the menu File/Examples/01.Basics/Blink.

Exercise 1.2: Modified blink

This is a continuation on the last exercise in which you learned how to make a LED blink with an Arduino och Genuino

Hardware Required

Arduino UNO R3

LEDs 1-5pcs

330 ohm resistor 1-5pcs

Circuit

This exercise uses the same circuit as the previous part. For more information check exercise 1.1. You are free to expand on this circuit with more LEDs and resistors if you'd like.

Code

In the blink code there's a line looking like this:

```
pinMode(LED_BUILTIN, OUTPUT);
```

If we want to use our own pins to control LED lights we can use the function in a similar way. Before the setup function begins (on row 5 for example), we can define our pins like this: **const int ledPin = 2;**

const means that the pin will stay the same throughout our code. **int** means that the variable will be a number, arduino digital pins are defined as numbers. **ledPin** is the name we give the variable, this is to make it easier later to know which pin is used for what. We can use multiple pins to light LEDs like this:

const int ledPin = 2;
const int bestLed = 3;
const int secretPin3 = 4;

In the setup function we have to set their pinmode to `ore using them. Something like this can be used:

```
pinMode(ledPin, OUTPUT);
pinMode(bestLed, OUTPUT);
pinMode(secretPin3, OUTPUT);
```
It is important that these are inside the setup function.

It's up to you to explore this exercise and create your own blinking circuit. The goal is to get a sense on how the code works and make your own patterns of blinking LEDs. You can use code from the previous example and expand on it or try to write your own.

Examples on how you can upgrade your circuit:

- Increasing or decreasing the delay to make the LED blink faster or slower
- Defining more pins so you can have more LEDs
- Trying to make some kind of pattern
- Explore and have fun

Exercise 2.1: Potentiometer

In this exercise you will learn about analog pins and how to use a variable resistor - a potentiometer to make a LED blink with different frequencies.

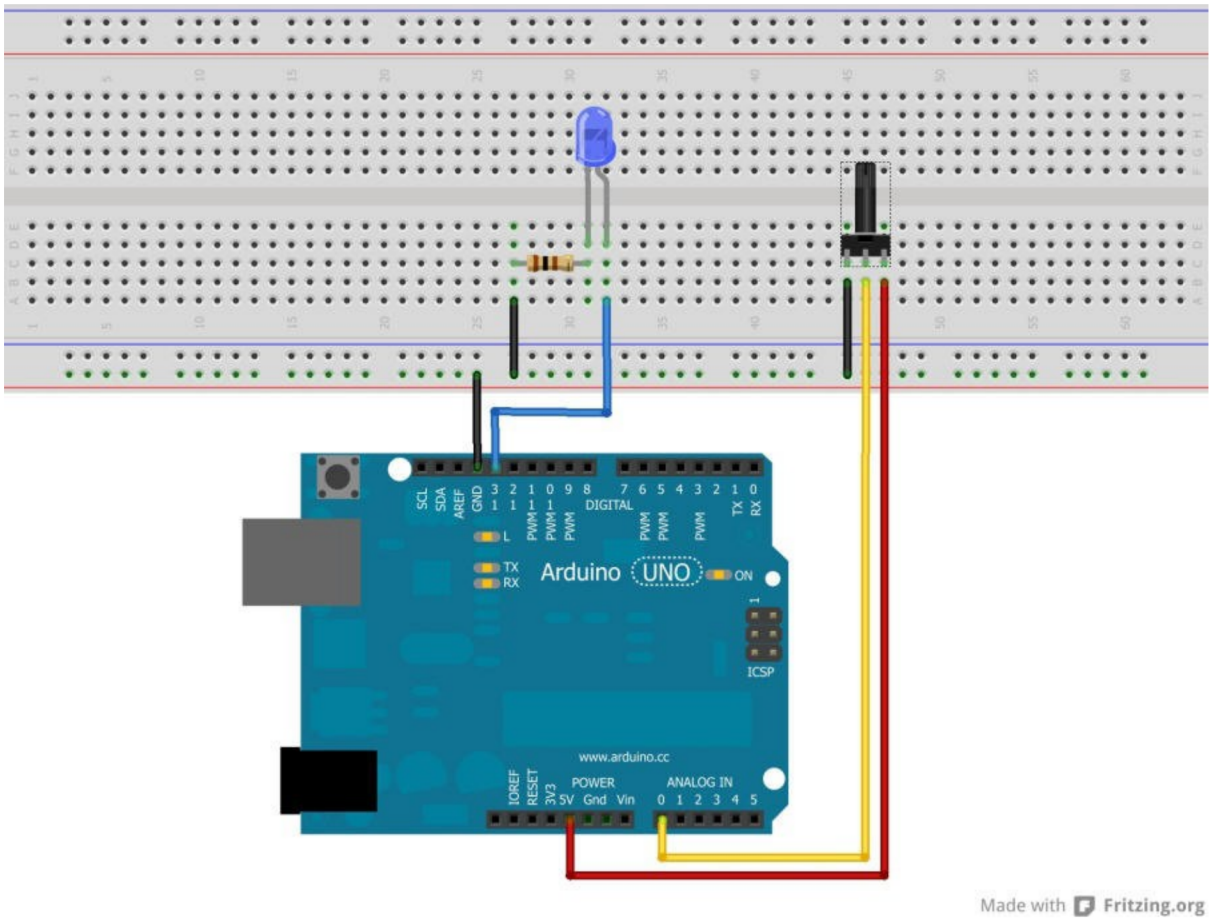
Hardware Required

- Arduino UNO R3
- LED
  - 330 ohm resistor
  - Potentiometer with knob

Circuit

This exercise will use some of the information you've learned about LED lights but we also add a potentiometer. A potentiometer is a resistor like the one we use for LED, but there is one difference. We can change the resistance by rotating the knob. With an arduino we can apply a voltage to one side and read the voltage on the middle pin. When we change the resistance the voltage changes too. We can use this fact to get values from 0-1023. Why 1023? Arduinos analog values are 10-bit. We won't go into detail about it, but this gives you enough information to look up what it means.

Back to the circuit. We can use these values to change our delay for example. So when we turn the knob the LED blinks faster or slower.



Code

The file for this exercise can be found under File/Examples/03.Analog/AnalogInput. This code changes at what rate the LED blinks.

Exercise 2.2: Potentiometer dim 1

In this exercise you will learn about analog pins and how to use a variable resistor - a potentiometer to dim a LED.

Hardware Required

- Arduino UNO R3
- LED
- 330 ohm resistors
- Potentiometer with knob

Circuit

This exercise uses a similar circuit as previous exercise but instead of connecting the LED to digital pin 13 you connect it to digital pin 3 which has PWM. This is used to dim a LED.

Code

For this exercise you will use the same code for reading the potentiometer but instead of connecting the LED to digital pin 13 you should instead connect it to digital pin 3.

Pin 3 has PWM, which stands for pulse width modulation. This is a method to change the voltage. Pin 3 and some other pins on the arduino have it enabled which lets you decide the brightness of the LED. To set different brightness we have to use something different from the HIGH or LOW that the digitalwrite uses. The function analogWrite() can do that. You can change the pin to your own and use values from 0-255 using analogWrite(yourpin,value). This is a bit lower than the analogRead() function and instead of taking in values we are pushing out values.

Instead of putting in our own value we can use the read value from our potentiometer. Remember that you need to divide the value from the potentiometer by 4 since this value goes from 0-1023 instead of 0-255.

Exercise 2.3: Potentiometer dim 2

In this exercise you will have to create a circuit that uses a potentiometer to simultaneously dim down one LED and brighten one LED. Keep your circuit for the next exercise.

Hardware Required

- Arduino UNO R3
- LEDs 2pcs
- 330 ohm resistors 2pcs
- Potentiometer with knob

Circuit

This exercise uses a similar circuit as previous exercise. For more information and ideas on how this works check exercise 2.1 and 2.2.

Code

In this part you are only going to get some structure for the code so you will have to write it yourself using the main concepts that you learned in previous exercises. If you find it hard or if there is something you don't understand you can look it up at https://www.arduino.cc/reference/en/ or ask one of the instructors for this course.

Here is some help on how the structure of the code might look like. Comments marked with red are more challenging but makes your code look cleaner and execute faster(though not noticeable in this case).

```
define your PWM pins for the LEDs
define your input pin for the potentiometer

void setup() {
  // put your setup code here, to run once:

Set up input for potentiometer
Set up output for leds
Set up two variable to handle value for LEDs

}

void loop() {
  // put your main code here, to run repeatedly:

Read input from the potentiometer
Use a function to decrease the value of one variable
Use a function to increase the value of one variable

If you want to challenge yourself you can try to do these two actions with one function

Use a function to take the value from one LED-variable and increse/decrease the brightness of one LED
Use a function to take the value from the other LED-variable and do the opposite to the brightness of the other LED

If you want to challenge yourself you can try to do these two actions with one function
}
```

Exercise 3.1: Serial.print

In this exercise you will have to use serial.println to get out the values that you receive from the potentiometer in exercise 2.1-2.3.

Hardware Required

Arduino UNO R3

LEDs 2pcs

330 ohm resistors 2pcs

Potentiometer with knob

Circuit

This exercise uses the same circuit that you built in exercise 2.1-2.3

Code

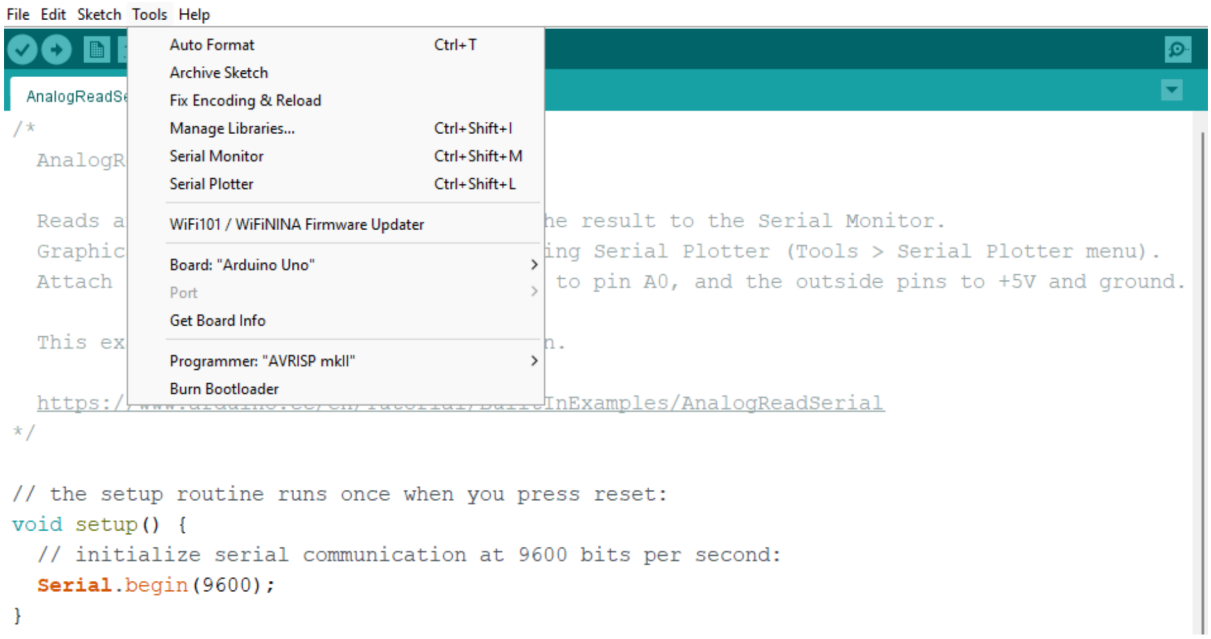
In this part you are only going to expand the previous code that you built in exercise 2.1-2.3 by printing the value received from the potentiometer using serial print. You can then read what the arduino prints out using the serial monitor and plot it using the serial plotter under tools. For inspiration see the arduino example ReadAnalogVoltage under basic examples.

Useful information

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

https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-plotter/

https://docs.arduino.cc/language-reference/en/functions/communication/serial/print/

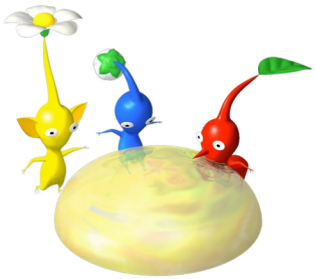

File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
AnalogReadSe
/*
AnalogR
Reads a
Graphic
Attach
This ex
https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogReadSerial
*/
// the setup routine runs once when you press reset:
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
}

Exercise 4.1: LDR fading

In this exercise you will have to create a circuit that uses another type of resistor called LDR(Light Dependent Resistor/photoresistor) to make a LED light up when it gets dark.

Hardware Required

- Arduino UNO R3
- LED
- 330 ohm resistor
- 1k ohm resistor
- LDR



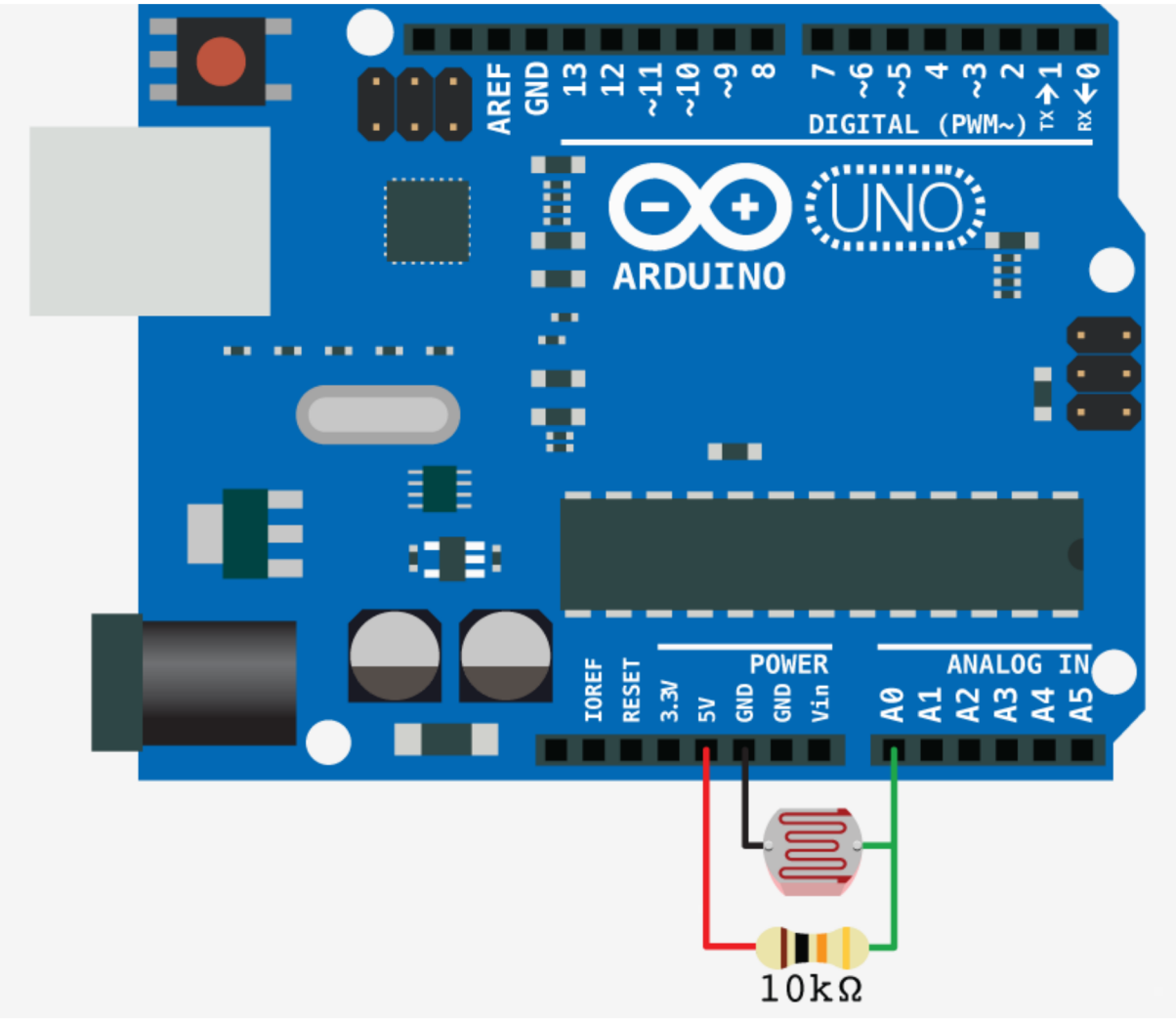
Circuit

In this exercise you need to create your own circuit, but to not make it too hard we will give you some help. A voltage divider is used to lower the total output voltage by letting some current flow into ground through a second resistor. So how do you create one? You connect two resistors in series(one after the other one). In this case the LDR is one of the resistors. In the middle between the resistors you connect an analog arduino pin so you can read the value.

Goals:

1. Create a voltage divider
2. Read analog value from LDR
3. Dim LED when its bright

In this example you get a picture on how to connect the LDR and resistor.



Code

In this part you have to write your own code with the basic concepts. If you find it hard or if there is something you don't understand you can look it up at https://www.arduino.cc/reference/en/ or ask one of the instructors for this course.

One tip is to divide the values you get from the LDR so that it is somewhere around 255 when it's not covered. This way your LED dim down with about the same increment as the LDR.

Exercise 4.2: LDR switch

In this exercise you will have to create a circuit that uses LDR to make a LED turn on when it is dark and turn off when it is bright.

Hardware Required

Arduino UNO R3

LED

330 ohm resistor

1k ohm resistor

LDR

Circuit

In this exercise you need to create your own circuit. The voltage divider is the same for this as the previous exercise.
Goals:

1. Create a voltage divider
2. Read analog value from LDR
3. Turn off LED when it is bright
4. Turn on LED when it is dark

It is up to you to decide when it's considered dark so don't be afraid to play around with the values. A real world example for this circuit is an automatic night light. When it gets dark it turns on and when it gets bright it turns off.

Code

In this part you have to write your own code with the basic concepts. If you find it hard or if there is something you don't understand you can look it up at https://www.arduino.cc/reference/en/ or ask one of the instructors for this course.

Exercise 4.3: LDR darkness

In this exercise you will have to create a circuit that uses a LDR to make sequential LEDs light up depending on how dark it is.

Hardware Required

Arduino UNO R3

LEDs 3-5pcs

330 ohm resistors 3-5pcs

1k ohm resistor

LDR

Circuit

In this exercise you need to create your own circuit. The voltage divider is the same for this as the previous exercise.

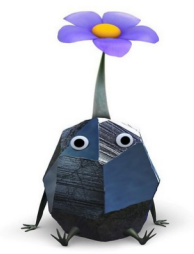
Goals:

1. Create a voltage divider
2. Read analog value from LDR
3. All LEDs should be off when it is bright
4. Depending on how dark it is, more LEDs should light up until all LEDs are lit.

It is up to you to decide when it's considered dark so don't be afraid to play around with the values. This is a simple way to create a brightness sensor

Code

In this part you have to write your own code with the basic concepts. If you find it hard or if there is something you don't understand you can look it up at https://www.arduino.cc/reference/en/ or as ctors for this course.



Exercise 5.1: RGB LED ring

In this exercise you are going to program a LED-ring. To get you started there is an example code at the course page. The aim of this exercise is to learn how to install an external code library to control the LED's and create your own RGB-effects.

Hardware Required

- Arduino UNO R3
- LED-strip
- Breadboard
- Cables

Circuit

Connect the LED-strip to your breadboard. From previous exercises you should have an idea of how the LED-strip should be connected. The LED-strip is marked with 5v for power, DIN for the signal and GND for ground. The corresponding cables are red for power, black for ground and yellow for data. Make sure the arduino is powered off when making all connections.

Installing the library

A library is a collection of functions you may use for different applications. For this application, you are going to need to download and install the Neopixel library which is used to control the type of LED's that we're using. In Arduino IDE go to *Sketch/Include -> library/Manage Libraries.* Search for Adafruit Neopixel and select the library named "Adafruit NeoPixel. Make sure it's the version description is *"arduino library for controlling single-wire-based LED pixels and strip". Once installed you should* restart your Arduino IDE after installation.

Test your installation: Example code

If the installation was successful you should now be able to load the example sketch “strandtest”. You will find it under *File/Examples/Adafruit Neopixel.* Select it and upload it to your board. If you have connected the wires correctly (Check your data pin) the LED’s should light up.

What is RGB anyway?

RGB refers to the color channels used by the LED’s to create all combinations of colors using only 3 primary colors - Red, Green and Blue. It is basically your digital color pallette where you can mix and match to get whatever color you like.

Remember the analog write value 255 we used in previous exercises? If you look in the example code you should see something like:

strip.Color(255, 0, 0)

The LED’s are controlled by setting the intensity of each color channel. A value of 0 means the chanel is off, a value of 255 means the channel is at it’s max intensity. The small code block above represents the color red, as the red channel is set to max intensity and the green and blue channels are set to off.

(0,255,0) = Green
(0,0,255) = Blue

Experiment with the example code

Look around in the example codes and try and figure out the different functions, rainbow, sweep etc. Change some color values and try to make your own special effect.

TIP: make sure you set the number of LED’s in the code to be the same as your connected LED-strip.

Exercise 6.1: custom sensor

In this exercise you will learn what you can do when you're bored.

Hardware Required

- Arduino UNO R3
- A sensor of your choice

Circuit

In this exercise you get to play with some sensors and make something fun, be creative.

There's an assortment of extra sensors, choose one and start coding. For some sensors there are examples in the arduino ide, for some there are not. If you need more information about the sensor, feel free to ask an instructor.

Code

You can find information about the arduino and its libraries on https://www.arduino.cc/reference/en/ or ask one of the instructors for this course.

Useful information

There isn't any useful information.