

# ROB - Basics lab

Adam Fabo

Brno University of Technology, Faculty of Information Technology  
Božetěchova 1/2. 612 66 Brno - Královo Pole  
[xfaboa00@fit.vutbr.cz](mailto:xfaboa00@fit.vutbr.cz)

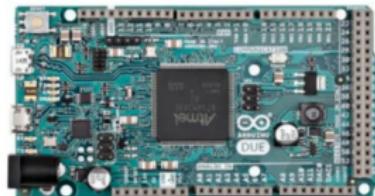


February 6, 2023

Today's lab will be focused on **Basics of Arduino**. You will learn

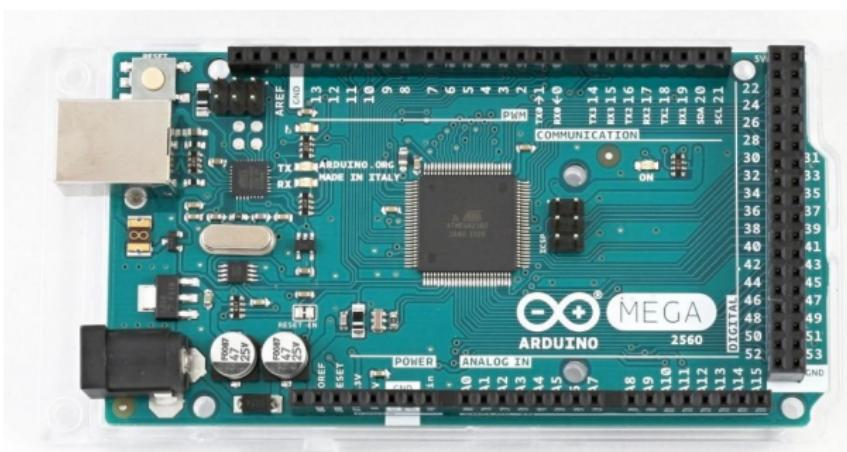
- Creating and uploading code for Arduino
- Prototyping electronic circuits using breadboard
- Reading and Writing **digital** signal
- Reading and Writing **analog** signal

- Arduino is set of open source hardware and software projects used for fast prototyping of electronics circuits
- Lots of boards - Arduino Mega, Arduino Uno, Arduino Nano, Arduino Micro, Arduino Zero ...
- Supports a lot of kits, shields, sensors, tools, accessories.
- <https://www.arduino.cc>

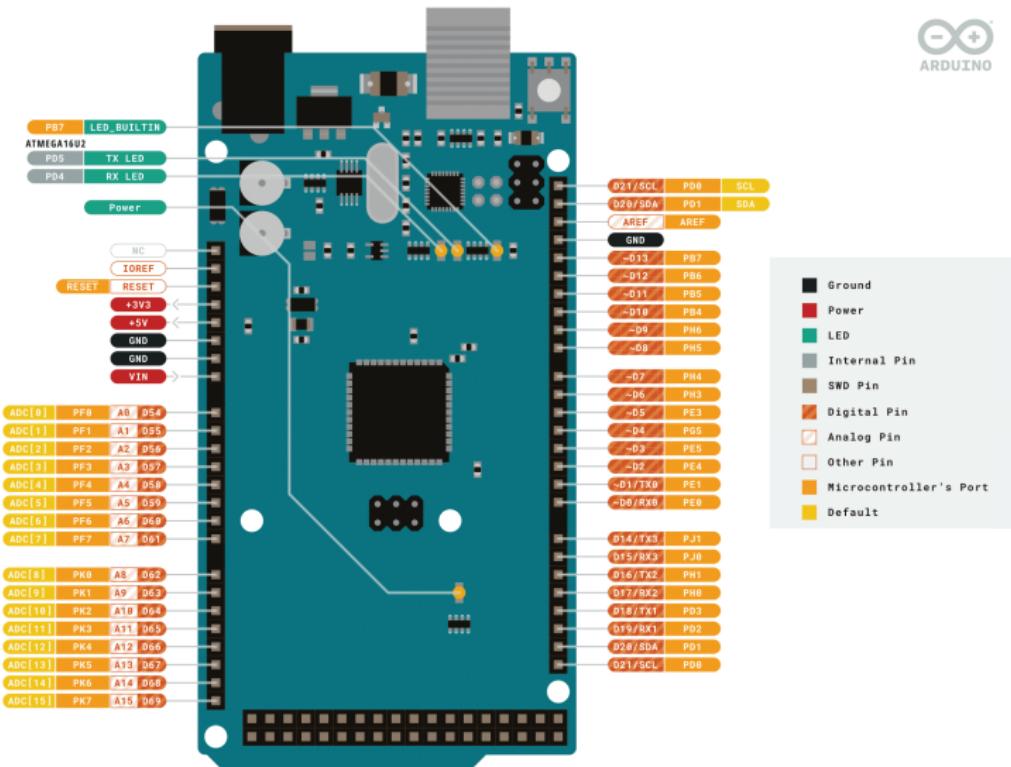


Arduino board used in this lab is [Arduino Mega](#). It consists of:

- 16MHz ATmega2560 processor
- 54 Digital I/O Pins (of which 15 provide PWM output)
- 16 Analog Input Pins
- 256 KB Flash, 8 KB SRAM, 4 KB EEPROM
- Built in LED at pin 13



- <https://store.arduino.cc/products/arduino-mega-2560-rev3>



- For programming Arduino boards, you need to know [c/c++](#)
- <https://www.arduino.cc/reference/en/>

// If condition

```
int x = 0;  
if(x == 10){  
    do_something();  
}else{  
    do_something_else();  
}
```

// For loop

```
for(int i=0; i < 10; i++){  
    do_something();  
}
```

// While loop

```
int i = 0;  
while(i < 10){  
    do_something();  
    i++;  
}
```

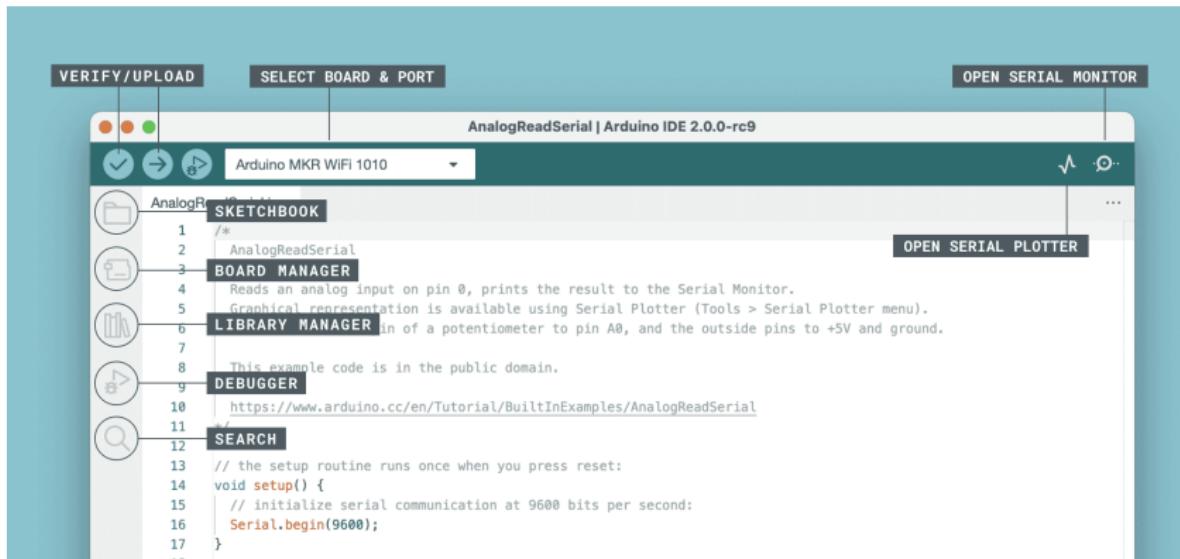
- Arduino IDE is used for programming Arduino boards.
- Old looking but well functional
- Good for fast prototyping, but harder to use for bigger projects.

The screenshot shows the Arduino IDE interface. On the left, the 'VERIFY/UPLOAD' window displays the code for the 'Blink' example sketch. The code is as follows:

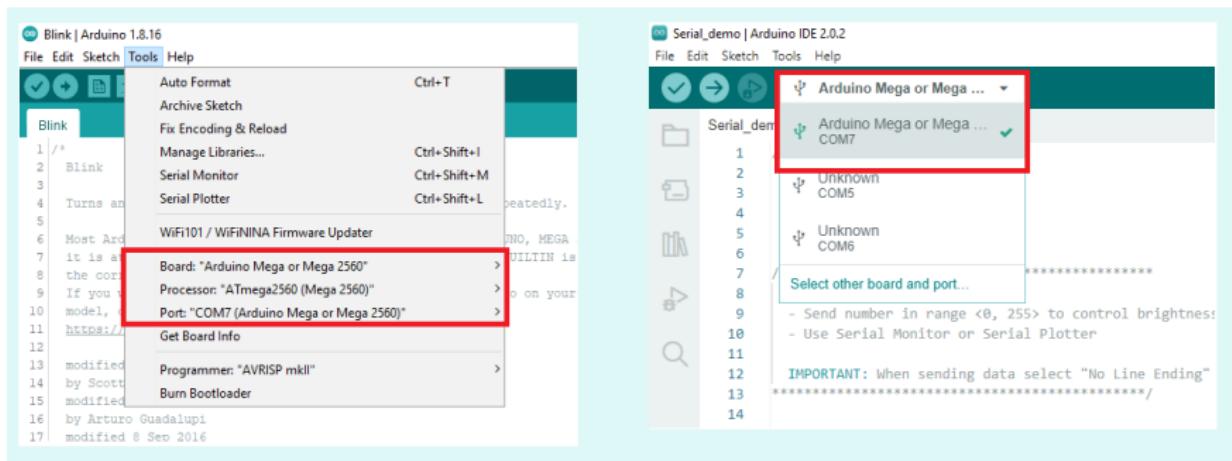
```
1 // 
2 // Blink
3 //
4 // Turns an LED on for one second, then off for one second, repeatedly.
5 //
6 // Most Arduinos have an on-board LED you can control. On the Uno, Mega and ZERO
7 // it is attached to digital pin 13, on MEGA2560 on pin 6. LED_BUILTIN is set to
8 // the correct LED pin independent of which board is used.
9 // If you want to know what pin the on-board LED is connected to on your Arduino
10 // model, check the Technical Specs of your board at
11 // https://www.arduino.cc/en/Main/Products
12 //
13 // modified 8 May 2014
14 // by Scott Fitzgerald
15 // modified 2 Sep 2016
16 // by Arturo Guadalupi
17 // modified 8 Sep 2016
18 // by Colby Neeman
19 //
20 // This example code is in the public domain.
21 //
22 // https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
23 */
24 //
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
```

On the right, the 'OPEN SERIAL MONITOR' window is visible, showing a blank white space where serial data would typically be displayed.

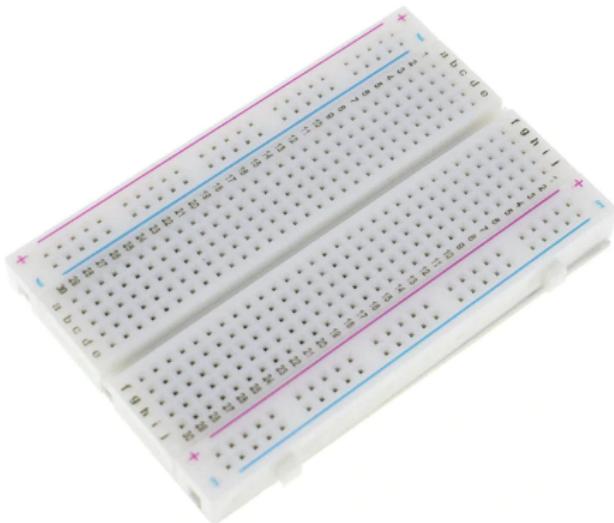
- Better looking than classic Arduino IDE
- Newer but still little bit buggy



- Before uploading, check if there is correct Port and Board selected. Otherwise, you won't be able to upload code to Arduino.
- For this lab, you should have selected **Arduino Mega**

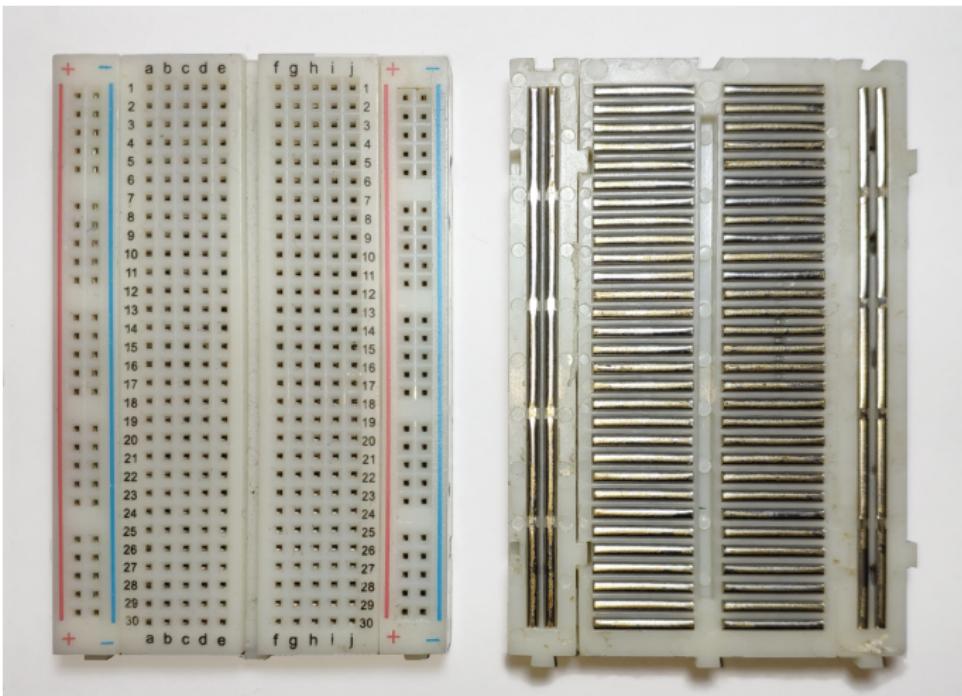


- Breadboard is used for prototyping electrical circuits
- Does not need soldering
- Every standard "trough hole" component can be fit into breadboard
- Does not support SMD



# Breadboard - How it works

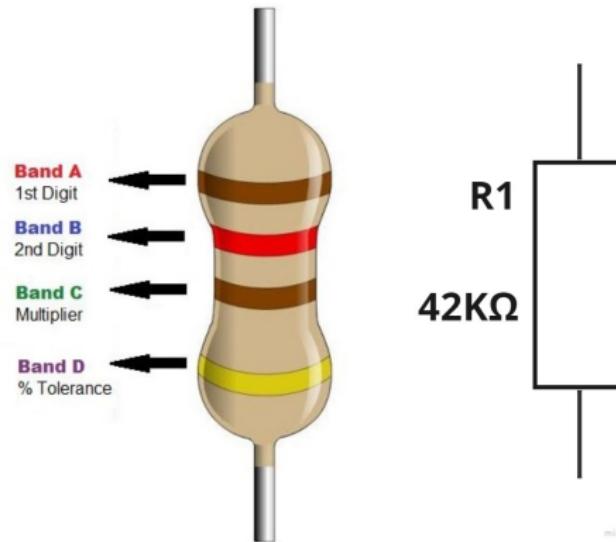
- Breadboard consists of 2 types of rails: Horizontal and Vertical
- Horizontal rails are used for any electrical parts
- Vertical rails are used for power supply and ground



Parts used in this lab:

- Resistor
- Variable resistor (Potentiometer)
- LED - Light Emitting Diode
- Button

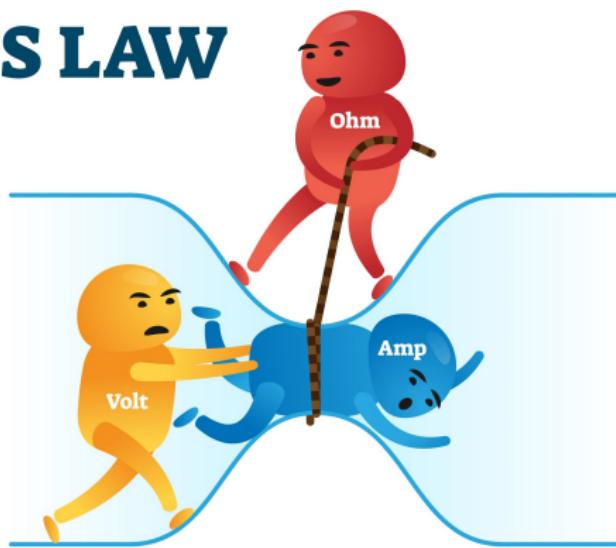
- Passive electrical component
- Limits flow of a current
- Unit of ohm ( $\Omega$ )



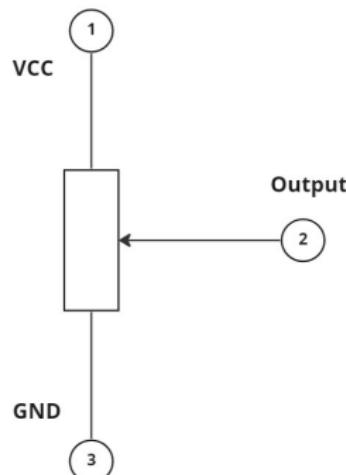
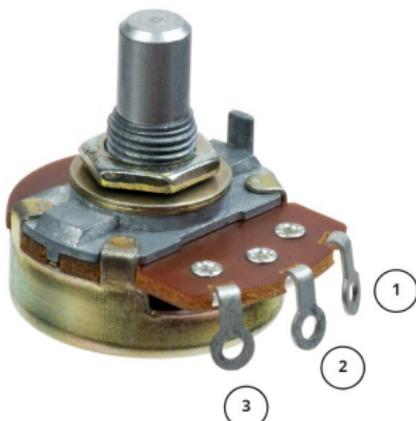
mirro

- Basic electrical law
- $U = R * I$
- Current flow through a conductor between two points is directly proportional to the voltage across the two points

## OHM'S LAW

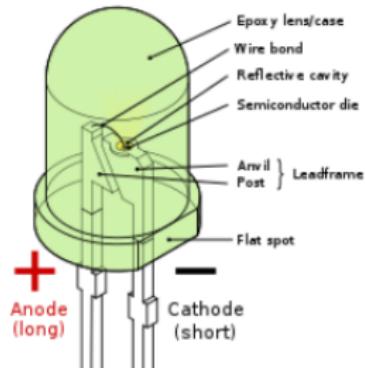
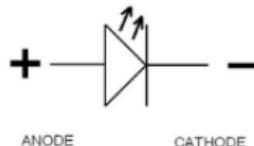
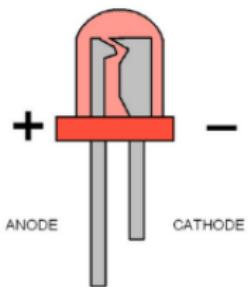


- Passive electrical component
- Same as resistor but its resistance can be changed
- Usage - for example volume control



- Light Emitting Diode - diode that emits light when current flows through it
- Diode allows flow of current only in one direction
- 2 pins: positive **Anode** and negative **Cathode**
- Usage - light indication

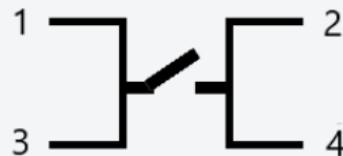
## What is a LED?



- Simple electrical component
- Connects or disconnects electrical circuit
- Usage - ON/OFF button



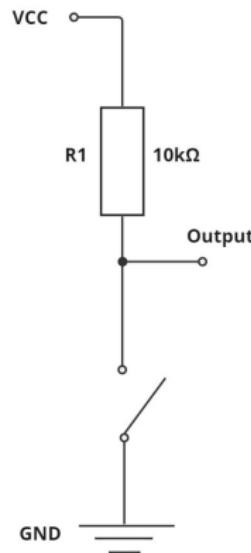
Pin 1 & 3 are connected  
Pin 2 & 4 are connected



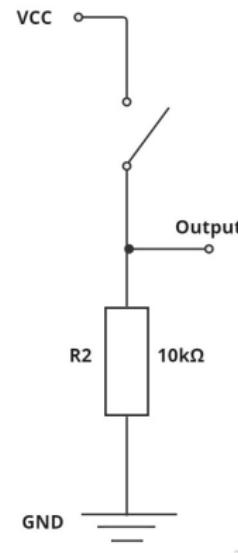
# Pull-up / pull-down resistor

- Used to ensure a known state for a signal
- Pull-up ensures default Logic **1** signal and Pull-down ensures logic **0**

Pull-up Resistor



Pull-down Resistor



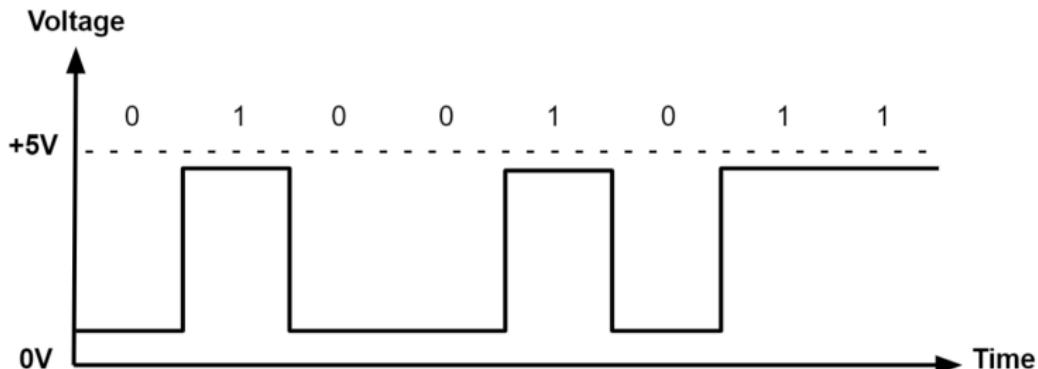
miro

There are 3 types of signal that will be explored in this lab:

- Digital signal
- PWM
- Analog signal

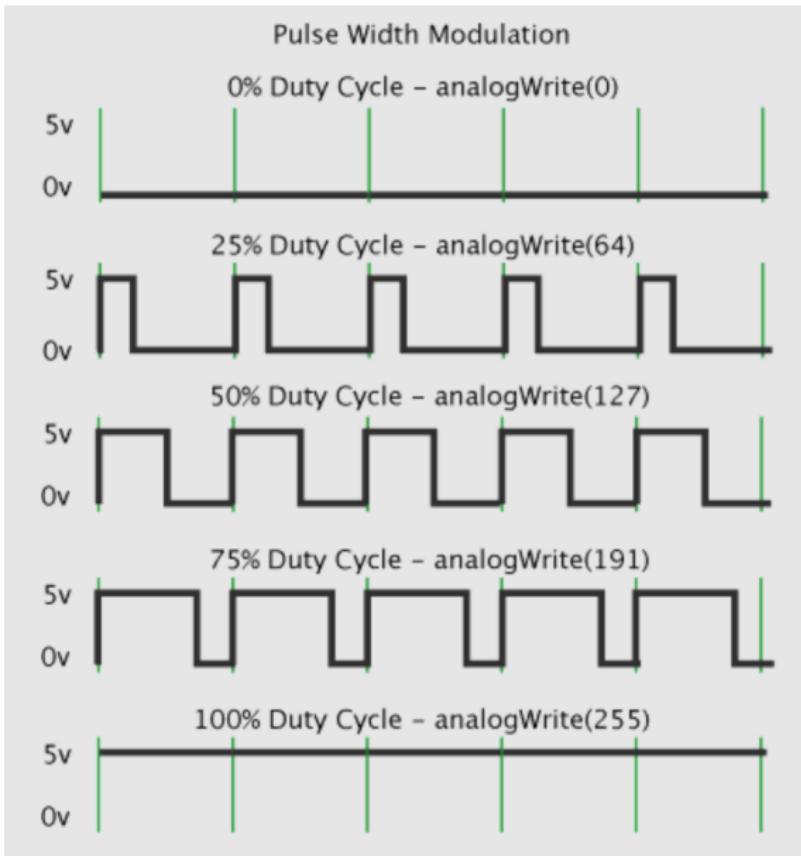
Digital signal consists of 2 possible states

- 1 - High - 5 Volts
- 0 - Low - 0 Volts



PWM or Pulse Width Modulation is

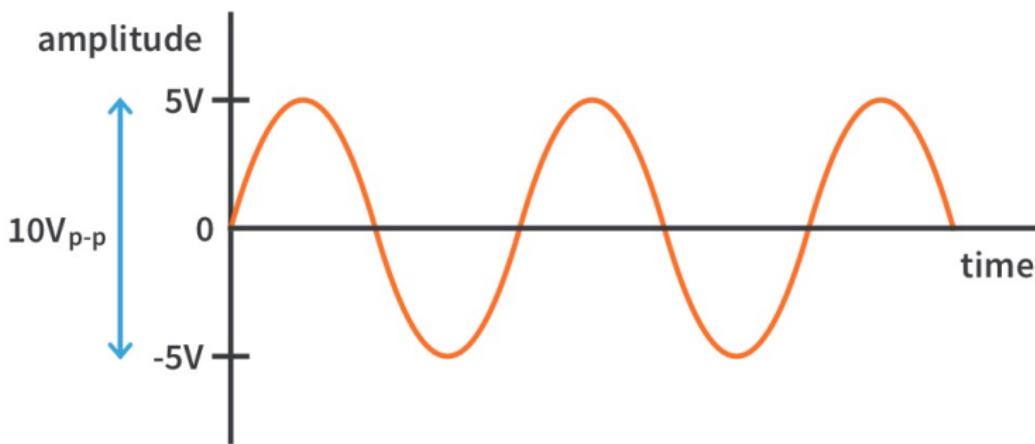
- method of reducing the average power delivered by an electrical signal
- Uses a repeating digital signal
- 2 attributes - **frequency** (or period) and **duty cycle**



# Analog signal

Analog signal is a continuous signal that can have any value from a given range

- Range in volts for Arduino: **<0V, 5V>**
- Arduino uses a 10 bit analog to digital converter to read the value
- Read value is in range **<0, 1023>**



Each Arduino code consists of 2 functions:

- **void setup()** - code in this function will run only once, used for setup
- **void loop()** - code in this function will run infinitely, place where you write main part of your code

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8  
9 }  
10  
11
```

- Before reading/writing signal from/to a pin there is need for that pin to be initialized
- `pinMode(pin, INPUT/OUTPUT)`
- Sets up given pin to be input or output
- Use this function before using any read or write functions
- Usually used in `setup()` function

- `digitalWrite(pin, HIGH/LOW)` - writes digital signal to given pin
- `digitalRead(pin)` - reads digital signal from given pin, returns its value 1 or 0

- `analogWrite(pin, value)` - writes analog value to given pin
  - Value can be from range **<0, 255>** which corresponds to range in voltage **<0V, 5V>**
  - Uses PWM
- 
- `analogRead(pin)` - reads analog value from given pin
  - Return value can be from range **<0, 1023>** which corresponds to range in voltage **<0V, 5V>**

# Exercises

Following exercises will focus on following:

- Creating electrical circuits on breadboard
- Outputting digital signal
- Reading digital signal
- Creating PWM (analog output)
- Reading analog signal
- Measurements with oscilloscope

Check if your Arduino is "Alive" and upload your first code.

- Open file [Blinking\\_LED.ino](#)
- Take a look at the code
- Specify correct port and board in Arduino IDE in tools-board and tools-port
- Upload the code by clicking at  or pressing **Ctrl+u**
- Built in LED on Arduino board should be blinking in 1 second intervals

Make your own circuit with LED on breadboard.

- Open file [Blinking\\_LED\\_Breadboard.ino](#)
- Take a look at scheme [LED\\_scheme.png](#) and create circuit as it shown at the image
- Upload the code by clicking at or pressing **Ctrl+U**
- LED should be blinking in one second intervals

- Continue working with the same code and scheme.
- Modify the time intervals that are stored in constants `TIME_ON` and `TIME_OFF` as it is described in table below
- After each change upload code to Arduino
- What behavior of the LED do you observe?

	TIME_ON	TIME_OFF
1.	100	100
2.	30	30
3.	1	1
4.	1	10
5.	1	20

Create PWM waves using knowledge from the last 2 exercises.

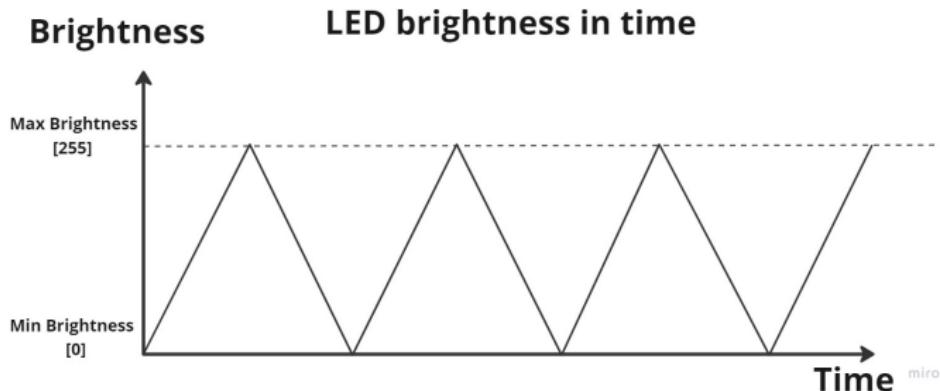
- Open file [PWM\\_custom\\_method.ino](#)
- Reuse circuit from previous exercise
- Upload the code
- Try and modify constant `DUTY_CYCLE_PERCENTAGE` and set it to values 1, 2, 5, 10, 100
- Observe behavior of the LED
- How would you do it, if you would have to blink two LEDs with different duty cycles?

Use built-in Arduino function to create PWM wave.

- Open file [PWM\\_proper\\_method.ino](#)
- Reuse circuit from previous exercise
- Upload the code
- Try and modify integer **brightness** and set it to values 0, 1, 5, 10, 50, 100, 255
- Observe behavior of a LED

Write a code that will continually brighten and dim a LED

- Open file [PWM\\_fading\\_LED.ino](#)
- Reuse circuit from previous exercise
- Follow the instructions and implement brightening and dimming of the LED
- Upload the code
- Test your implementation by uploading your code and observing LED



Control LED using serial communication.

- Open file `Serial_Demo.ino`
- Reuse circuit from previous exercise
- Upload the code
- Open Serial Plotter (or Serial Monitor), select "No Line Ending" and send values in range <0,255> to control brightness of the LED

Sending signals is only half of the knowledge. The other half is to know how to read them.

- Open file [Serial\\_Demo.ino](#)
- Take a look at scheme [Button\\_scheme.png](#) and create the circuit as it shown in the image
- Upload the code
- Press the button and observe the behavior using Serial Plotter (or Serial Monitor)

Read analog signal from potentiometer.

- Open file [Serial\\_Demo.ino](#)
- Take a look at scheme [Potentiometer\\_scheme.png](#) and create the circuit as it shown in the image
- Upload the code
- Rotate the potentiometer and observe the behavior using Serial Plotter (or Serial Monitor)

Use an oscilloscope to read Arduino output signal.

- 1 Reuse any code that controls LED
- 2 Take a look at scheme [LED\\_Oscilloscope\\_scheme.png](#) and create circuit as it shown at the image
- 3 Upload the code
- 4 Press **autoSet** and wait for the oscilloscope to set
- 5 Take a look at the measurements taken by the oscilloscope
- 6 Rotate knobs that are used for horizontal and vertical scaling

Thank You For Your Attention !