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

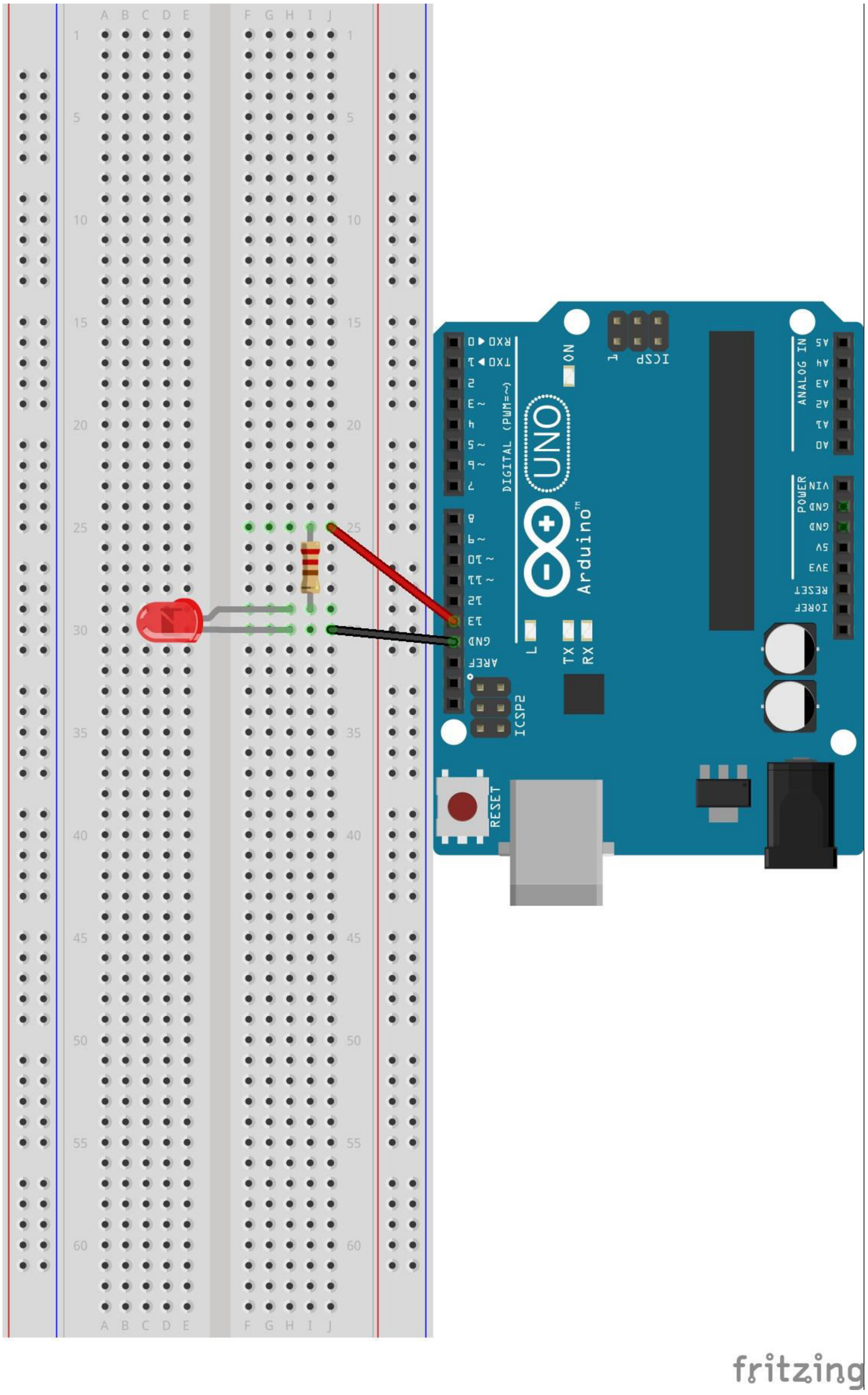
330 ohm resistor

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*.

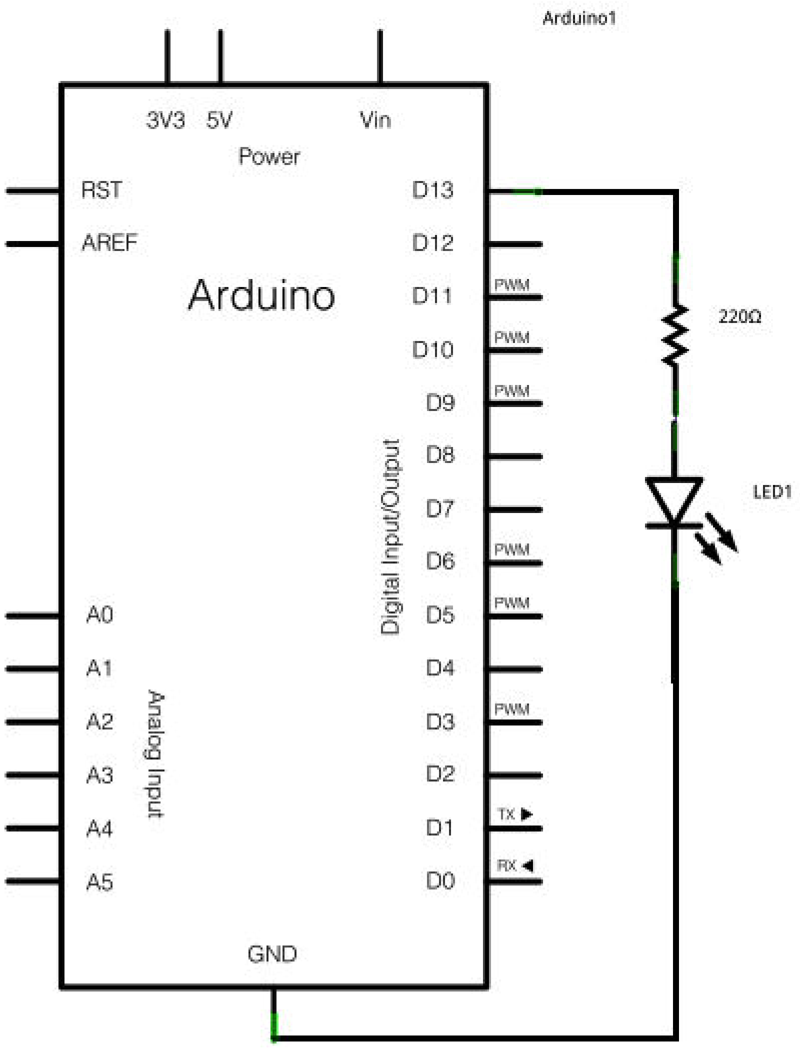
If you want to light an external LED with this sketch, you need to build this circuit, where you connect one end of the resistor to the digital pin correspondent to the *LED_BUILTIN* constant. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect 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.



A B C D E
F G H I J
1
5
10
15
20
25
30
35
40
45
50
55
60
DIGITAL (PWM=~)
RX0 ◀ 0
TX0 ▶ 1
AREF
GND
13
12
~11
~10
~9
8
L
TX
RX
ON
ICSP
ICSP2
UNO
Arduino™
ANALOG IN
A0
A1
A2
A3
A4
A5
POWER
VIN
GND
5V
3V3
RESET
IOREF
fritzing

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 output before 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

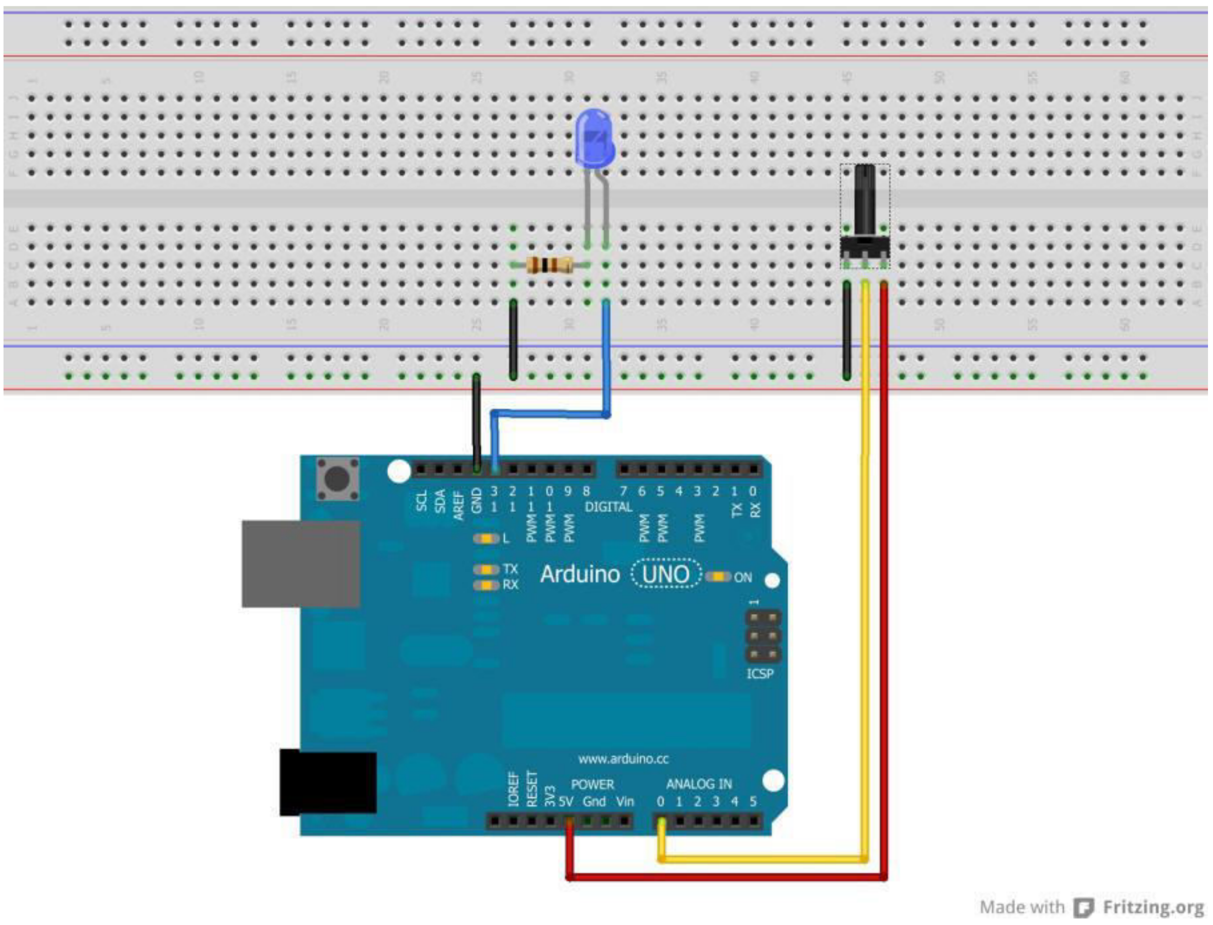
330 ohm resistor

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 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.

Hardware Required

Arduino UNO R3

LEDs 2pcs

330 ohm resistors 2pcs

Potentiometer with knob

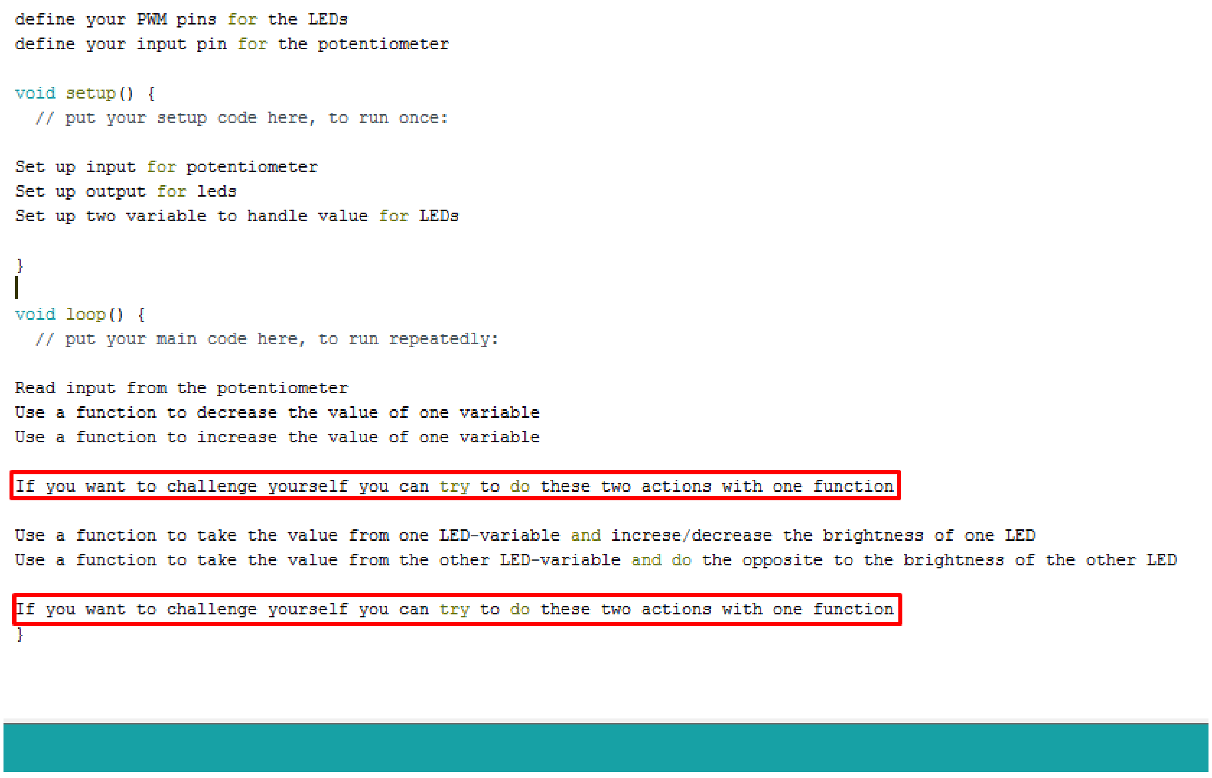
Circuit

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: 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

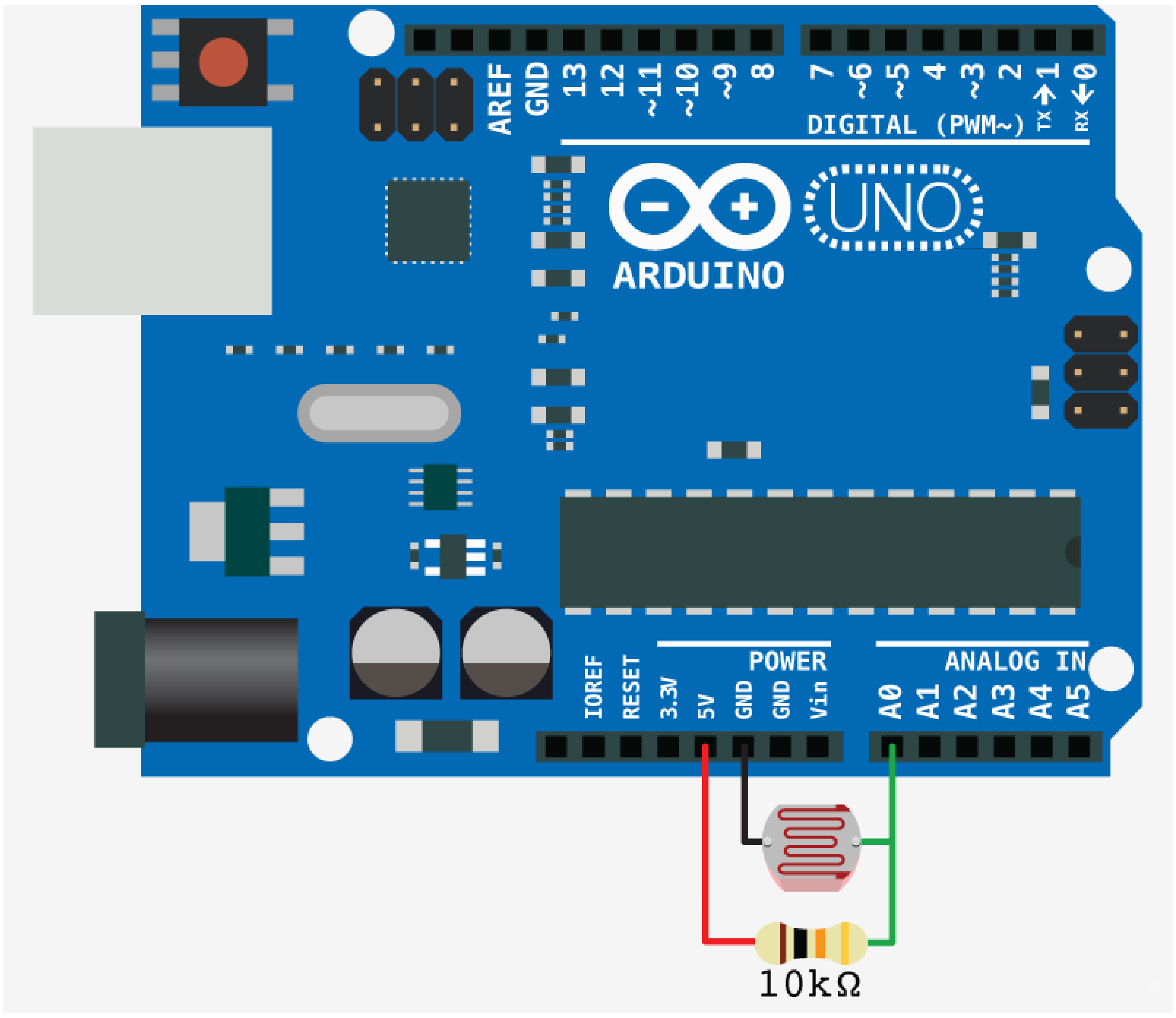
LDR

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 3.2: 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 3.3: 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 ask one of the instructors for this course.

Exercise 4.1: shift register

In this exercise you will learn what you can do when you run out of pins on an arduino.

Hardware Required

Arduino UNO R3

LEDs 8pcs

330 ohm resistors 8pcs

Shift register

0.1μF capacitor

Button

Potentiometer

Circuit

In this exercise you will get help with the circuit and some of the basic code to make the shift register work.

You can either:

1. Control the leds through serial monitor
2. Use a button to make leds light up the way you want
    a. Make one successive led light up each time you press the button. When you get to 8 turn all off or turn one off successively.
    b. Make a bit counter 1 = 1000000, 2 = 01000000, 3 = 11000000, and so on
3. Use a potentiometer to either make a pattern or to increase the “bit counter” the more you turn the potentiometer.

Code

You will get some basic example code so that you know what to do and how to use the shift register. It’s up to you to decide what you want to try.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

If you don’t know about binary numbers here’s some useful information.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Each number used becomes a one and the rest becomes zeros. You can use a decimal to binary converter like https://www.rapidtables.com/convert/number/decimal-to-binary.html if you don’t want to calculate the number.

The binary strings sent to the shift register is an 8 bit string so it contains any combination of 8 bits. It is easier in practise, and explained in more detail further down. To light up a LED(make a pin HIGH) we send a one and to turn one off we send a zero.

Binary	Decimal
0000 0001	1
0000 0010	2
0000 0011	3
0000 0100	4

0000 0101	5
0000 0110	6
0000 0111	7
0000 1000	8
0000 1001	9
0000 1010	10
0000 1011	11
0000 1100	12
0000 1101	13
0000 1110	14

When using a shift register there’s some basics you need to know. We use 3 pins on the arduino to control the 8 outputs/inputs of the shift register.

Clockpin:

Is used to tell a component like a shift register when a byte has been sent to make sure it is saved. When we send the information the clockpin is low and when a bit has been sent the clockpin should be set to high to confirm that value.

Datapin:

Is used to send the data to the shift register, telling it which pins that should go HIGH when the latchpin goes HIGH.

Latchpin

When this pin is enabled(set to high) the shift register sets ones to HIGH and zeros to LOW for its outputs.

If we want to make it easy for ourself we can use the function shiftOut(), more information on: https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftout/

Exercise 4.2: more shift register

In this exercise you will learn what you can do when you run out of pins on an arduino.

Hardware Required

Arduino UNO R3

LEDs 8pcs

330 ohm resistors 8pcs

Shift register

0.1µF capacitor

Button

Circuit

In this exercise you will use the circuit from the last exercise, including a button.

The goal in this exercise is to make a stopwatch. If you want to make it easy for yourself you can make a binary stopwatch using 8 leds. If you want to make it a challenge you can use a 7-segment display.

Code

You will get some basic example code so that you know what to do and how to use the shift register. It's up to you to decide what you want to try.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

Here’s some help to get you started with the stopwatch. You can use the function millis() to count the time from when the button is pressed and then you can “push” out the values to the shift register to be displayed on the LEDs or 7-segment.

https://www.arduino.cc/reference/en/language/functions/time/millis/

Exercise 6: 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.