Introduction to Arduino Prototyping

Introduction to Electronics and Arduino

Nicola Dall'Ora

Sebastiano Gaiardelli

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

ISD
Cyber-Physical & IoT Systems Design

Support

- On the new e-learning portal:
  - Support material
  - Forum for Q&A

Exercises and questions are a good training to get ready for the exams!

 Questions

- Office hour:
  - Ca' Vignal 2, Floor 1, Room 71
  - **Always "on demand"** after arranging a meeting **via email!**
    - **2 working days in advance:**
      - **Ask Thursday for Monday, Ask Monday for Wednesday**
      - **UNIVR emails only, addressed to:**

nicola.dallora@univr.it

sebastiano.gaiardelli@univr.it

Outline

- Objectives
- Basic concepts of electronics
- Useful circuits connections
- Course materials
- Prototyping Boards
- Analog and Digital Signals
- Practical Examples
  - Light Emitting Diode (LED)
  - Button
  - Buzzer
  - Temperature sensor (DS18B20)
  - Current sensor (INA219)

Objectives

Lecture Objectives

1. Providing a **basic** knowledge of **electronics**;
2. Providing a clear view of the **components** we are going to use from the course;
3. **Setup** the design and development software;
4. **Design** and develop the first **example** setups with Arduino.

Basic concepts of elctronics

Let there be light

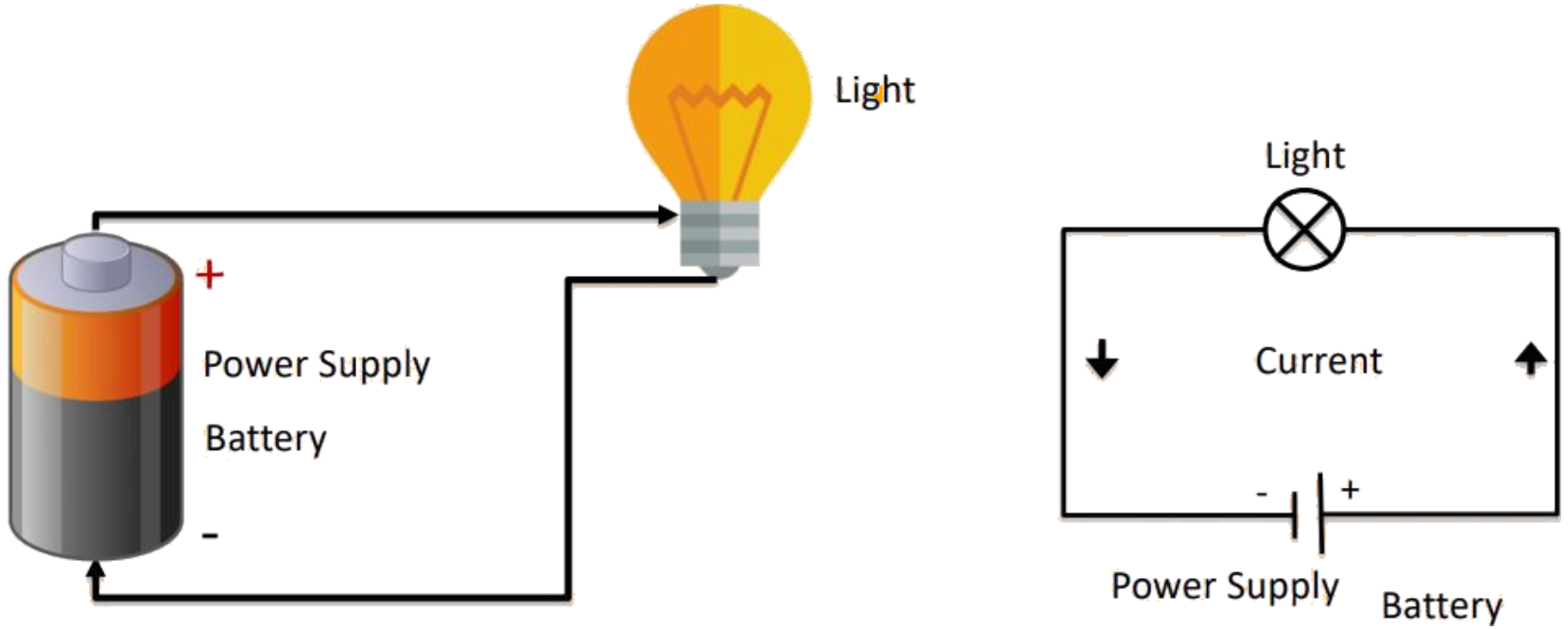


Figure: A simple circuit with a power source and a light bulb.

Let there be light

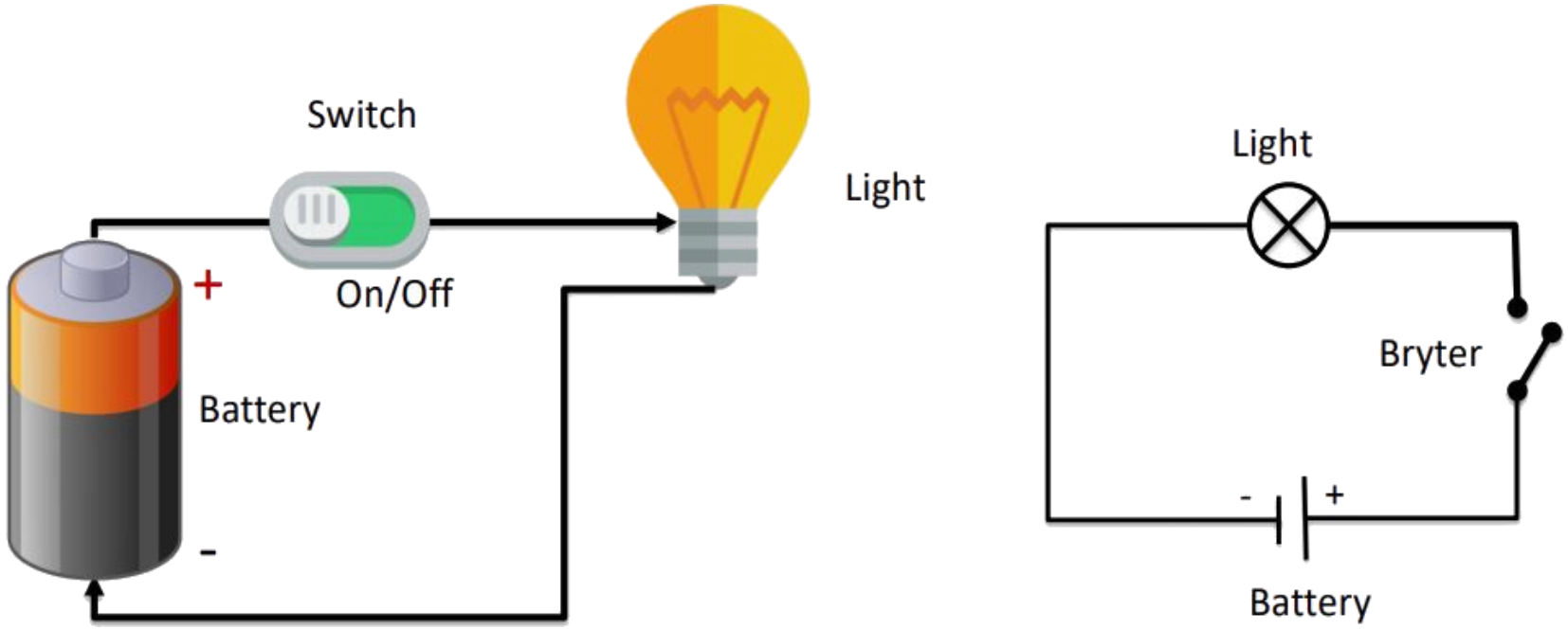


Figure: A simple circuit with a power source, a light bulb, and a switch that allows the user to activate/deactivate the light bulb.

Let use try with an analogy

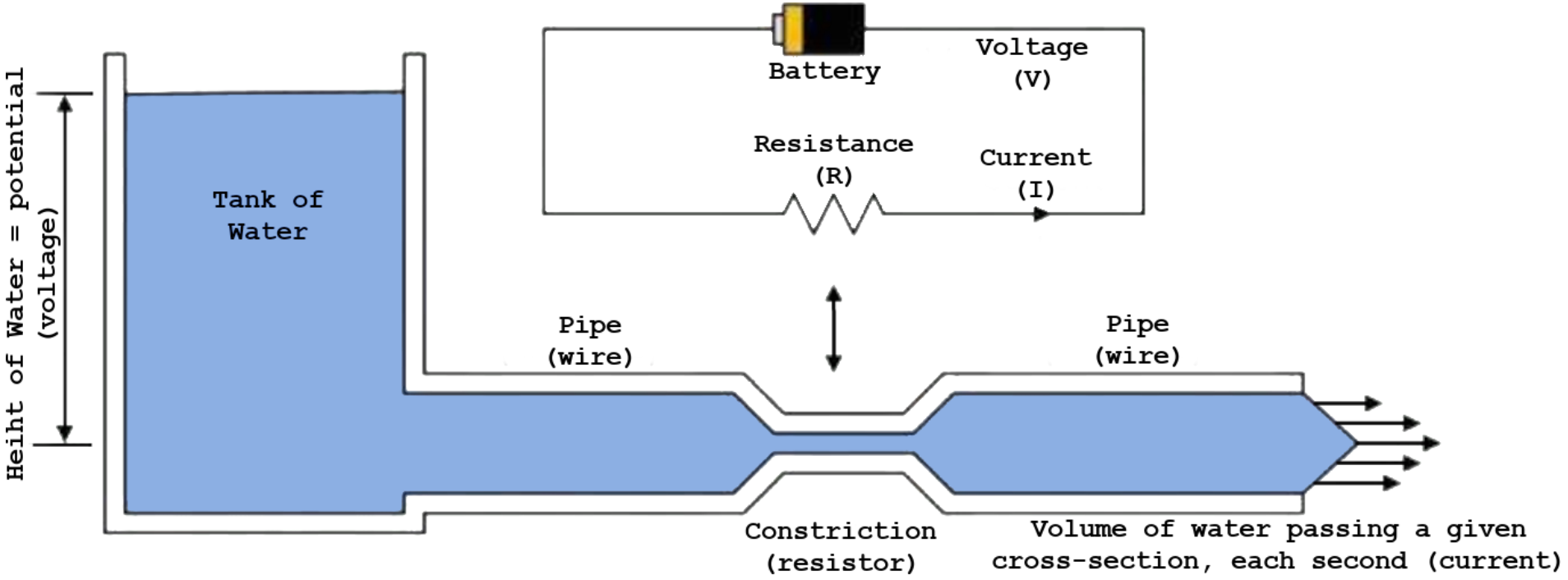


Figure: Electronic–hydraulic analogy, explains electrical current and Ohm's Law (V=I/R).

In this analogy, water **pressure** is analogous to electrical pressure or **voltage**.
Water **flow** corresponds to electrical **current**, the flow of electrical charge.
Electrical **resistance** is analogous to a **constriction** in the water flow path, such as the inverse size of a water pipe. The smaller the pipe, the greater the resistance to water flow. It's the same with the diameter of a wire.

Main electrical components

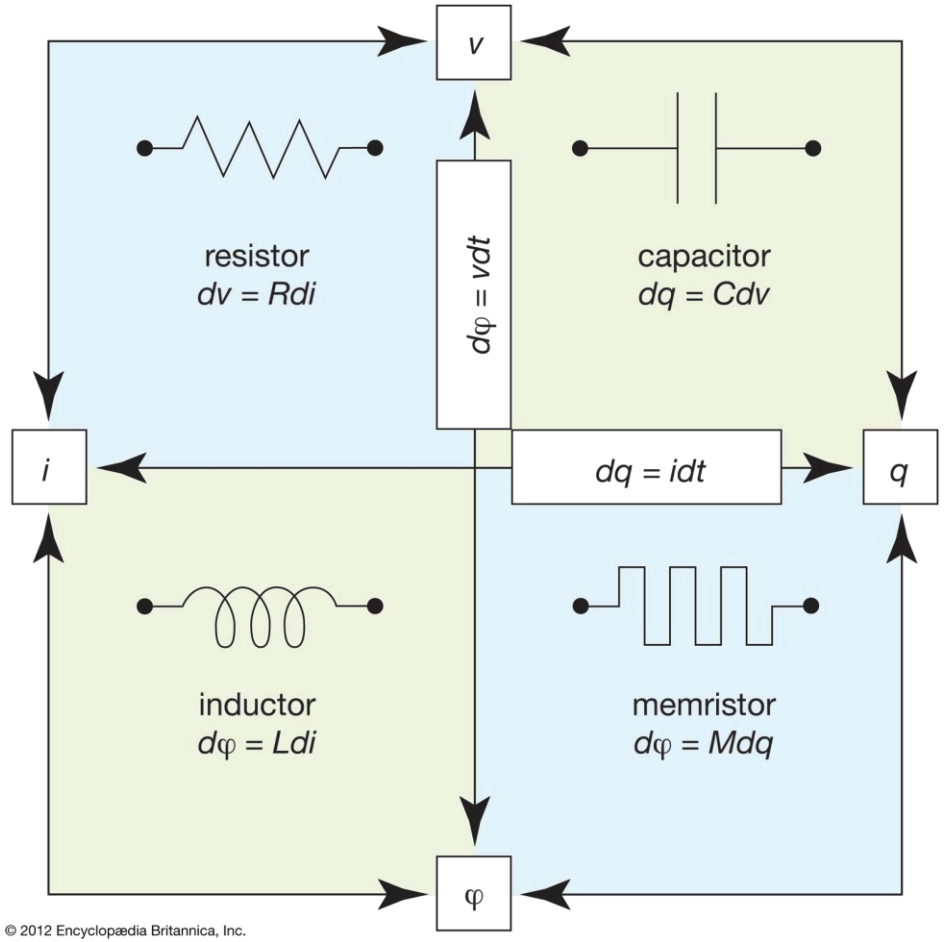


Figure: The four fundamental passive electrical components, i.e., those that do not produce energy. The four cardinal symbols stands for electric current (I), voltage (V), charge (q), and magnetic flux (φ or ϕ).

Other circuit components

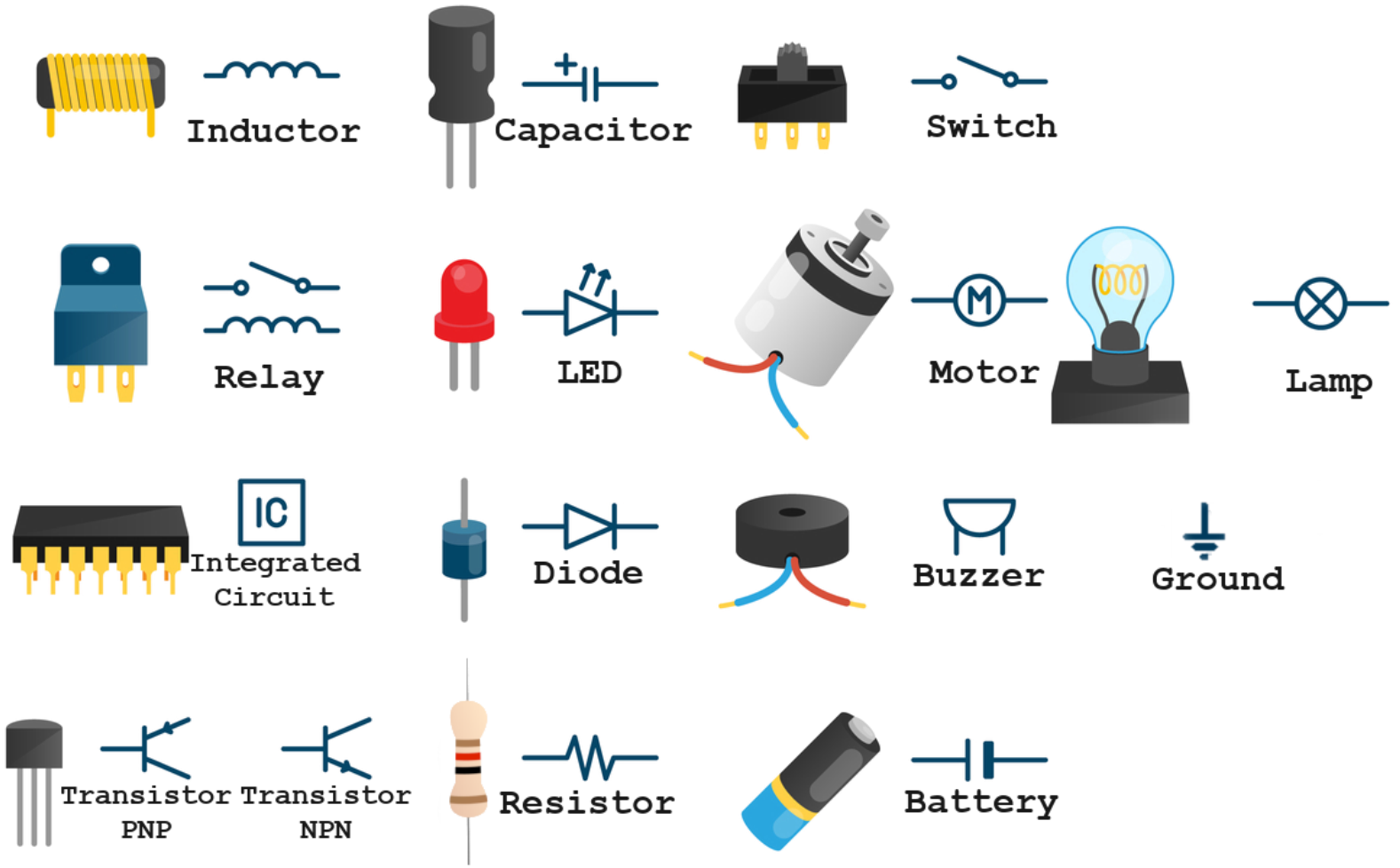


Figure: A common set of circuit components.

Useful circuits connections

Resistor (Ohm's law)

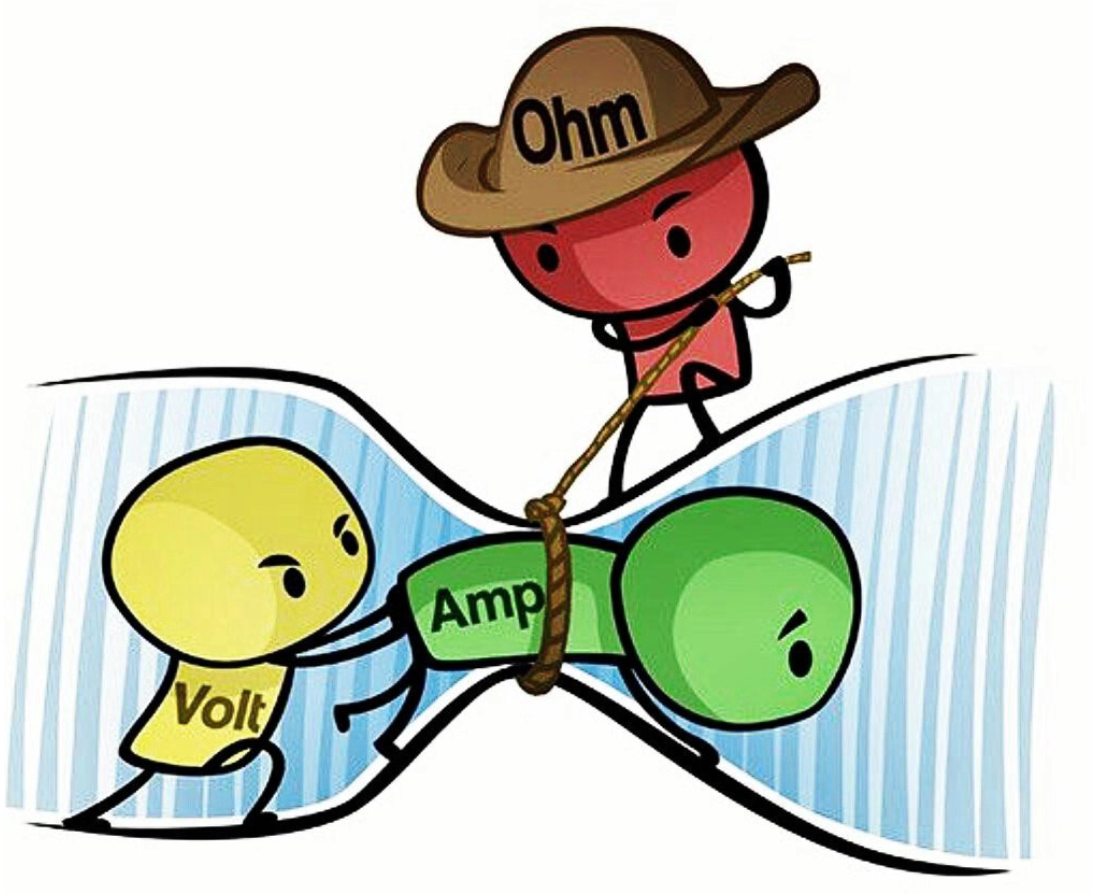


Figure: Ohm's law states that the current through a conductor between two points is directly proportional to the voltage across the two points.

Resistor (Color codes)

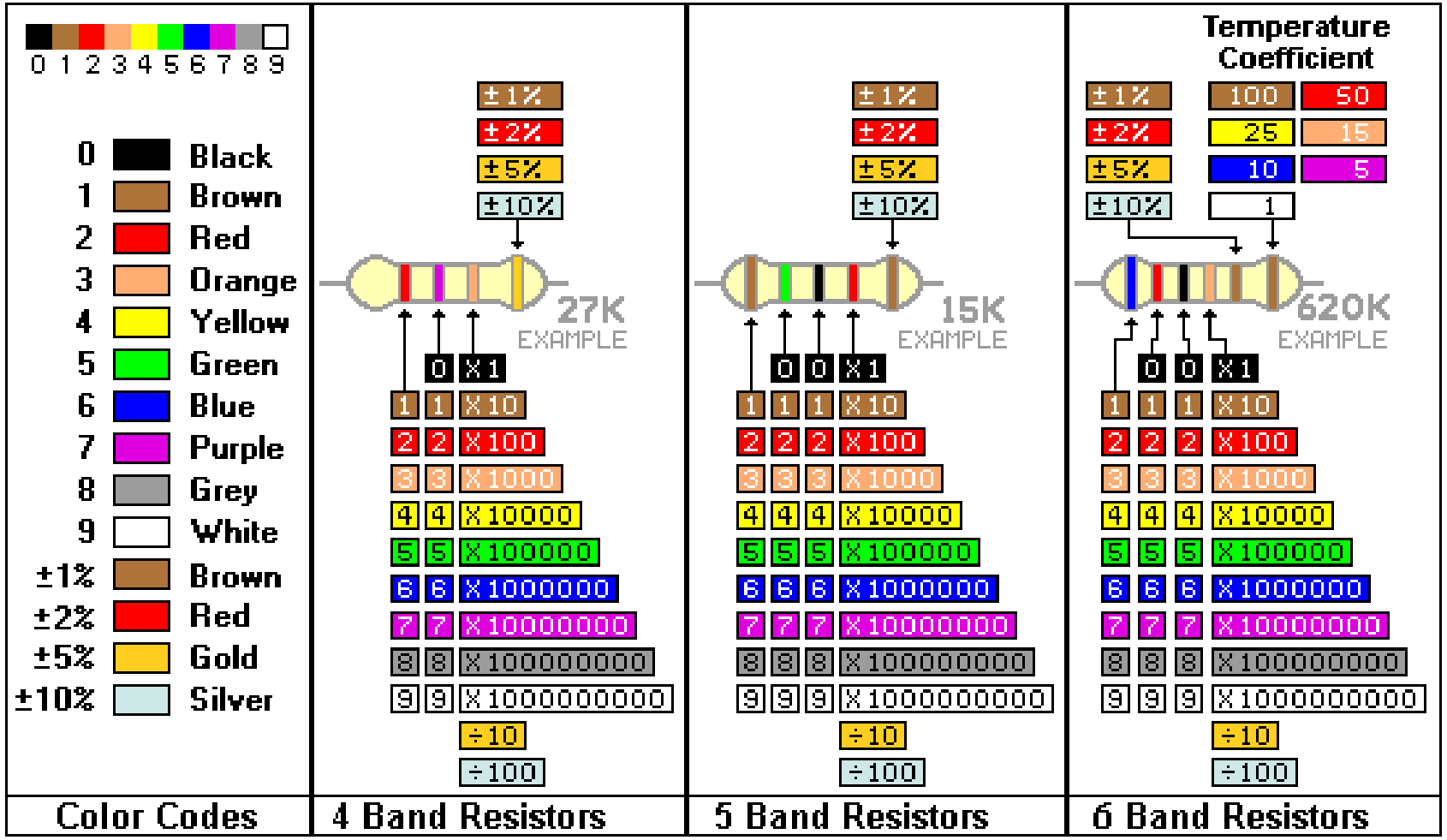
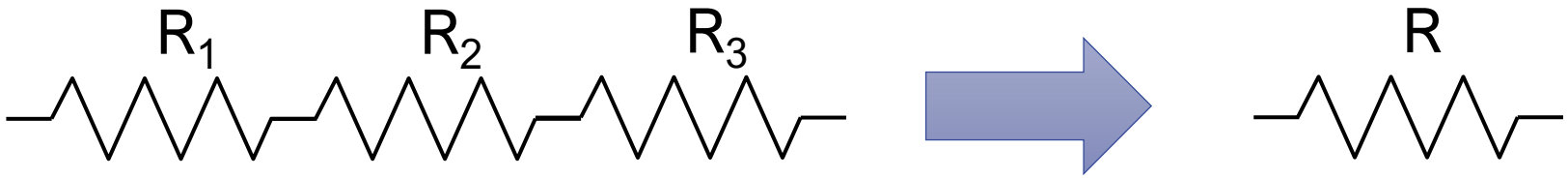


Figure: The three main type of resistors, and the table for reading them.

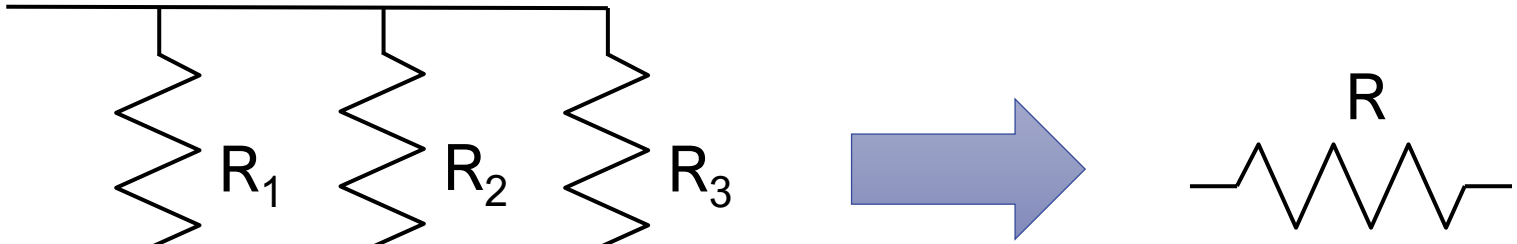
Resistor (Series and Parallel)

Series resistors



$$R = R_1 + R_2 + R_3 + \ldots$$

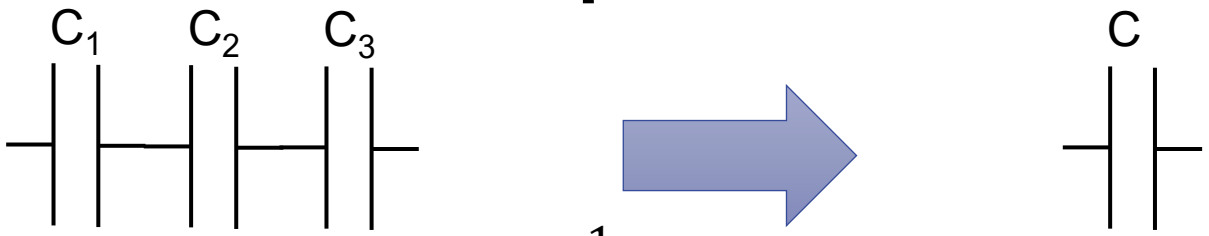
Parallel resistors



$$R = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

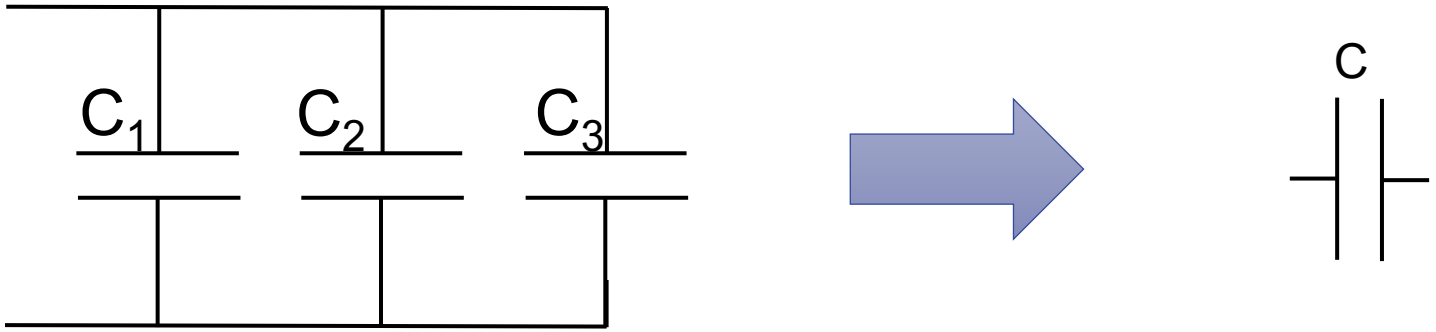
Capacitor (Series and Parallel)

Series capacitors



$$C = \frac{1}{\frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3}}$$

Parallel capacitors

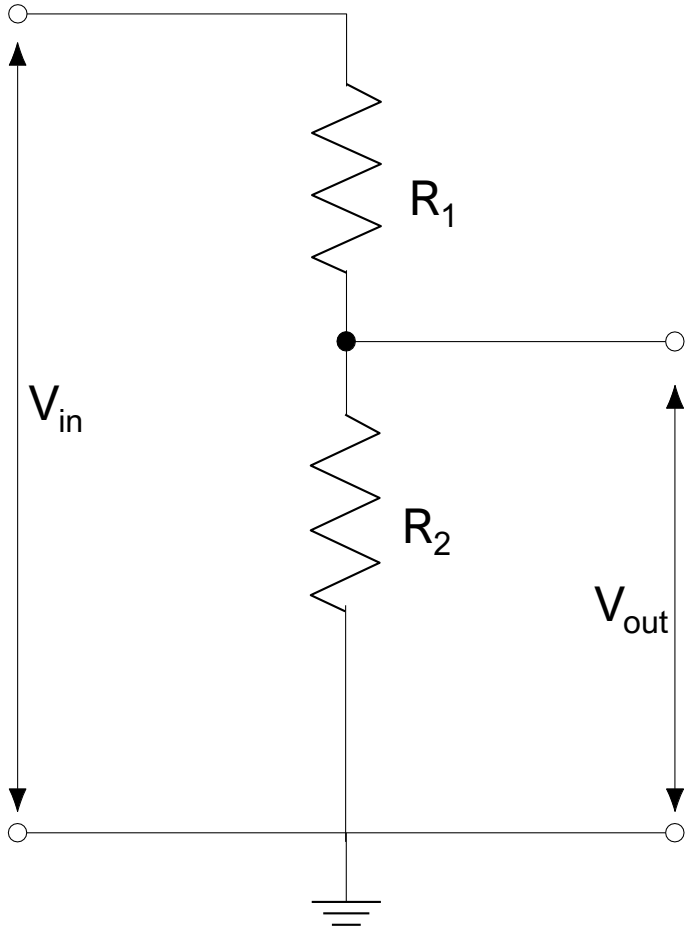


$$C = C_1 + C_2 + C_3 + \ldots$$

Voltage divider

The **two-resistor voltage divider** is often used to supply a **different voltage** from that of an available battery or power supply.

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



https://ohmslawcalculator.com/voltage-divider-calculator

Course materials

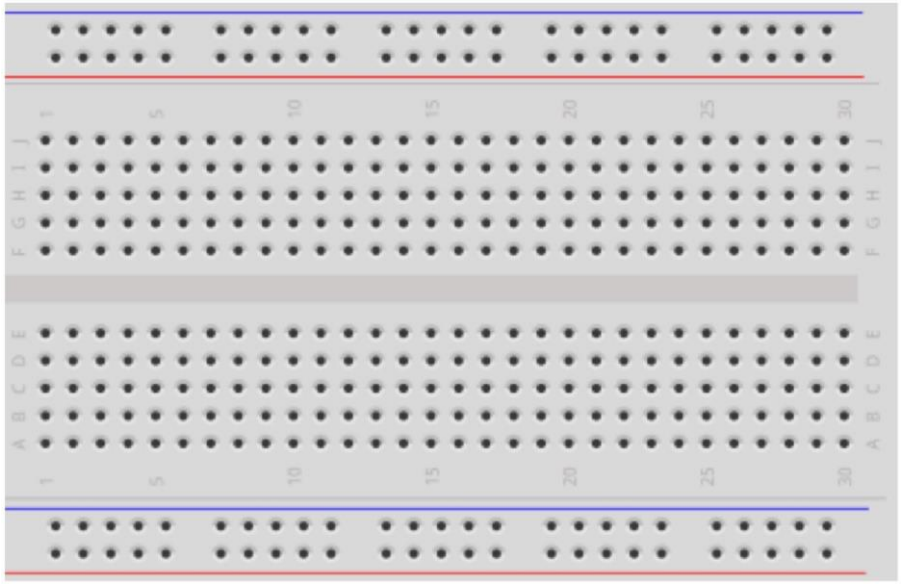
All the material

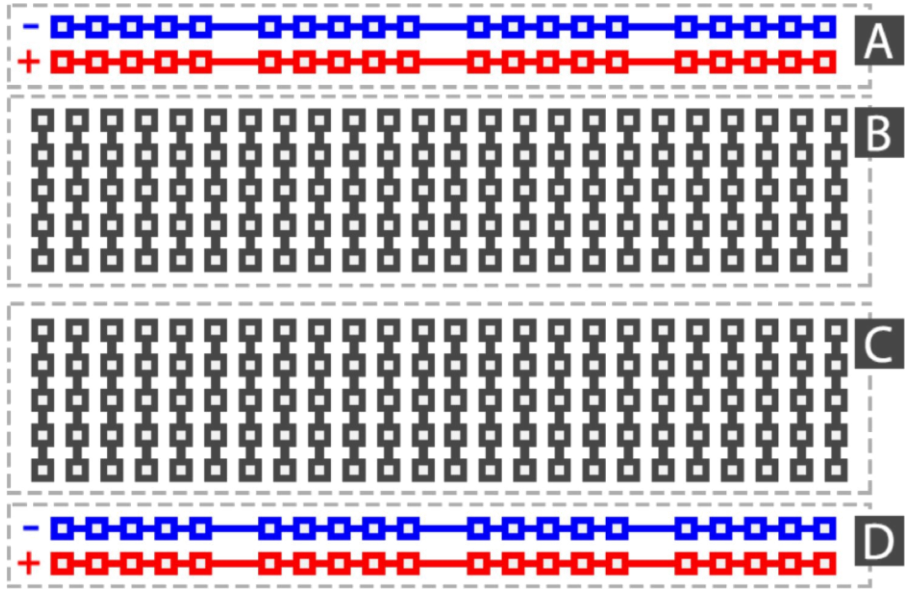
- The course will require the following **hardware**:
  - **Your PC** (Windows, Mac, Linux)
  - Breadboard
  - Arduino UNO/ESP32/ESP8266
  - Electrical components (wires, resistors, etc.)

- And the following **software**:
  - Arduino IDE/Platformio IDE (free)
  - Fritzing (free)

Breadboard (1/2)

A breadboard is used to wire electric components together.





You must be careful, sections **A** and **D** are connected horizontally, while sections **B** and **C** are connected vertically.

Breadboard (2/2)

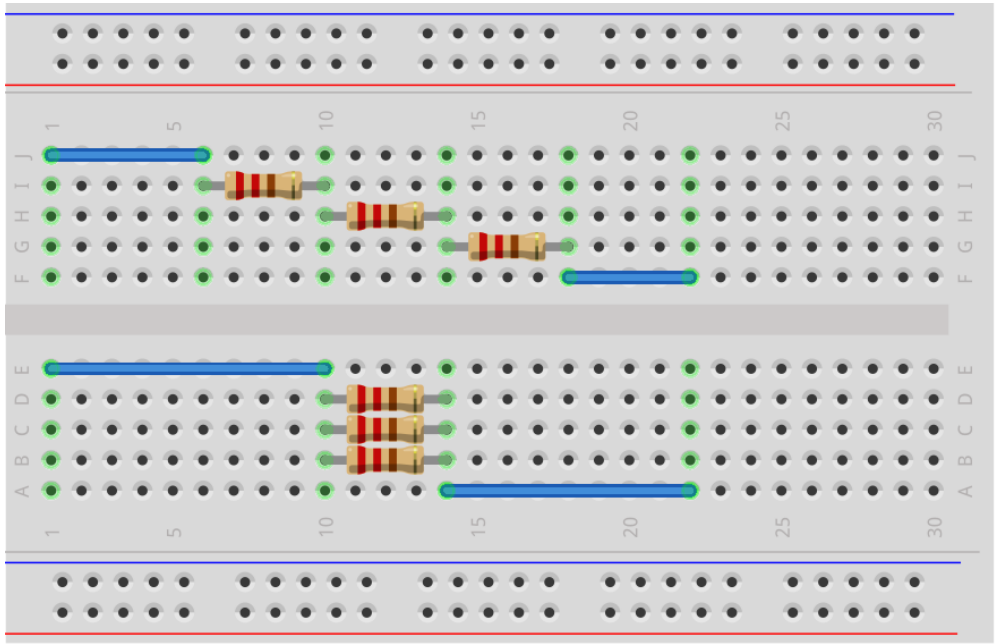
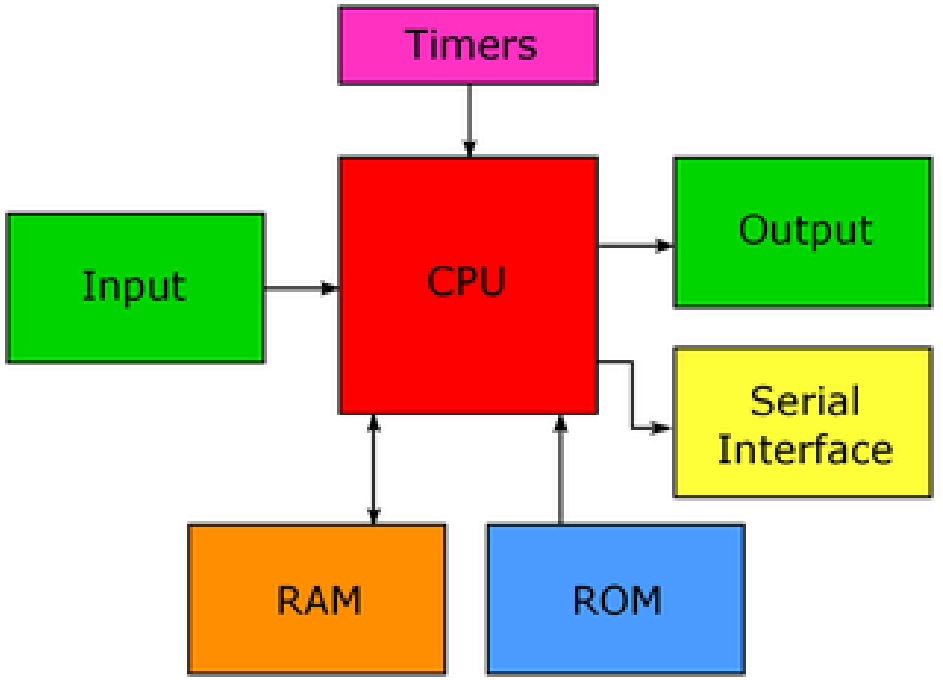


Figure: On the breadboard here you can see the series/parallel configurations of resistors.

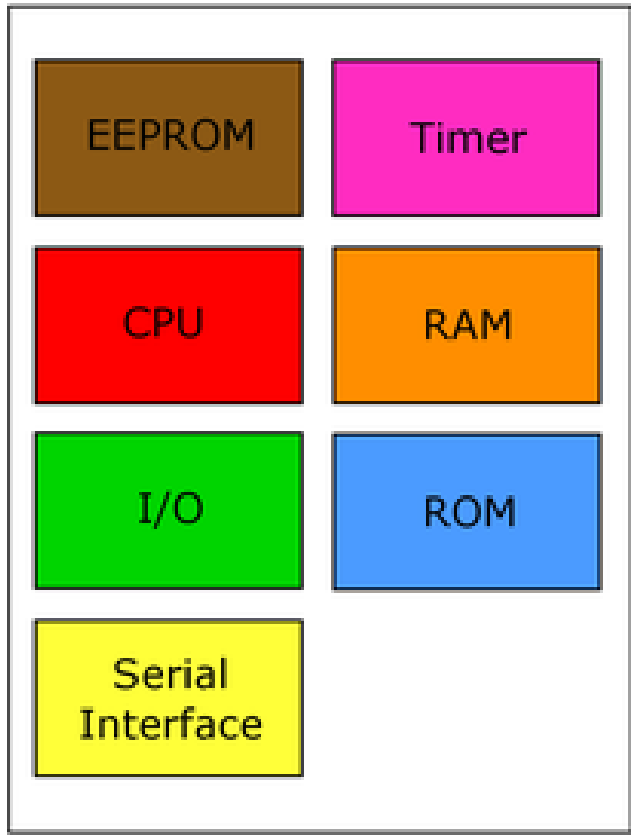
Is Arduino a Microprocessor or a Microcontroller? (1/3)

- A **microprocessor** is the brain of all computing systems (*e.g.*, PC, smarthphone, home assistant, measuring device, *etc.*);
- It's the unit **responsible** for all necessary calculations which allow a system to work and produce the **expected output**;
- It cannot work alone because it needs to **receive data** from other units, and this is why you will need other parts such as registers, memory units and Input/Output ports (at least).



Is Arduino a Microprocessor or a Microcontroller? (2/3)

- A **microcontroller** is an embedded system, embedding several units in one single chip:
  - Microprocessor;
  - Memory units (RAM, ROM, FLASH);
  - Input/Ouput Ports;
  - other peripherals (e.g, Analog-to-Digital Converter or Analog-Comparator or Timers, etc.);
- Allow developers to build a functioning system in a **short time**;
- It cannot work alone because it **needs power** and a proper interface to load and flash your program into it, as well as ways to display the processed data out of it.



Is Arduino a Microprocessor or a Microcontroller? (3/3)

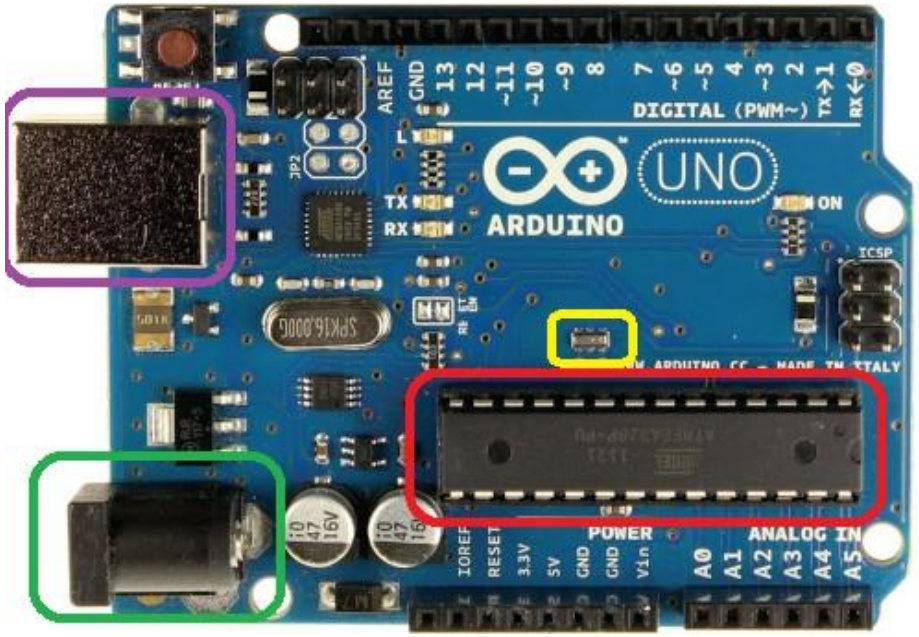
- A **development board** is a microcontroller (or microprocessor), and provide it with usb-port, HDMI-port, power-input port, display units such as Alphanumeric-LCD or other meaningful ways to display information (such as LEDs or Seven-Segment) and you'll have a development board;

- **Arduino** is one of the most famous (and very simple) development boards and there are many versions and types of Arduinos, each of which has different capabilities in terms of computing characteristics (type of microcontroller, size of memories, max. clock speed. . .) as well as interface functionalities (usb, hdmi, ethernet, number of Input/Output ports, LCD, LEDs);

- Development boards will allow you to start **testing** your **projects** and ideas in a very fast way, but it also has a downside which is the limitation of the available hardware inside it;

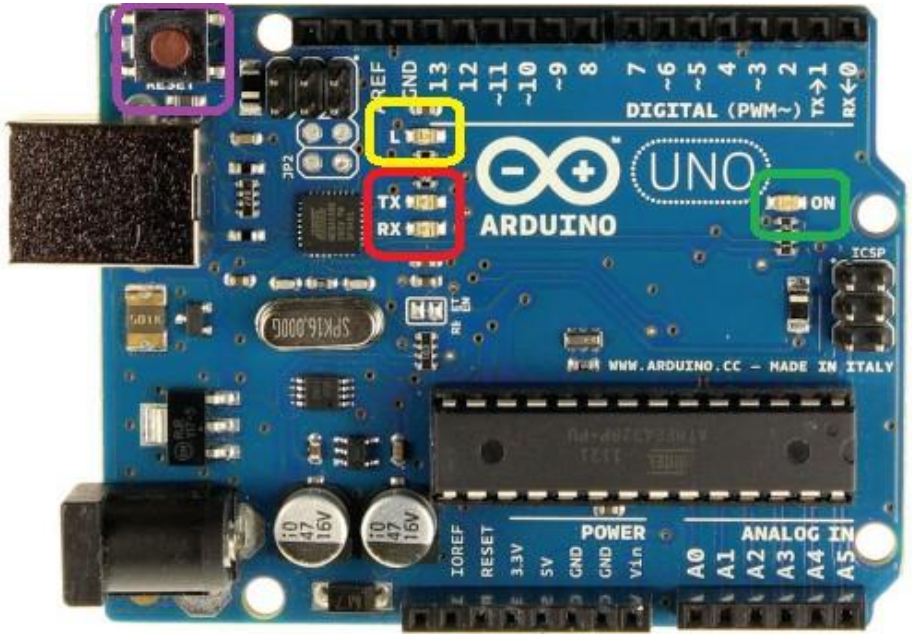
- Arduino isn't a microcontroller nor a microprocessor: It's a simple and easy-to-use development board that is relying on a microcontroller in it!

Arduino Structure (1/3)



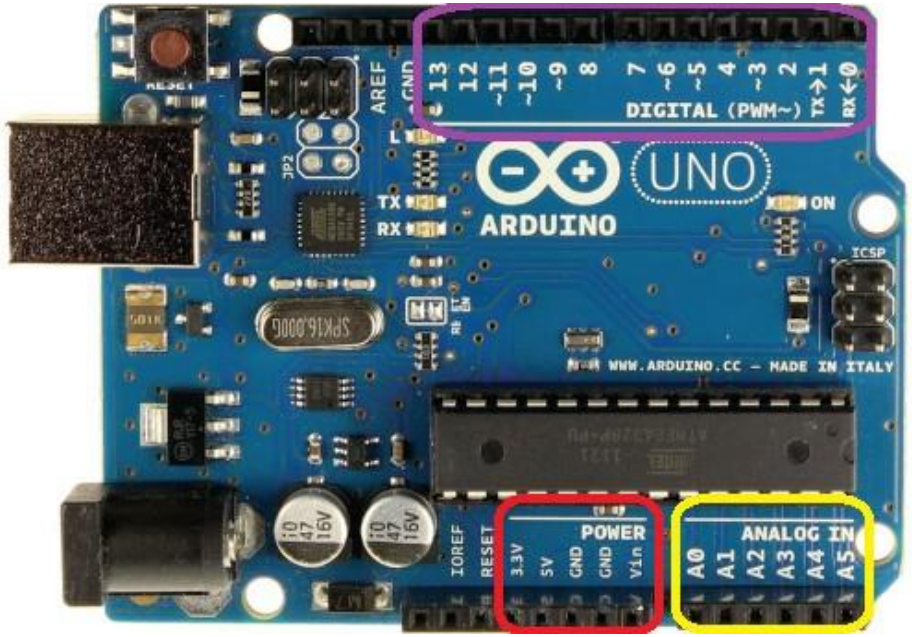
- **red:** the **microcontroller**, **ATmega328**;
- **yellow**: the clock, the **timer** with frequency **16MHz** that discretizes the time by subdividing it in intervals to perform the operations of the microcontroller;
- **purple**: the **USB** connector to perform the connection with a computer;
- **green**: **power** connector that accepts from 7 to 12 Volts of direct current.

Arduino Structure (2/3)



- **red**: led that blink when there is a **transmission (TX)** and a **receiving (RX)** of data;
- **yellow**: a led that you can turn on and off through the pin 13;
- **purple**: the **reset** button, which allows restarting the software run by the microcontroller;
- **green**: a led that warns us when the board is on.

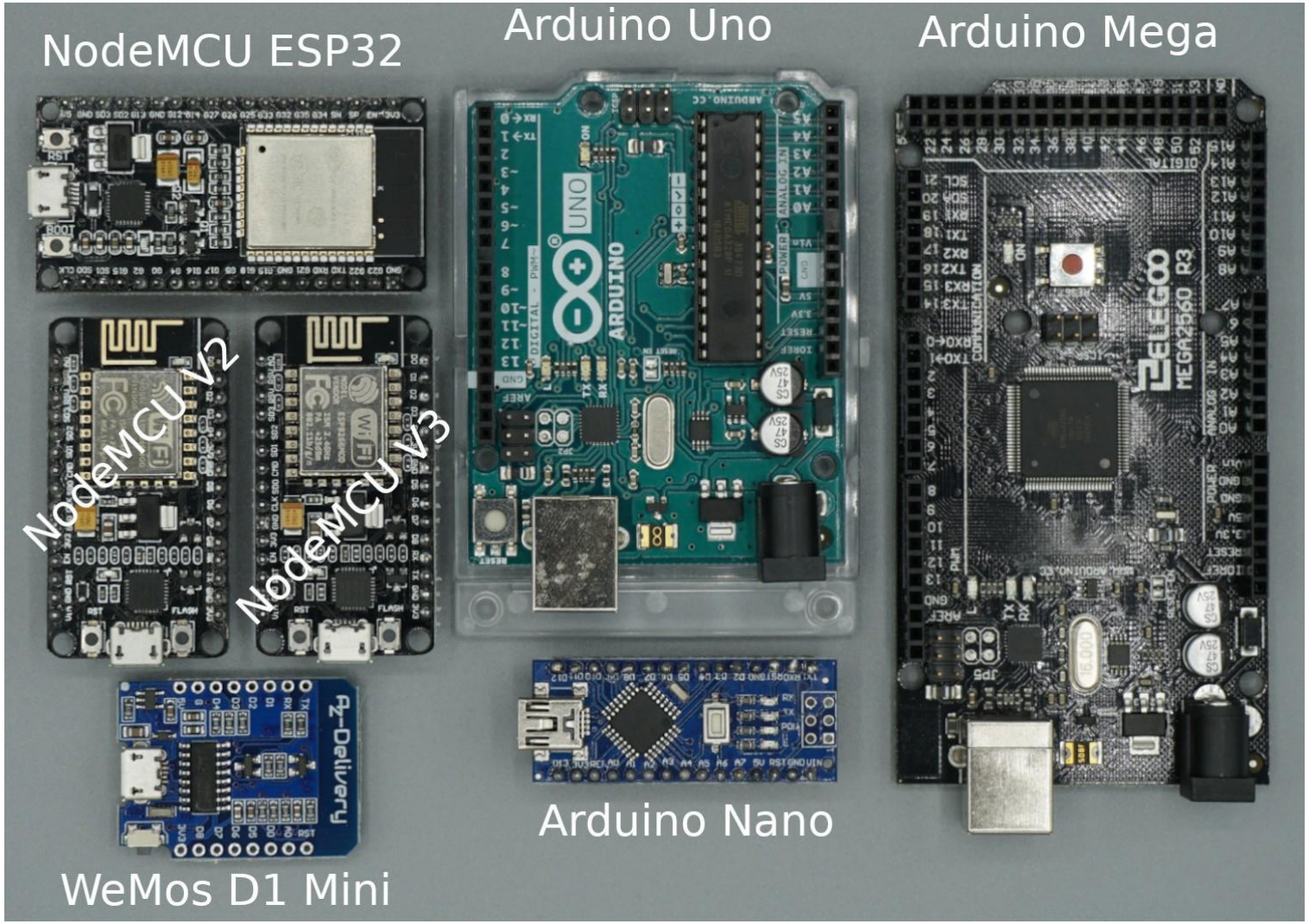
Arduino Structure (3/3)



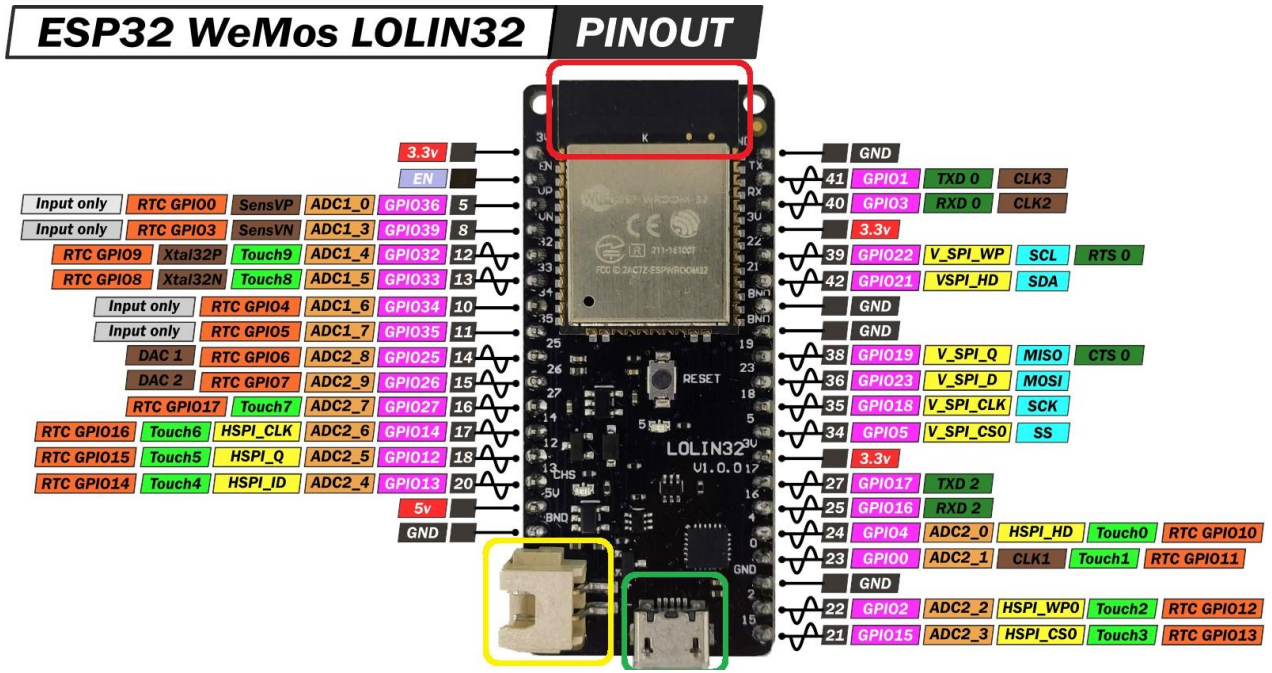
- **red**: the pins off **ground** (or **ground**) and the **power** pins (3.3V and 5V);
- **yellow**: analog pins;
- **purple**: digital pins.

Prototyping Boards

Prototyping Board Comparison



Another Board that's Catching On

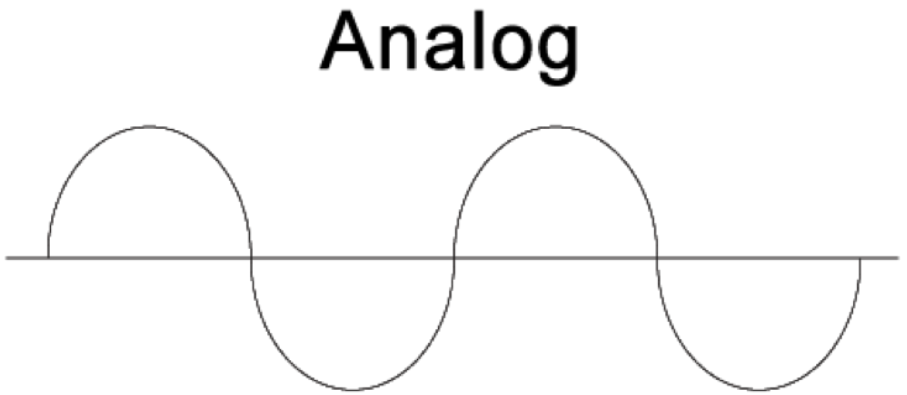


- **red**: an integrated PCB antenna;
- **yellow**: connection for battery, ESP32 works between 2.3V to 3.6V, 3.3V is better;
- **green**: micro-usb port to connect it to your computer and load the software;

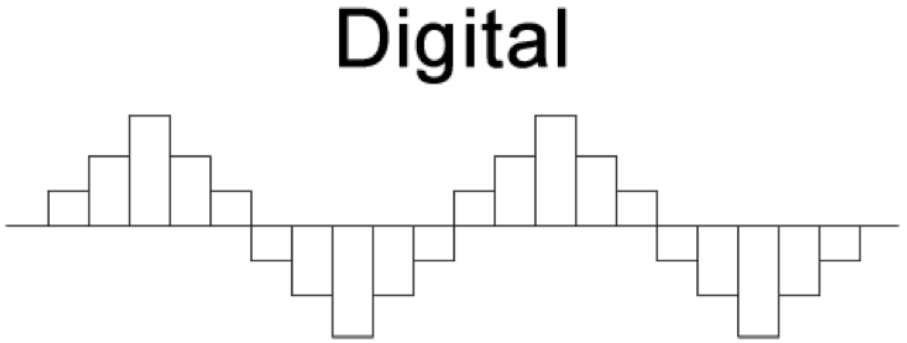
Analog and Digital Signals

Difference between an Analog and Digital Signal

- Analog Signal:
  - It is continuous in amplitude;
  - It is continuous in space and/or time;



- Digital Signal:
  - It is discrete in amplitude;
  - It is discrete in time and/or space;

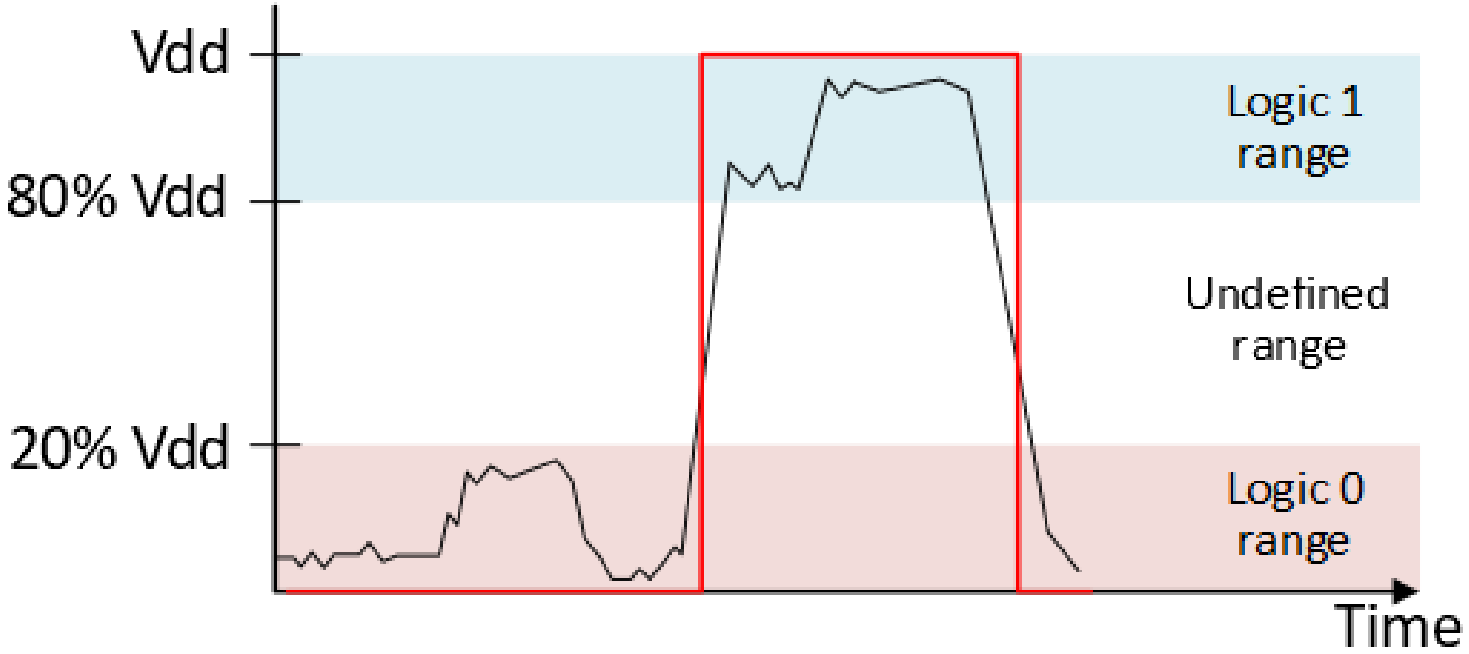


Analog Signal vs Digital (1/2)

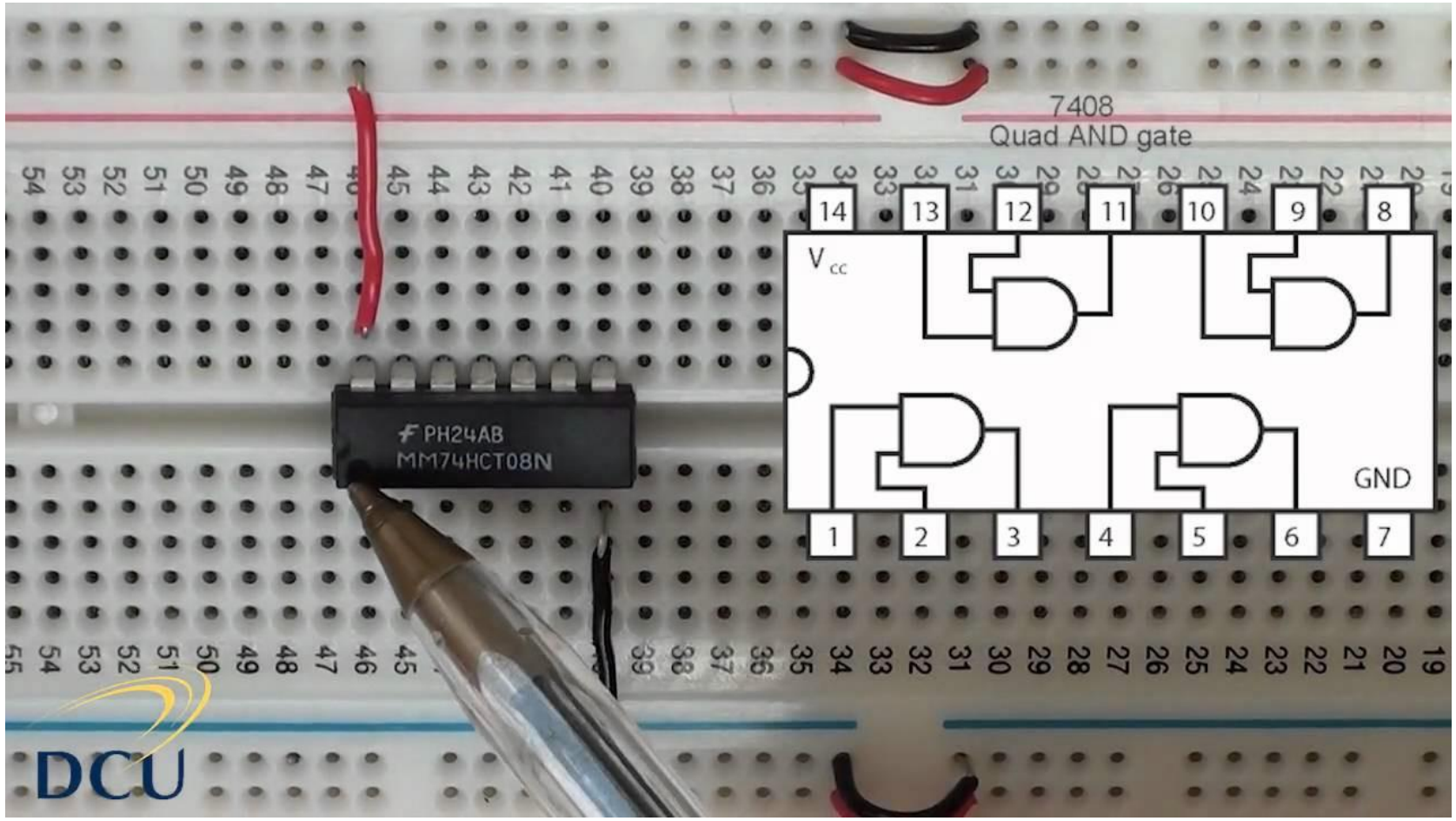
- In electronic circuits, by “textbfsignal” we mean a time-varying voltage or current that carries information. Signals are typically carried along **wires** that carry relatively low voltages (e.g., 3.3V) or low currents (perhaps a few milliamps).

Analog Signal vs Digital (2/2)

- Most signals use “textbfvoltage levels” to carry information. Information in digital circuits is encoded using two voltage levels: a “**logical voltage high**” or Vdd, represented by a ’1’; and a “**logical voltage low**”, or ground, represented by a ’0’.



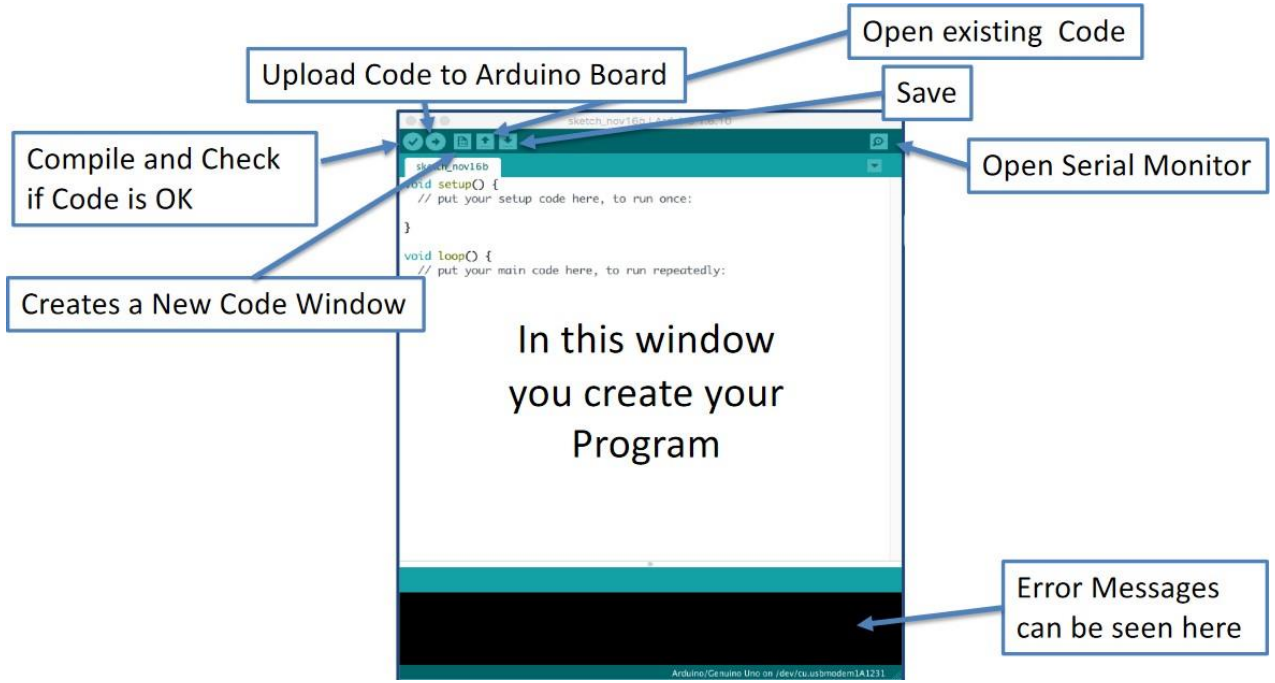
Digital Circuit on a Breadboard



Practical Examples

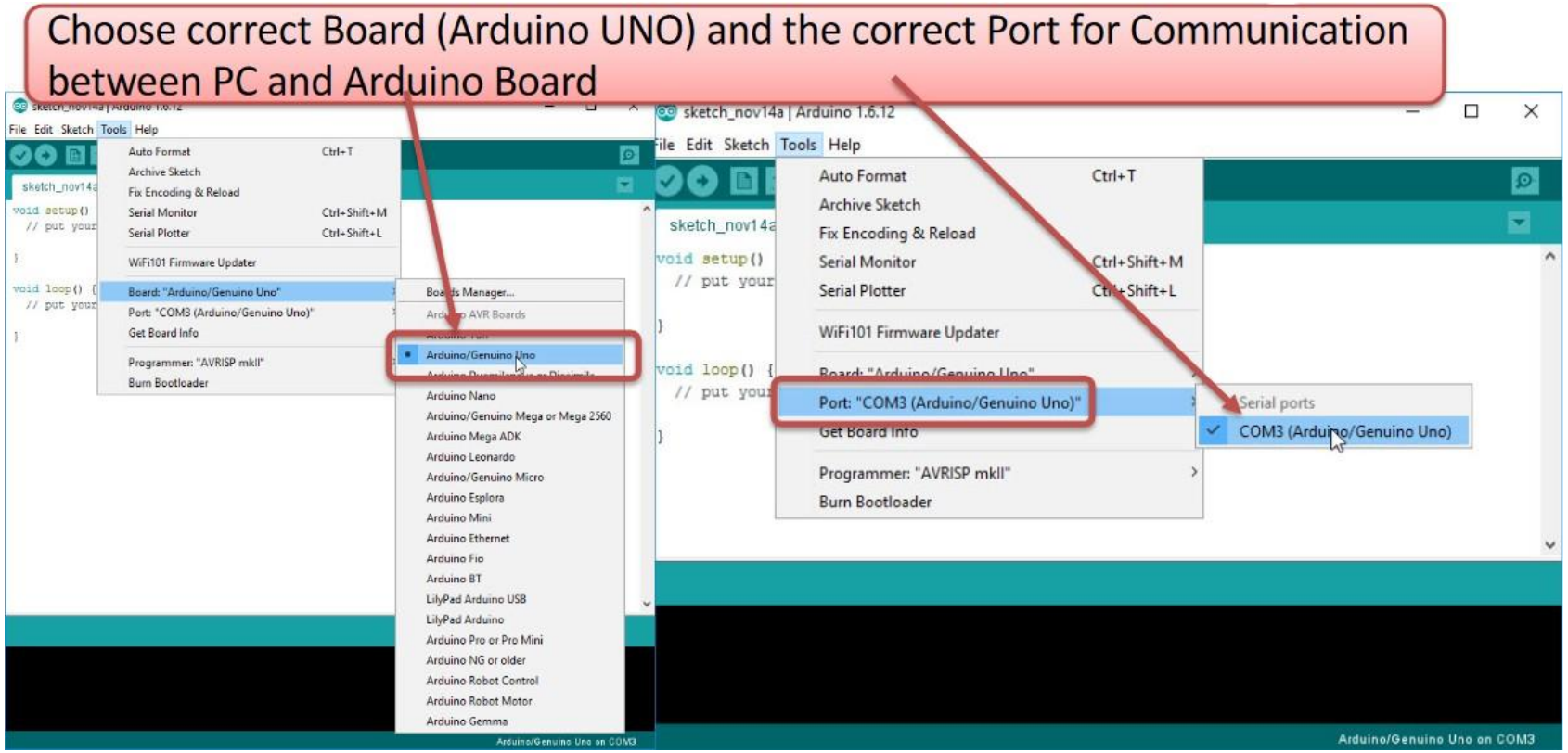
Arduino IDE (1/2)

- In order to use the Arduino board, download the Arduino IDE fro:
  - https://www.arduino.cc/en/Software
- To install a new library, you just need to click on:
  - Sketch –→ Include Library –→ Manage Libraries



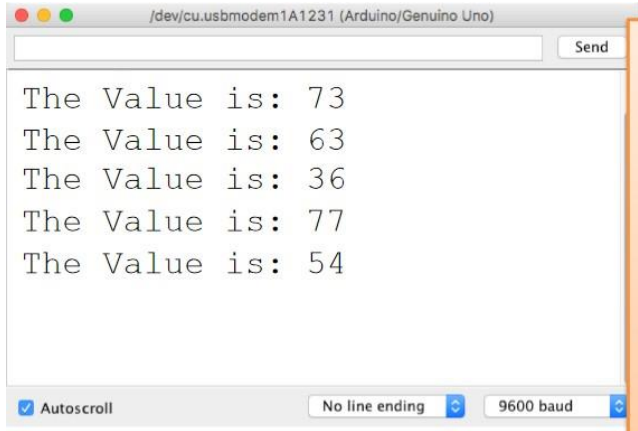
Arduino IDE (2/2)

- In order to execute the IDE, use the file named `arduino` inside the downloaded package.



Serial Monitor

- The serial monitor allows reading the debug messages written by the Arduino connected to the PC.
  - It can also be used to send commands to the Arduino



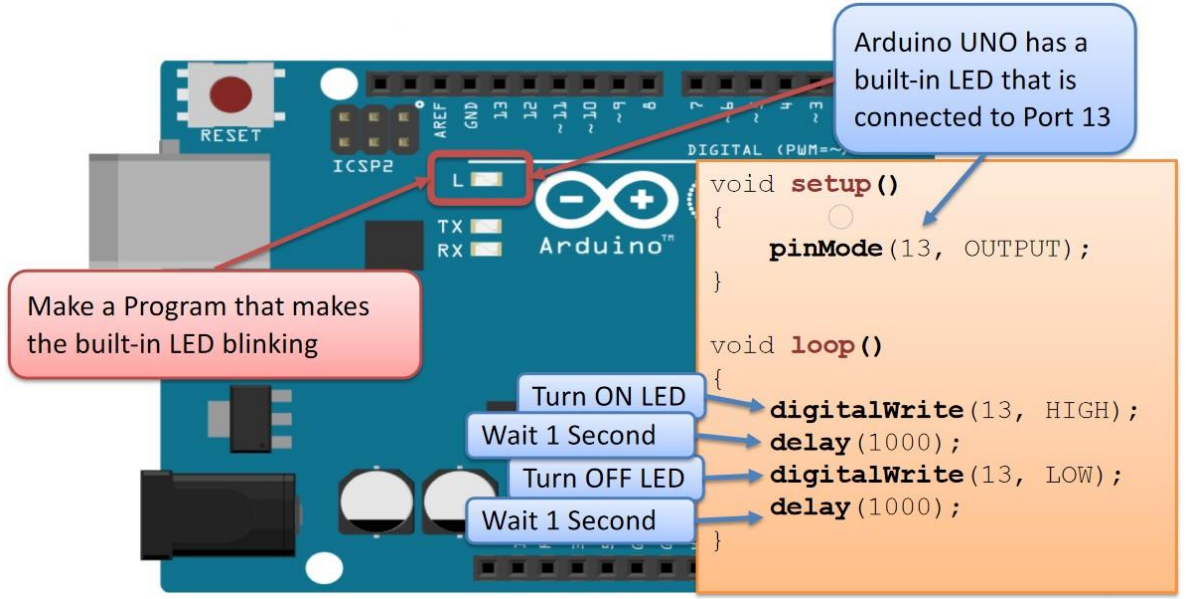

```
int myValue = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    myValue = random(100);
    Serial.print("The Value is: ");
    Serial.println(myValue);
    delay(1000);
}
```

LED Blinking

```
1 void setup () {
2    pinMode (13 , OUTPUT );
3 }
4
5 void loop () {
6    digitalWrite (13 , HIGH );
7    delay (1000 );
8    digitalWrite (13 , LOW );
9    delay (1000 );
9 }
```



Fritzing Modeling SW

Fritzing is an open-source hardware initiative that makes electronics accessible as a creative material for anyone.
It comprises a software tool, a community website and services in the spirit of Processing and Arduino, fostering a creative ecosystem that allows users to *document* their prototypes, *share* them with others, *teach* electronics in a classroom, and layout and *manufacture* professional PCBs
https://fritzing.org/

Open source code available on GitHub: https://github.com/fritzing/fritzing-app

Fritzing Simple Example

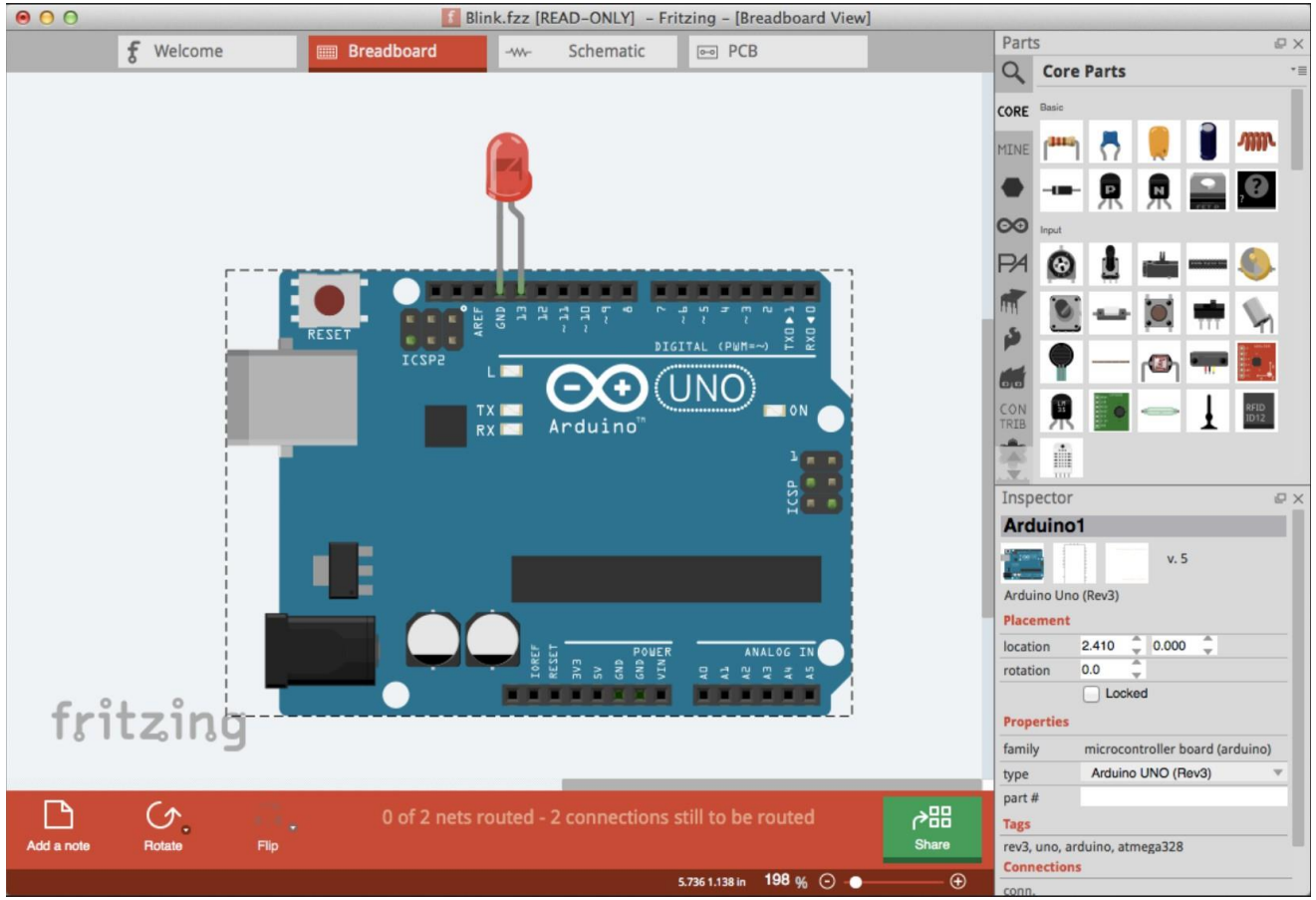
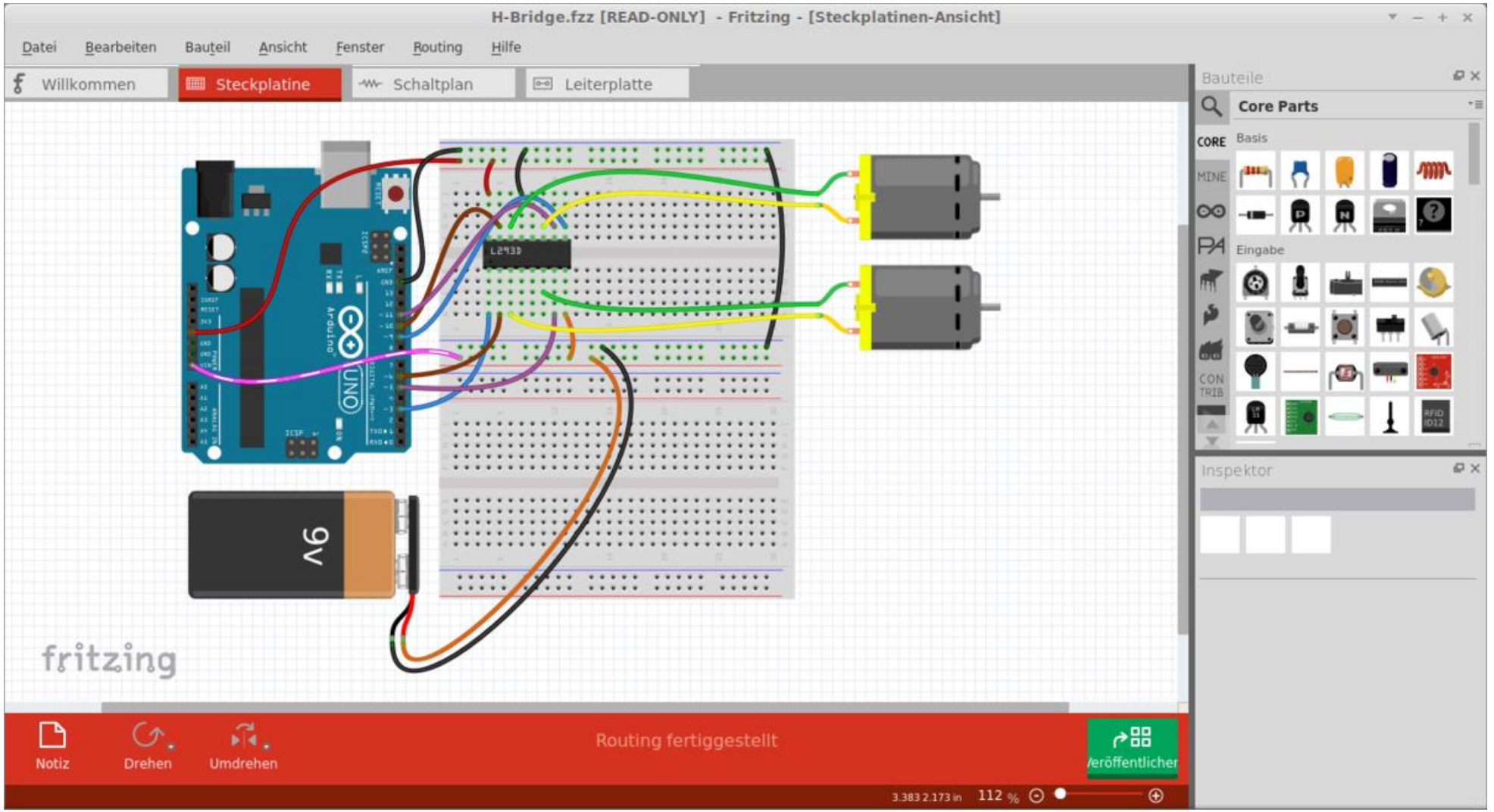


Figure: Easy framework to build virtual prototypes.

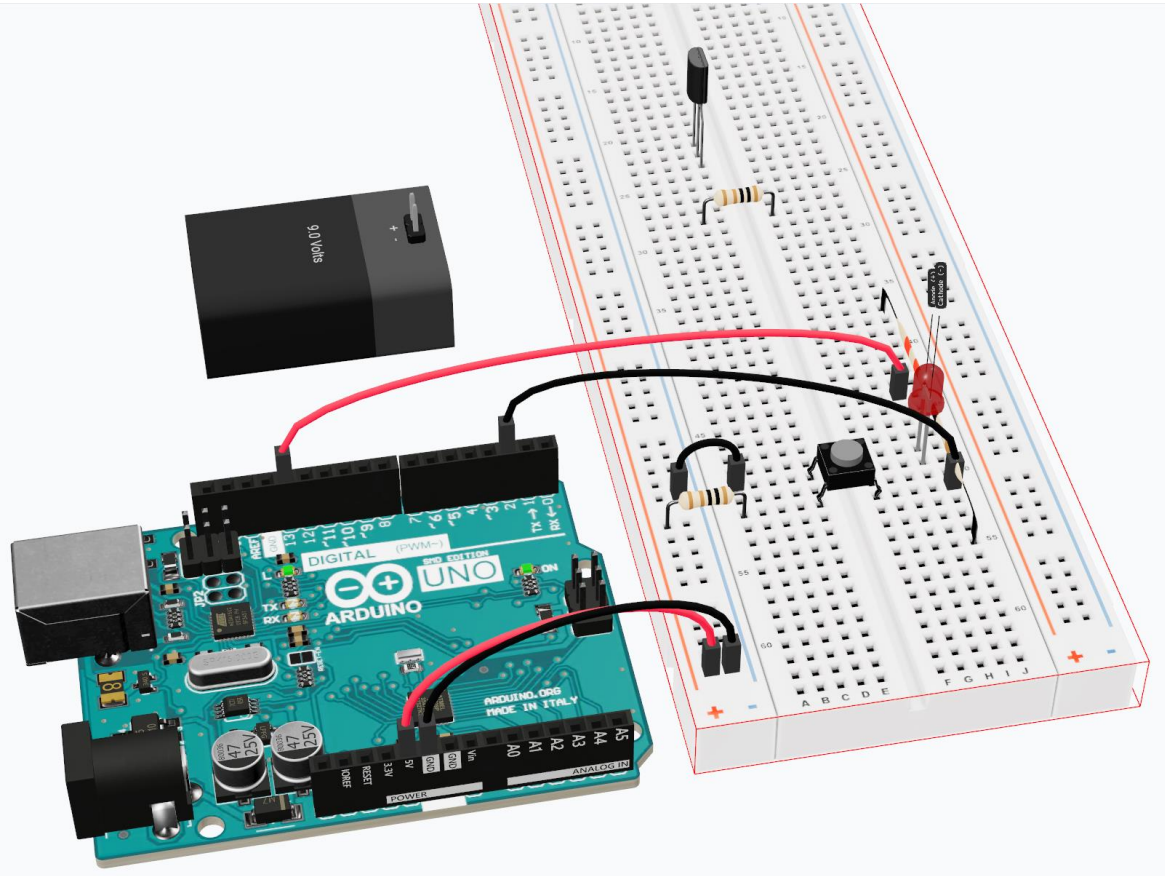
Fritzing Complex Example



Build, Program, and Run Hardware Projects in the Browser

Diode is a free 3D hardware simulator capable of simulating Arduino, integrated circuits, capacitors, transistors and much more

https://www.withdiode.com/



Practical Examples

Light Emitting Diode (LED)

Light Emitting Diode (LED) (1/2)

- A Light Emitting Diode (LED) comprises an **anode** and a **cathode**.
- The former is connected to a voltage source (i.e., PIN 13 of Arduino) while the latter (i.e., the negative pole) is connected to **Ground**.

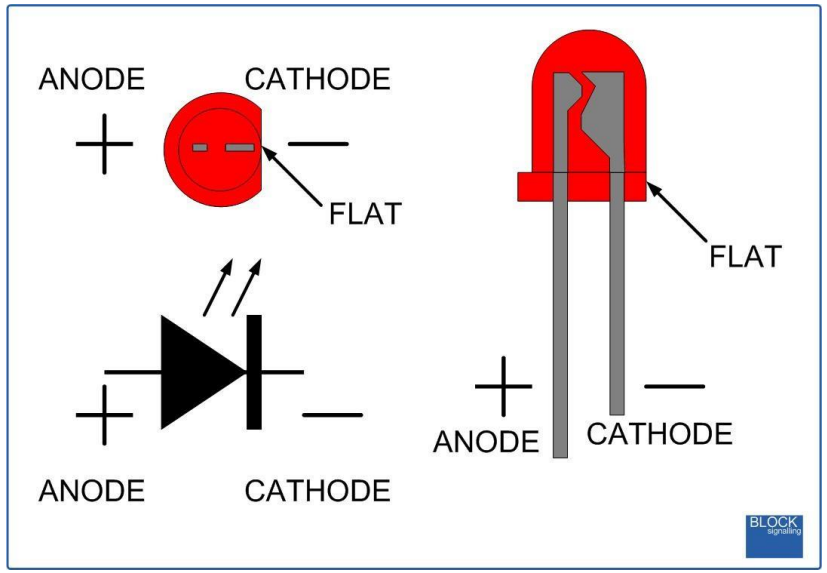


Figure: Led structure.

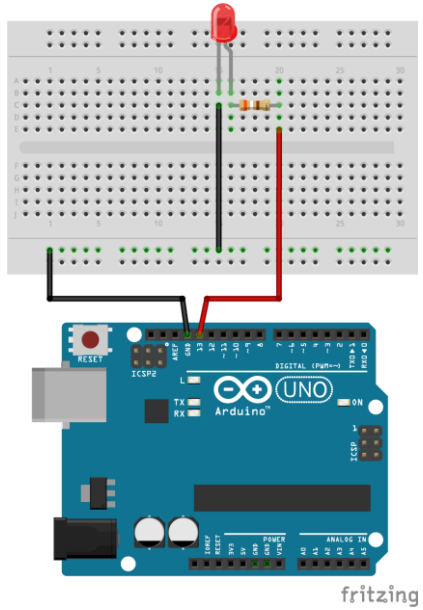


Figure: Led setup.

Light Emitting Diode (LED) (2/2)

- During the setup phase you have to set the **pin number** to which the LED is connected and its type (i.e., INPUT or OUTPUT).
- To change the state of the LED, use the digitalWrite command and pass the pin number and pin state.
- Using HIGH will turn on the LED while LOW will turn it off.

```
1 #define led0 13
2 int status = 0;
3 void setup () {
4     pinMode (led0 , OUTPUT );
5     digitalWrite (led0, status);
6 }
7 void loop(){
8     if (status == LOW) status = HIGH;
9     else                status = LOW;
10    digitalWrite (led0, status);
11    delay (200);
12 }
```

Practical Examples

Button

Button (1/2)

- The button must be powered and thus it must be connected to the ground (i.e., GND) and to a potential source (i.e., VCC of Arduino).
- Even though it has four pins, they are actually only two since the pins on the same side are connected in series. A cable must be used to send the signal from the button to the PIN 4 of the Arduino.

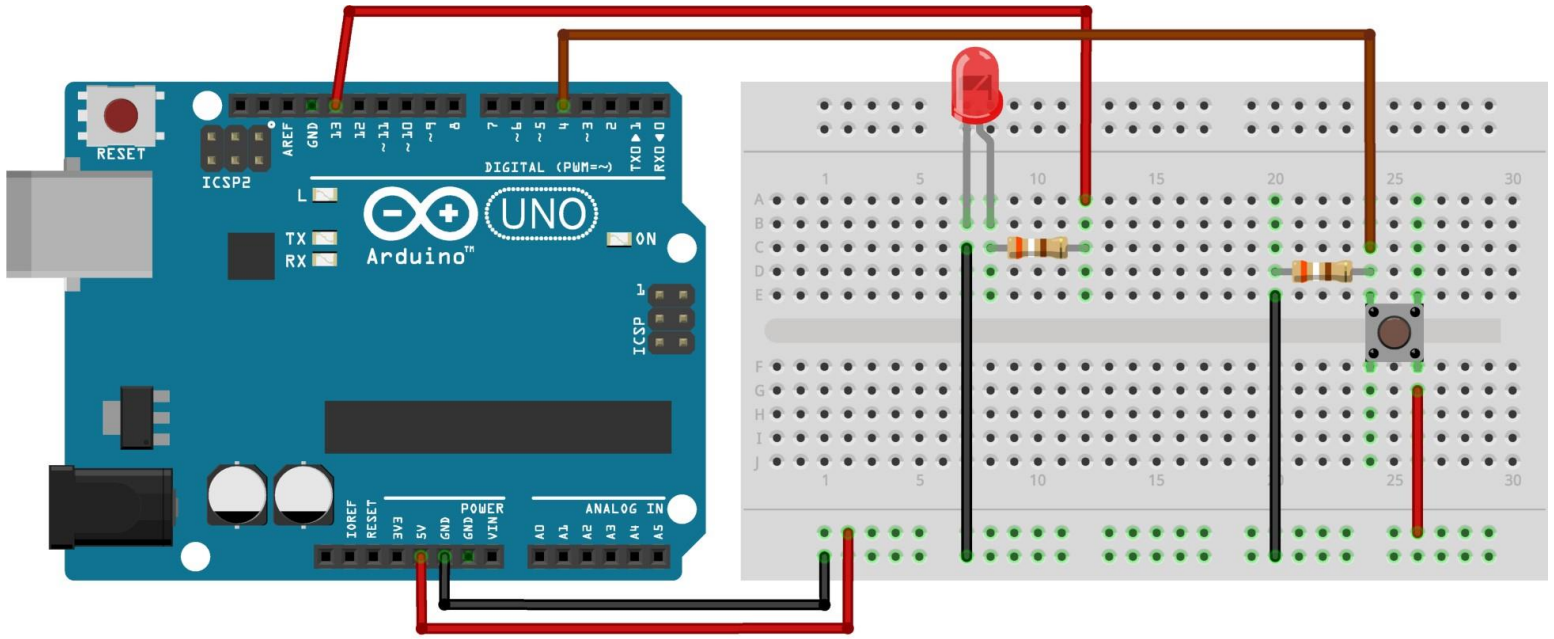


Figure: Button Setup.

Button (1/2)

- The button is an input device, during the setup phase you have to initialize the pin by setting it as INPUT pin.
- During the loop phase, the code has to verify if the button is pressed by means of the digitalRead function.
- This function indicates value 1 if the button is pressed and value 0 if the button is not pressed.

```
1 #define pinButton 4
2 #define pinLed 13
3 void setup () {
4     Serial.begin(9600);
5     pinMode (pinButton , INPUT );
6     pinMode (pinLed, OUTPUT );
7 }
8 void loop(){
9     if (digitalRead (pinButton) == 1){
10        digitalWrite (pinLed, HIGH);
11        delay(1000);
12        digitalWrite (pinLed, LOW);
13    }
14 }
```

Practical Examples

Buzzer

Buzzer (1/2)

- A buzzer must be connected to a power supply (or an Arduino digital pin) and ground.

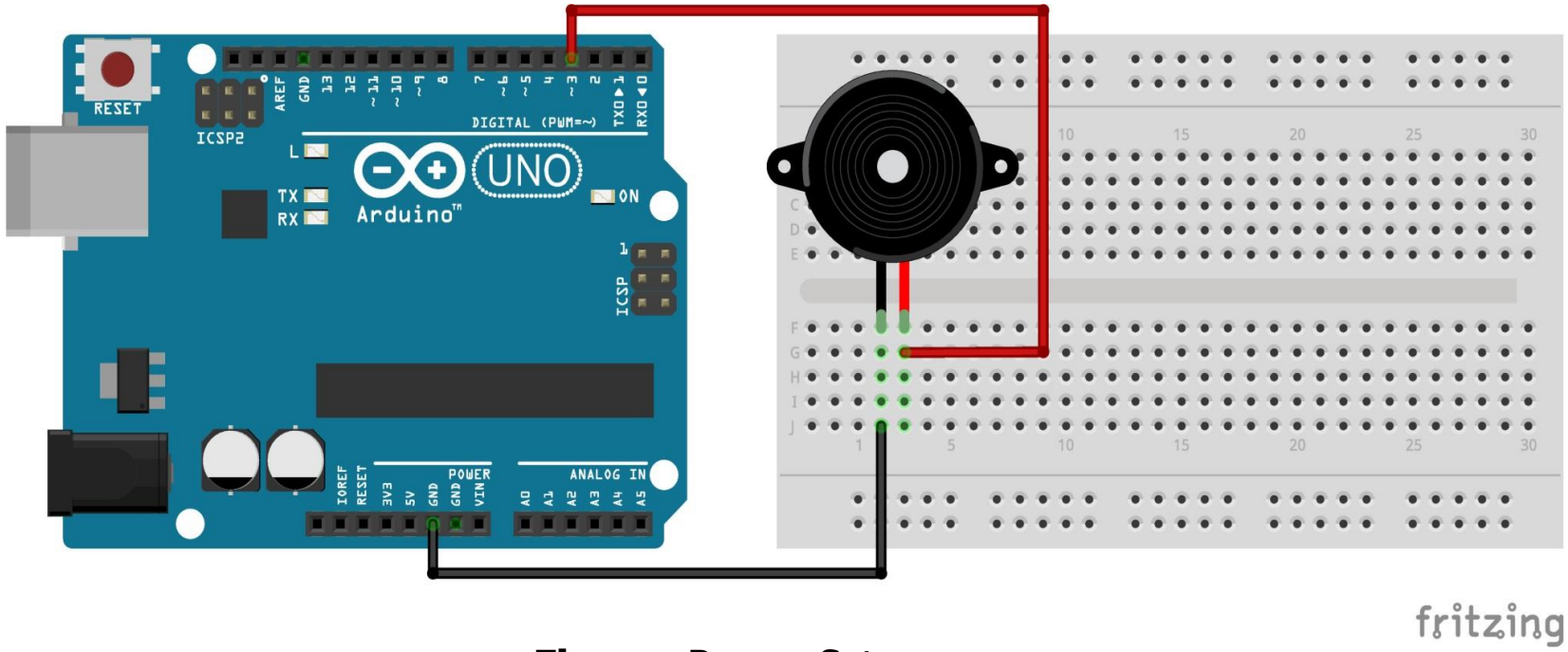


Figure: Buzzer Setup.

Buzzer (2/2)

- During the setup the buzzer must be initialized like a LED, that is with the pinMode(pinBuzzer, OUTPUT) command;
- Inside the loop function the buzzer is set through the command tone(pinBuzzer, frequency, milliseconds).

```
1 #define pinBuzzer 3
2 void setup () {
3     pinMode (pinBuzzer , OUTPUT );
4 }
5 void loop(){
6     tone(PinBuzzer, 1000, 200);
7     delay(500);
8     noTone(PinBuzzer);
9     delay(1000);
10    tone(PinBuzzer, 750, 300);
11    delay(500);
12    noTone(PinBuzzer);
13    delay(1000);
14    tone(PinBuzzer, 500, 400);
15    delay(500);
16    noTone(PinBuzzer);
17    delay(1000);
18 }
```

Practical Examples

Temperature sensor (DS18B20)

Temperature sensor (DS18B20) (1/4)

- The DS18B20 sensor can be powered with a voltage between 3.0V and 5.5V, thus you can simply connect VDD to the 5V Arduino pin. However, the DS18B20 can also extract its power from the data line, which means it only needs two wires for the connection. *This makes it ideal for use as an external sensor*. The GND pin must be connected to VCC and to the ground pin of the Arduino. While the data line pin must be connected to a digital pin of the Arduino (in this case pin 2). To power the component, the data line pin must be connected to the VDD pin.

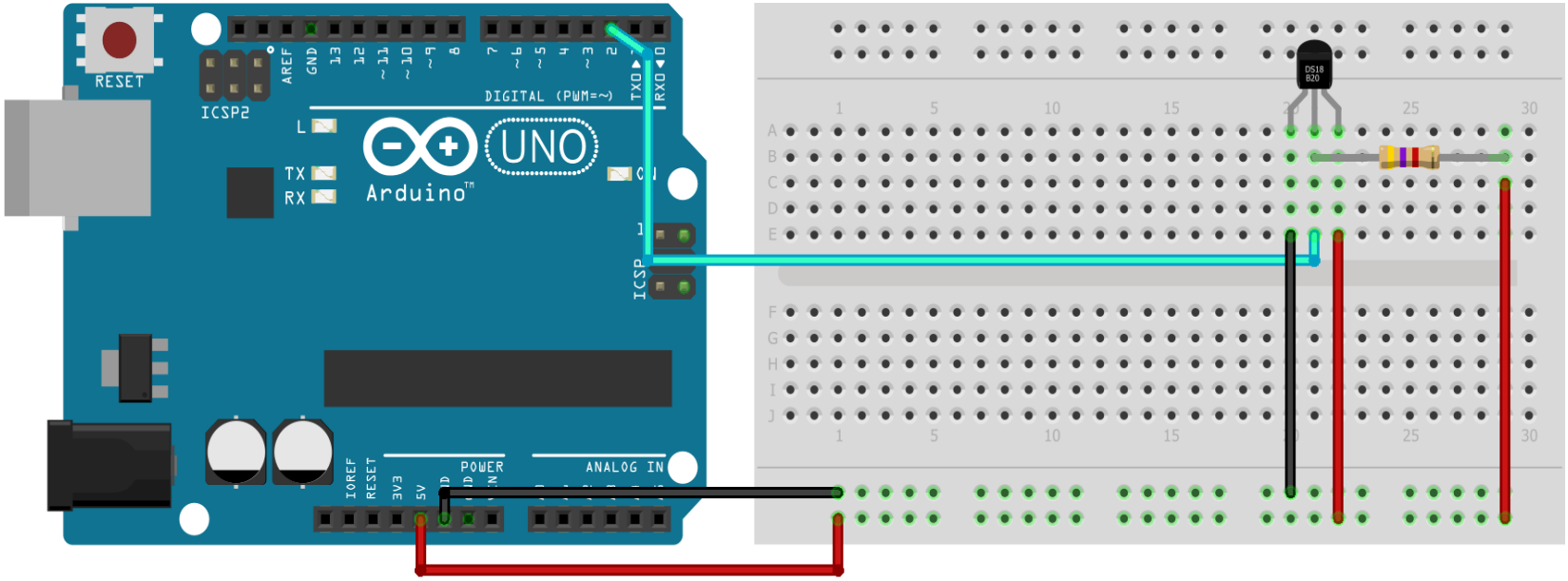


Figure: DALLAS DS18B20 sensor setup.

Temperature sensor (DS18B20) (2/4)

- The following libraries are required:
  - DallasTemperature
  - OneWire
- The OneWire library is a specific library for certain sensor classes, to which the Dallas Temperature sensor belongs.

Temperature sensor (DS18B20) (3/4)

- The code required to acquire the temperature is the following:

```
1 #include <DallasTemperature.h>
2 #include <OneWire.h>
3 #define ONE_WIRE_BUS 2
4 OneWire oneWire (ONE_WIRE_BUS );
5 DallasTemperature sensors (&oneWire);
6 void  setup () {
7      Serial.begin (9600);
8      sensors.begin();
9 }
10void loop(){
11     sensors.requestTemperatures ();
12     Serial.println (sensors.getTempCByIndex (0));
13   }
```

Temperature sensor (DS18B20) (4/4)

- The line OneWire oneWire(ONE WIRE BUS) sets the oneWire communication mechanism, while DallasTemperature sensor(&oneWire) instantiate the sensing library. Finally, by using the Sensors.begin(), the sensor begins to acquire the temperature.
- Inside the loop, two basic commands are used:
  - Sensors.requestTemperatures()
    - Sends a temperature request to the sensor.
  - Sensors.getTempCByIndex(0)
    - Acquires the temperature in Celsius.

Practical Examples

Current sensor (INA219)

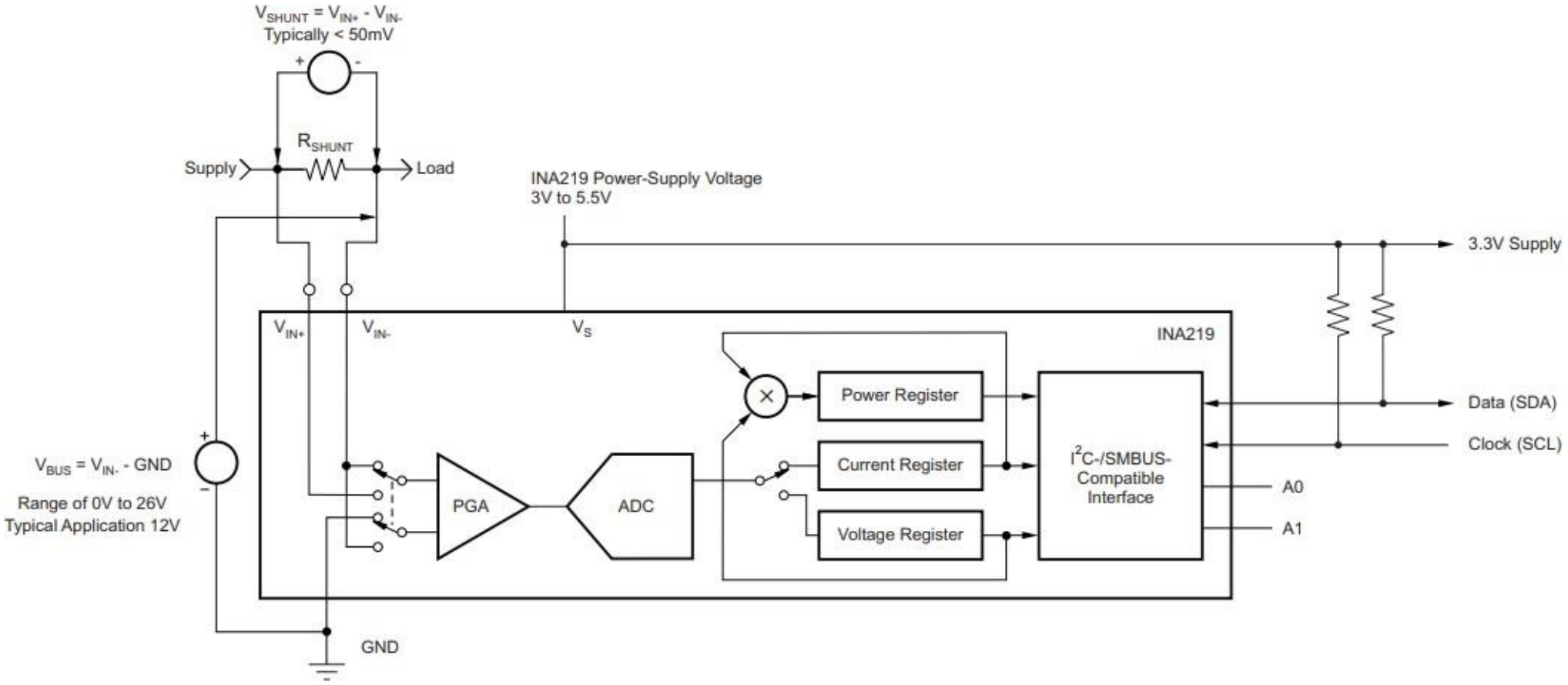
Current sensor (INA219) (1/6)

- The following table shows the technical datasheet of the INA219 voltage and current sensor.

Measurement Voltage	0V to 26V
Max Current	3.2A
Max Power	83W
Operation Voltage	3V to 5.5V
Communication Protocol	I2C

- The maximum voltage that can be measured with the INA219 sensor is 26V and the maximum current is 3.2A. Based on the maximum current of 3.2A and a maximum bus voltage of 26V, the INA219 can measure up to 3.2A * 26V = 83W of power.
- Because the operation voltage of the INA219 is between 3V and 5.5V, we can use different Arduino boards with an operation voltage of 5V. We can also use the wireless chip ESP8266 that we are going to see in the next slides with an lower operation voltage of 3.3V.

Current sensor (INA219) (2/6)



Current sensor (INA219) (3/6)

- The INA219 sensor has a **2-pin screw-terminal**, connected to the high side of the measuring in series, able to measure voltages up to 26V. The screw-term is connected to a 0.1Ω 1% sense shunt resistor in parallel.

- The INA219 measures two different voltages at the high side:
  - **V shunt**: is the voltage drop across the shunt resistor;
  - **V bus**: is the voltage from the negative pole with respect to ground.

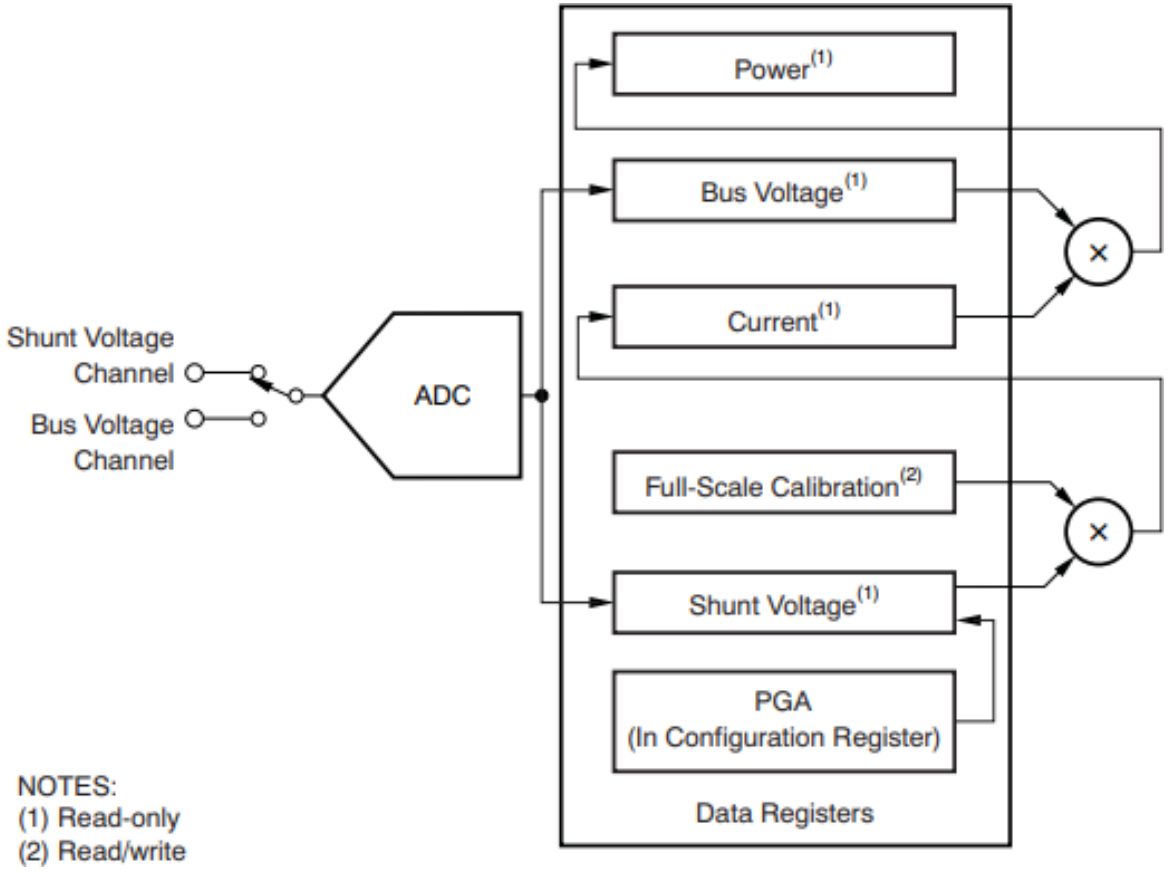
- Both voltages are then forwarded to the Programmable Gain Amplifier (PGA) to increase the sensitivity of the measurement.

- The INA219 increases the full-scale range up to 2, 4 or 8 times (320mV). Also the bus voltage measurements has two ranges: 16V or 32V.

Current sensor (INA219) (4/6)

- After the sensitivity of the voltage measurement is increased, the current and also the power is calculated.

- The following picture shows how the calculation is done based on the shunt and bus voltage.



Current sensor (INA219) (5/6)

- The current flow on the **high side** of the measurement is calculated by **multiplying** the shunt voltage with the calibrated resistance of the shunt resistor.
- Because the shunt resistor is 0.1Ω and the maximum shunt voltage at the scale of 8 is 320mV, the maximum current that can be measured is 320mV / 0.1Ω = 3.2A.
- Then the power is calculated with this current multiplied with the bus voltage. Therefore, the INA219 is able to provide four different measurements:
  - **Shunt voltage**: voltage drop across the shunt resistor;
  - **Bus voltage**: total voltage seen by the circuit under test;
  - **Current**: derived via Ohms Law from the measured shunt voltage;
  - **Power**: current multiplied by the bus voltage.
- Each of the measurements and calculations are stored in a register that is connected to the I2C interface to forward the values to the Arduino or ESP microcontroller.

Current sensor (INA219) (6/6)