Workshops Elisava 2011

Introduction to programming and electronics (Scratch & Arduino)

What is programming?

Make an algorithm to do something in a specific language programming.

- **Algorithm:** a procedure or formula for solving a problem.
- **Programming language:** artificial language designed to communicate instructions to a machine.

It also involves the process of designing, writing, testing, debugging, and maintaining the source code of a **computer program**.

What is Arduino? (I)

It's an open source electronics prototyping platform:

- **Open source:** resources that can be used, redistributed or rewritten free of charge, often software or hardware.
- **Electronics:** technology which makes use of the controlled motion of electrons through different media.
- **Prototyping**: an original form that can serve as a basis or standard for other things.
- **Platform:** hardware architecture with software framework on which other software can run.

What is Arduino? (II)

The Arduino board is like a small computer that can be programmed as many times as needed.

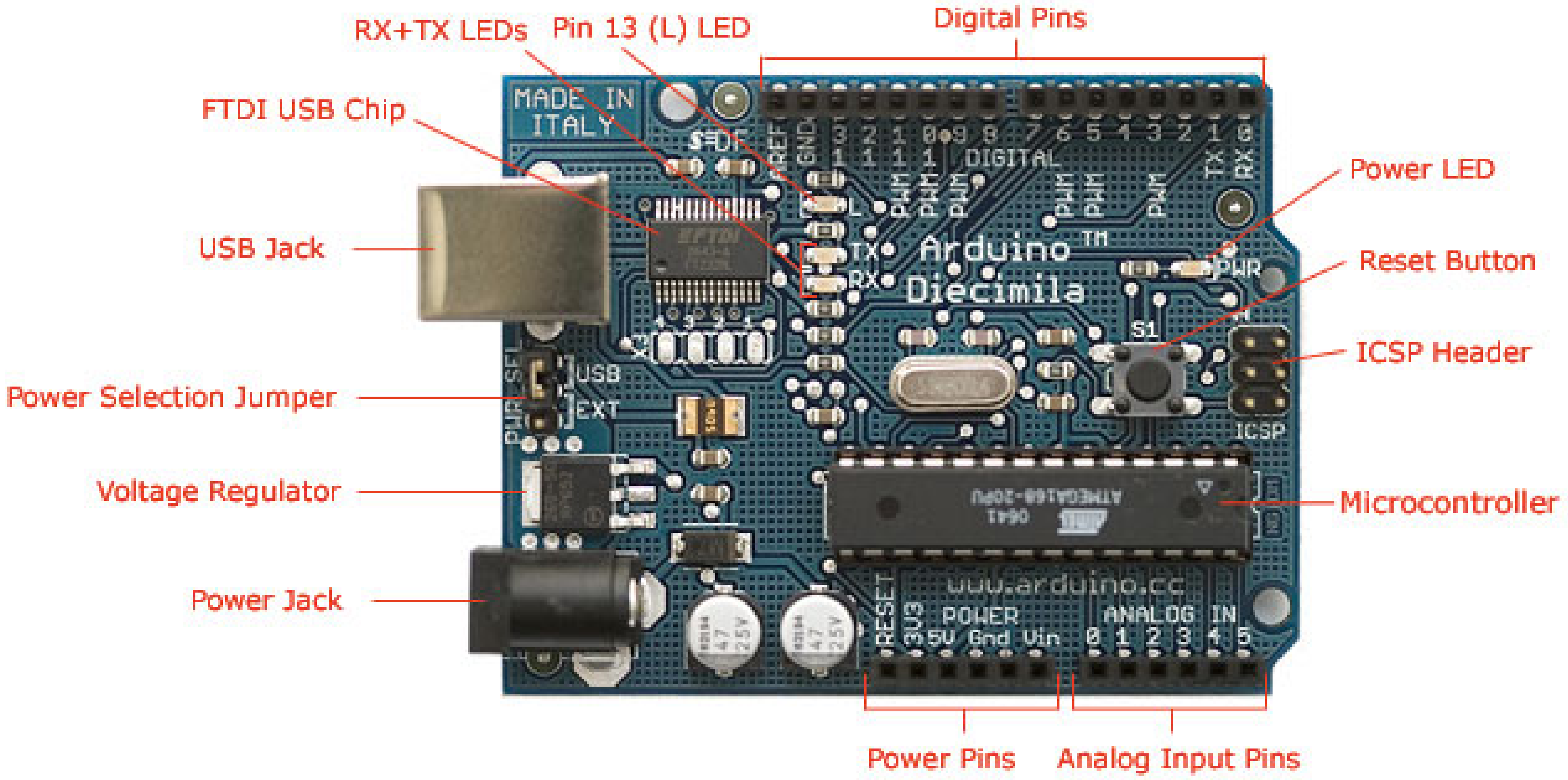
As a computer, it provides I/O interaction, through digital (input and output) and analog input pins.

- **Digital:** discrete and finit, described in two states: 1/0, ON/OFF.
- **Analog:** continuous, can have infinite number of values.

The sketch made with the Arduino IDE is loaded to the board and stored in the microcontroller.

More info at: http://arduino.cc/

Parts of Arduino board



What is Scratch?

It's an open source and educational software focused mainly for children, designed by the Lifelong Kindergarten group at MIT, and implemented in Smalltalk (Squeak).

The programming instructions are pieces that have to be stick each other in an order to form blocks and make a coherent program, just like a puzzle.

More info and downloads at: http://scratch.mit.edu/

What is S4A?

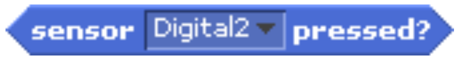
Scratch for Arduino (S4A) is a modified version of Scratch ready for communication with Arduino boards.

An Arduino sketch (S4AFirmware) has to be loaded to the board to work properly with S4A.

Both the installer and firmware can be downloaded from our website: http://seaside.citilab.eu

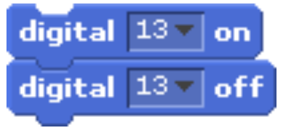
Pin mapping in S4A:

- **Digital read:** digital pins 2 and 3.



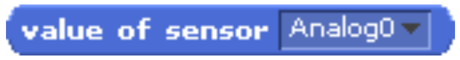
```
sensor Digital2 pressed?
```

- **Digital write:** digital pins 10, 11 and 13.



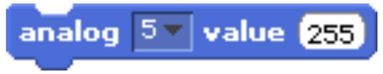
```
digital 13 on
digital 13 off
```

- **Analog read:** analog pins 0-5.



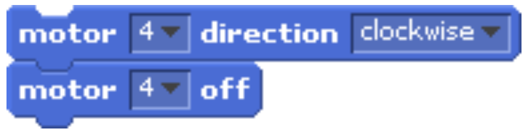
```
value of sensor Analog0
```

- **Analog write:** digital pins 5, 6 and 9.



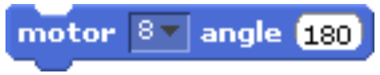
```
analog 5 value 255
```

- **Servo control:**
    - digital pins 4, 7 (continuous rotation).



```
motor 4 direction clockwise
motor 4 off
```

    - 8 and 12 (standard).



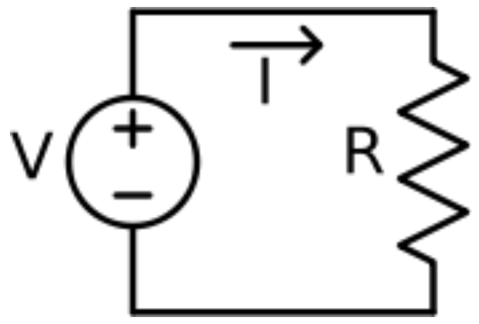
```
motor 8 angle 180
```

Basic electronics: Ohm's Law

Electricity if the flow of energy (electrons) through a conductive material.

- **Voltage (V):** is the measure of electrical potential, measured in *Volts (V)*.
- **Current (I):** is the amount of flow through a conductive material, measured in *Amperes* or *Amps (A)*.
- **Resistance (R):** is the material's opposition to the flow of electric current, measured in *Ohms* (Ω).

$$I = \frac{V}{R}$$



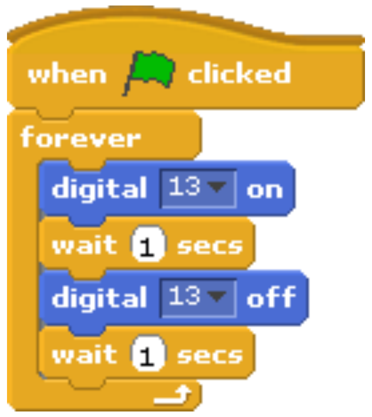
Sample program: Blink

Comparison of a simple program that blinks a LED connected to pin 13 on Arduino and in S4A. We can appreciate the differences between the form and syntax of programming languages in both cases:

Arduino

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

S4A

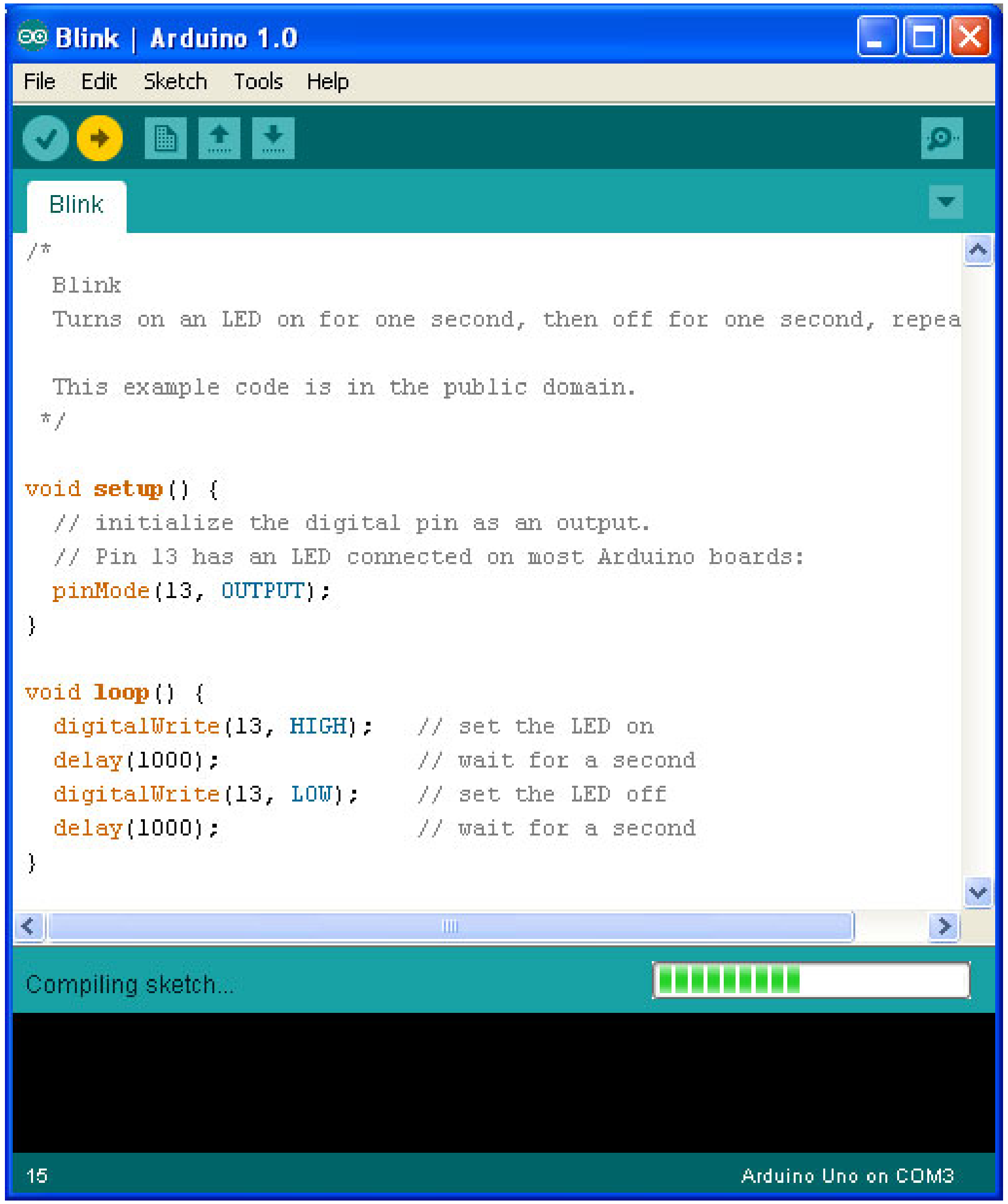


```
when [flag] clicked
forever
  digital 13 on
  wait 1 secs
  digital 13 off
  wait 1 secs
```

Uploading a sketch (I)

- Download the Arduino IDE at: http://arduino.cc/en/Main/Software and extract it.
- Open an example: *File* > *Examples* > *Basics* > *Blink*.
- Select the board version at: *Tools* > *Board* and serial port where the board is connected on: *Tools* > *Serial port.*
- Click on upload button.
- Done!

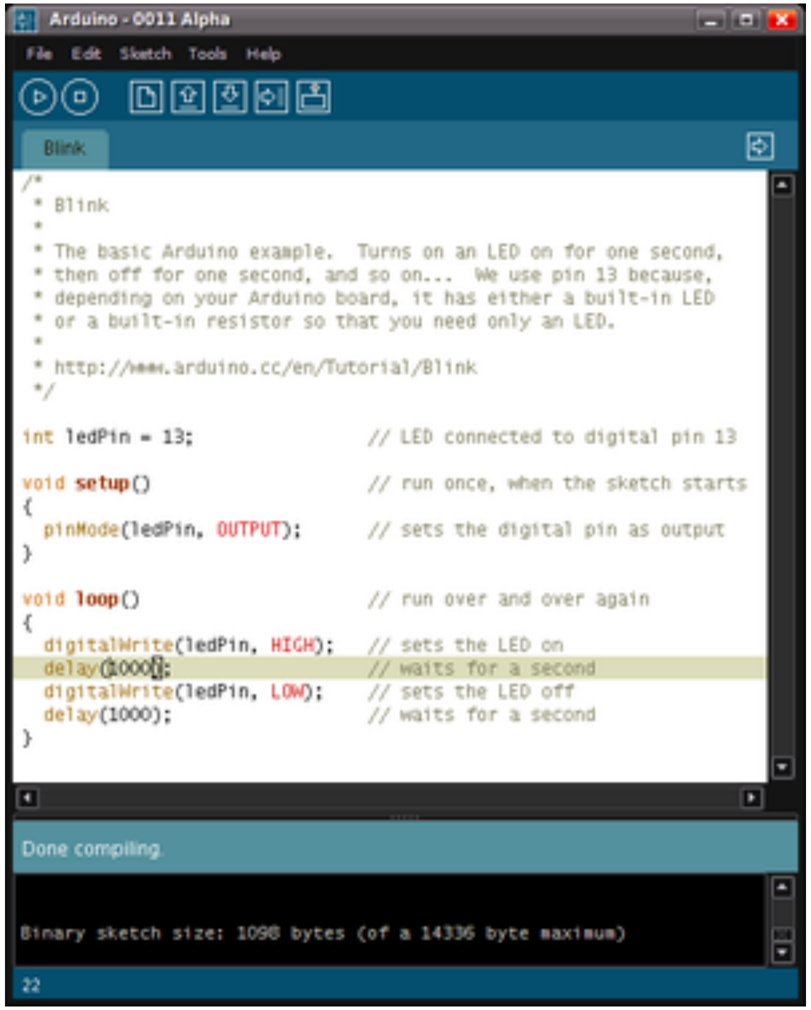
Uploading a sketch (II)



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repea

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

```
Compiling sketch...
```

```
15                                    Arduino Uno on COM3
```

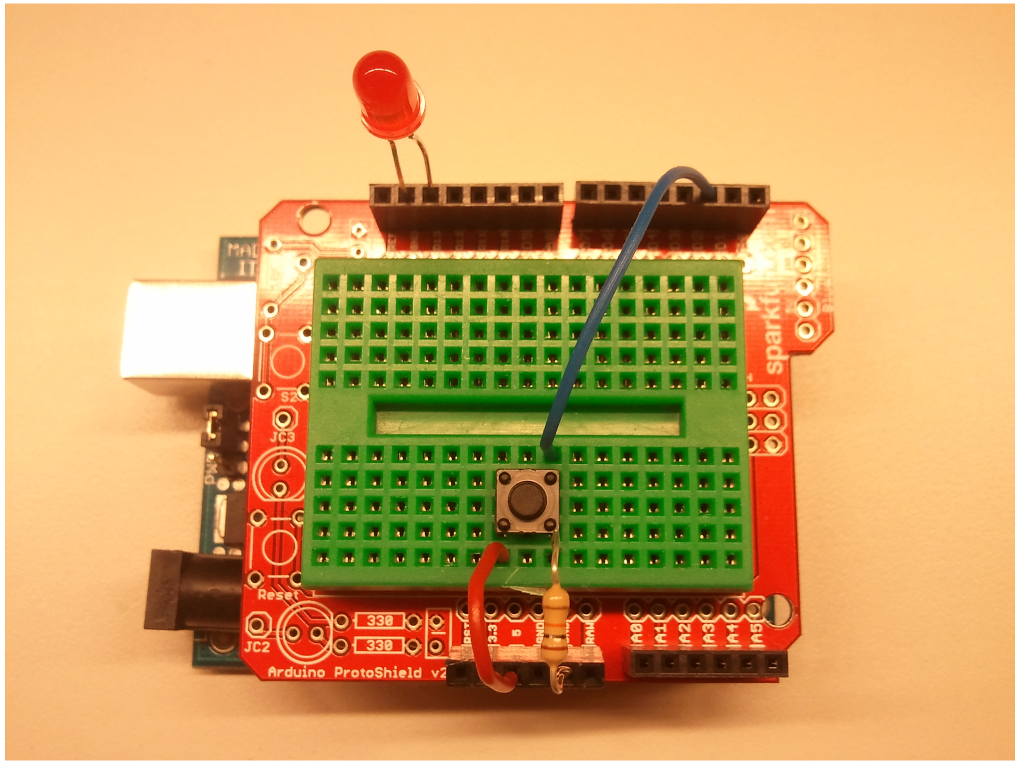


```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

```
Done compiling.

Binary sketch size: 1098 bytes (of a 14336 byte maximum)

22
```

Ex. 1: Button + LED

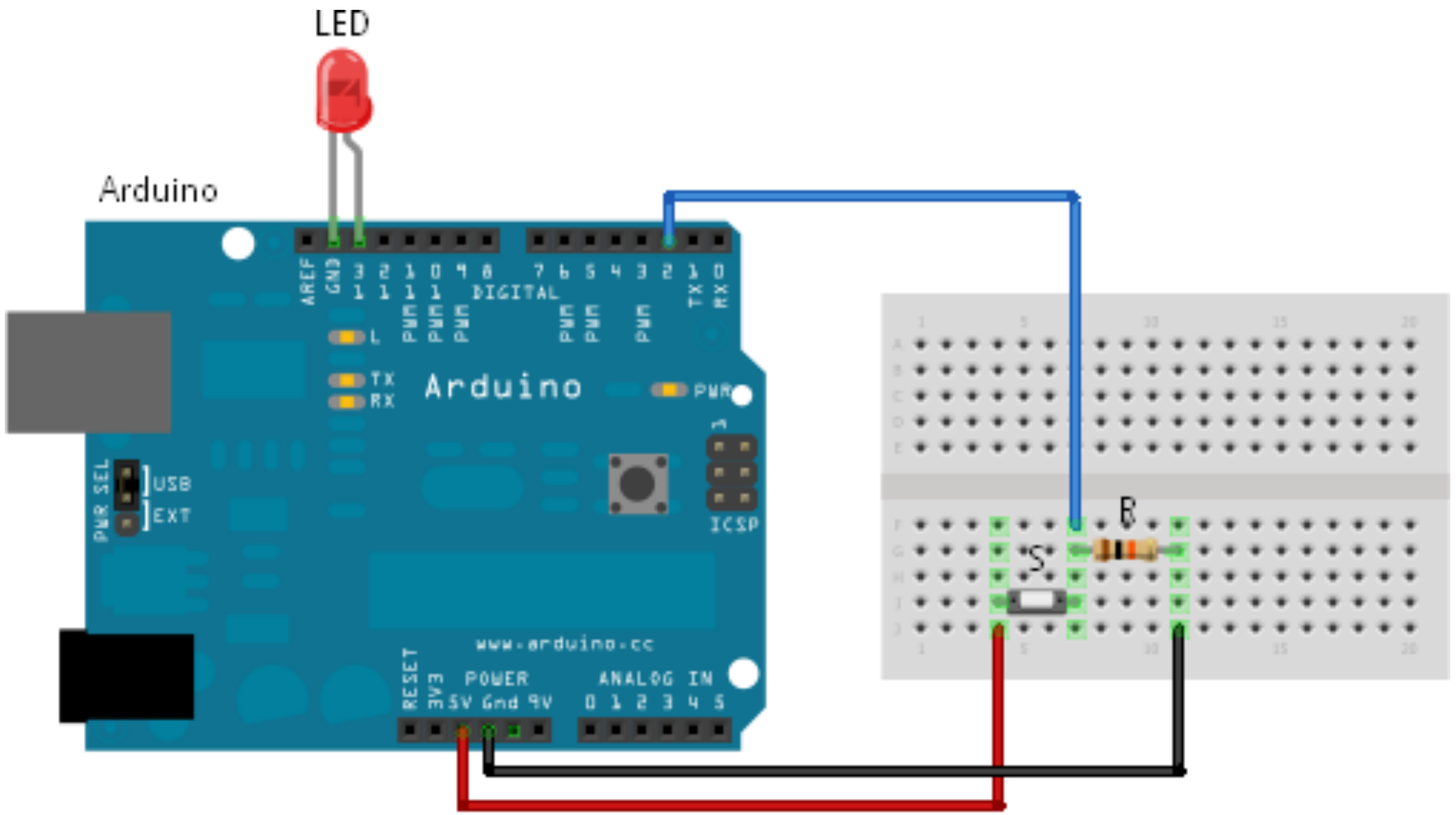
In this exercise we will mount a LED controlled by a button. The objective is turn on the LED on whenever the button is pressed, and turn it off when not pressed.



What we need:

1 x Arduino
1 x Protoboard
1 x Button
1 x LED
1 x 10 kΩ resistor

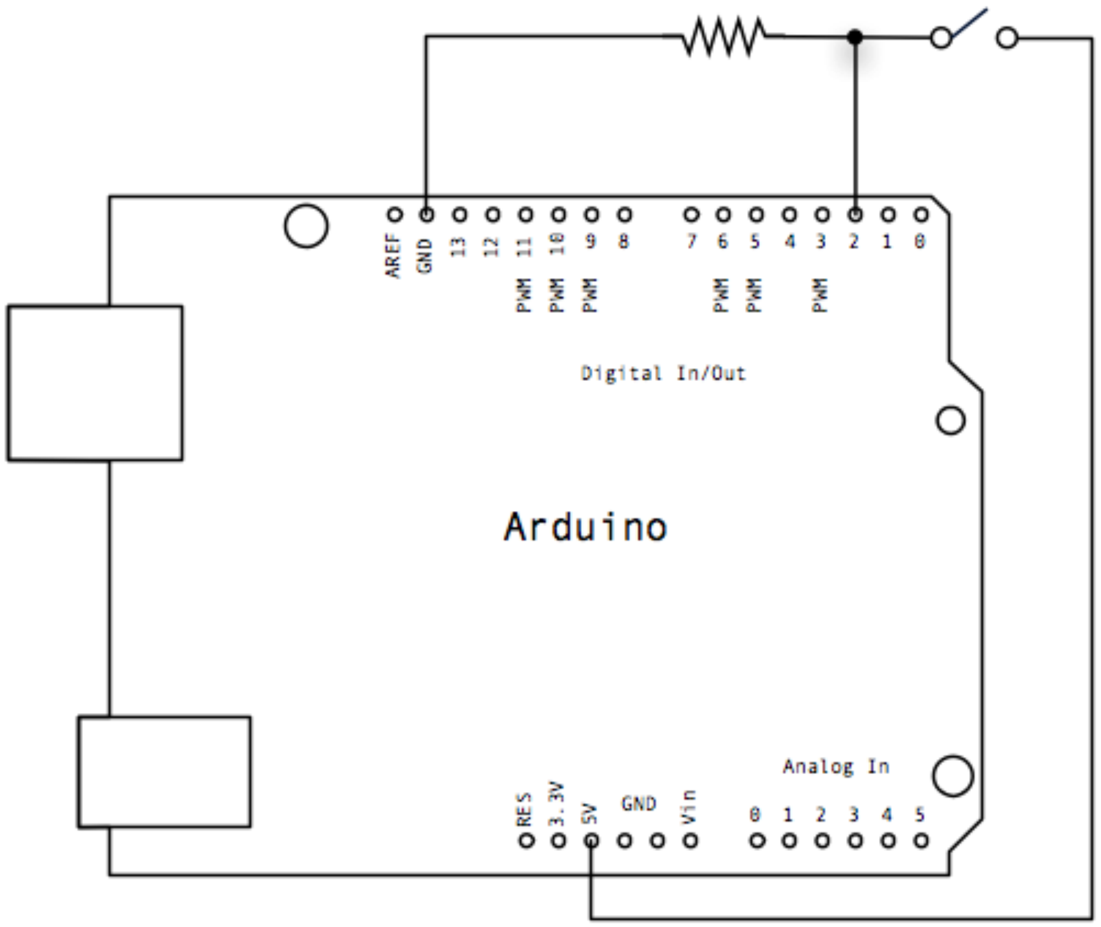
Ex. 1: Circuit scheme



R = 10 kΩ

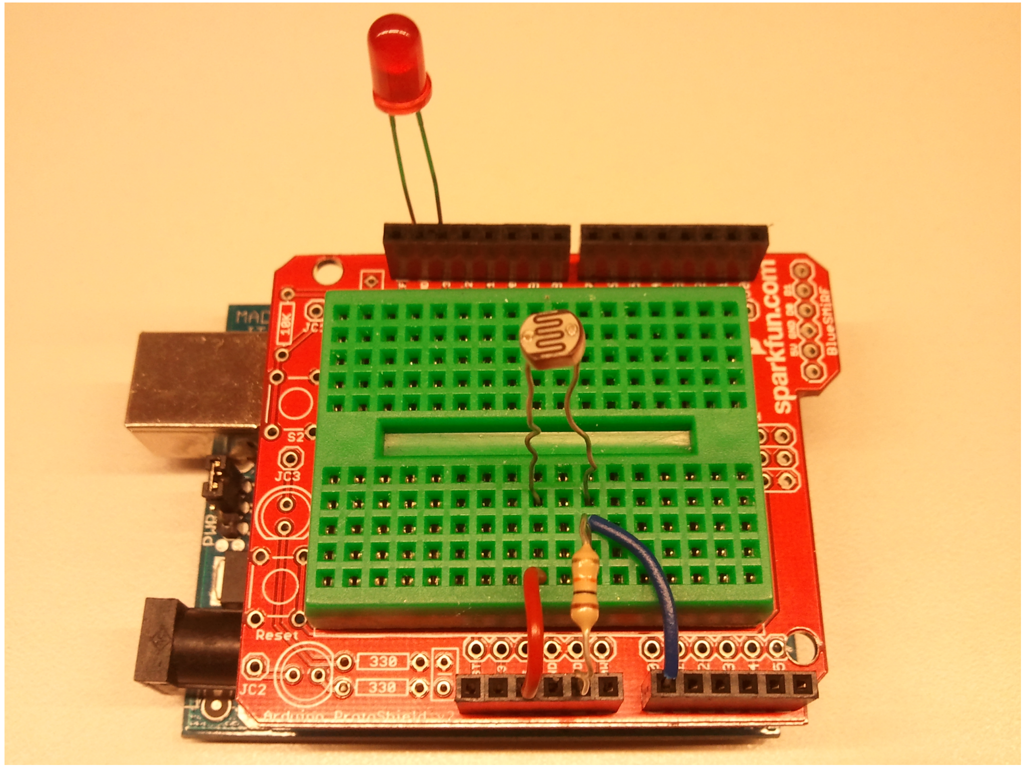
Ex. 1: Why do we need a resistor?

A pull-down resistor is needed because, without it, we can induce a dead short when the button is pushed down.



Ex. 2: LDR + LED

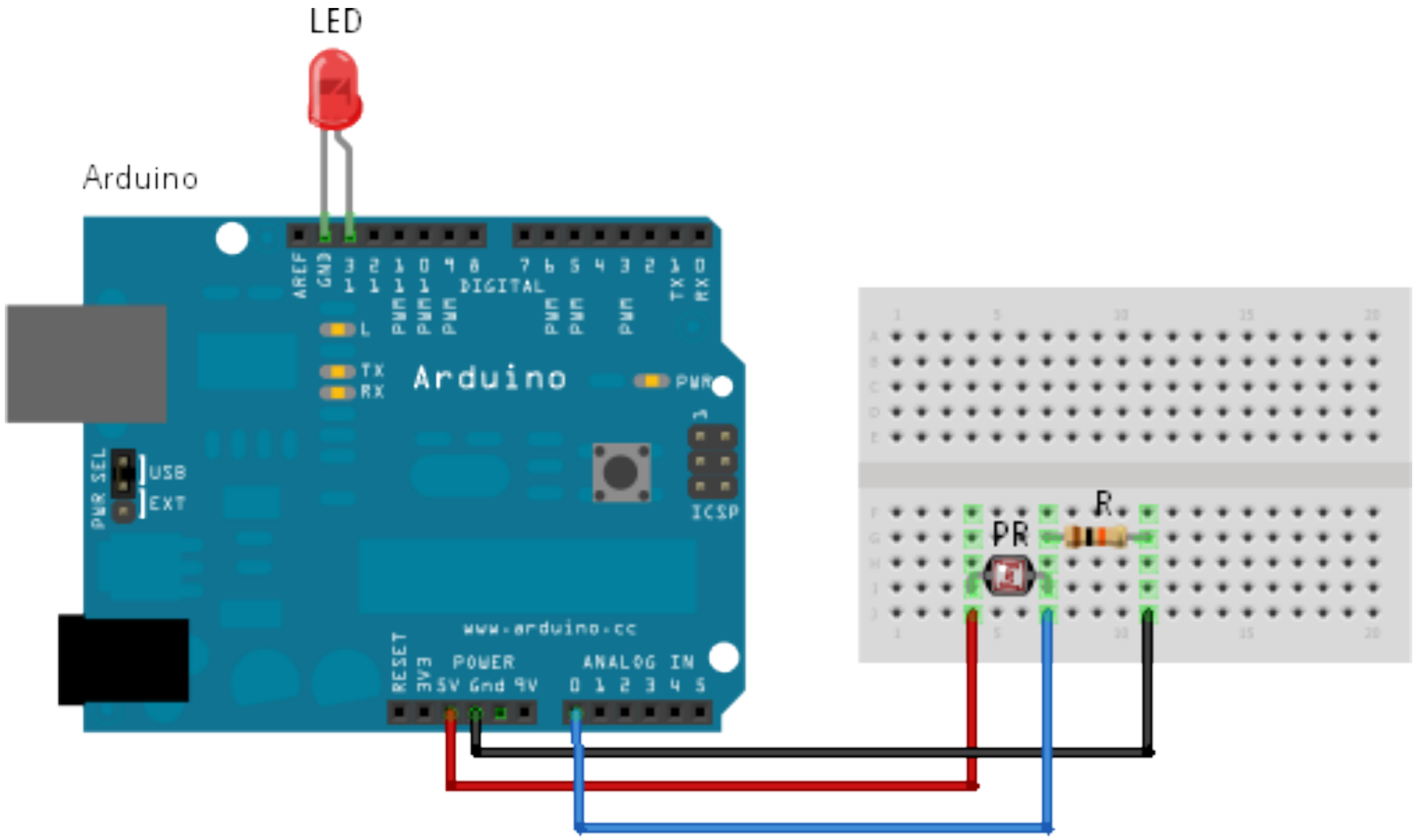
Now we will replace the button by a LDR (Light Dependent Resistor). The objective is turn on the LED when the LDR is covered.



What we need:

1 x Arduino
1 x Protoboard
1 x LDR
1 x LED
1 x 10 kΩ resistor

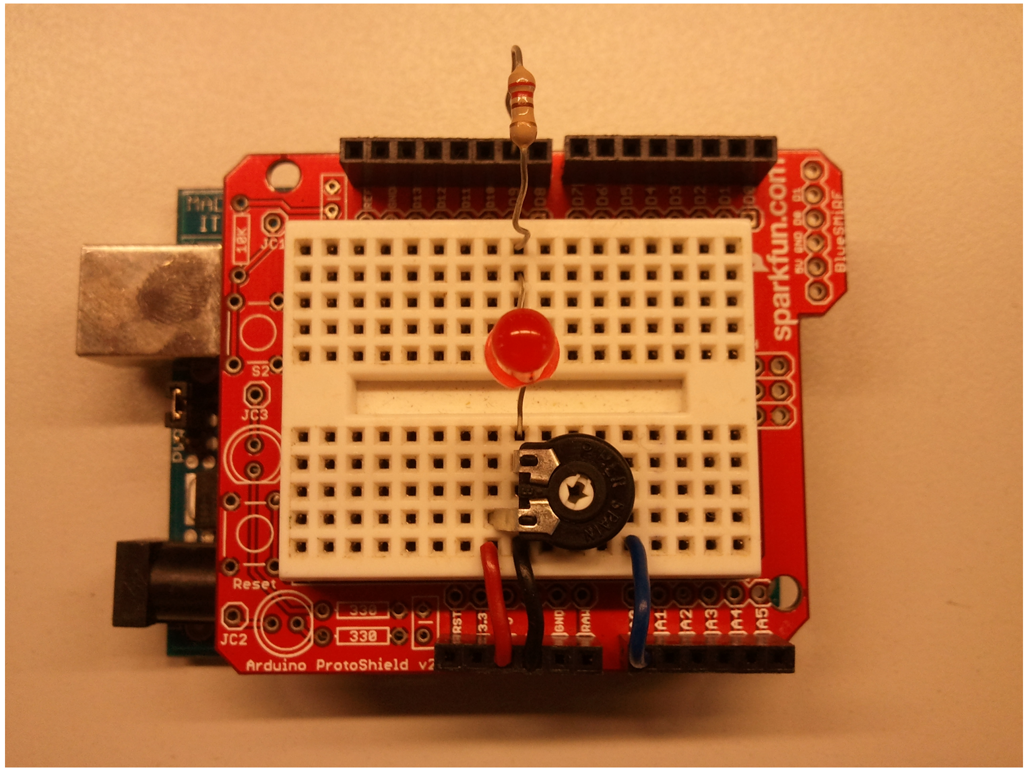
Ex. 2: Circuit scheme



R = 10 kΩ

Ex. 3: Potentiometer + LED

Now we're going to use a potentiometer instead of fotoresistor, with which we'll control the light of the LED gradually. This means that we need to use a PWM pin (for example 5) instead of the pin 13:



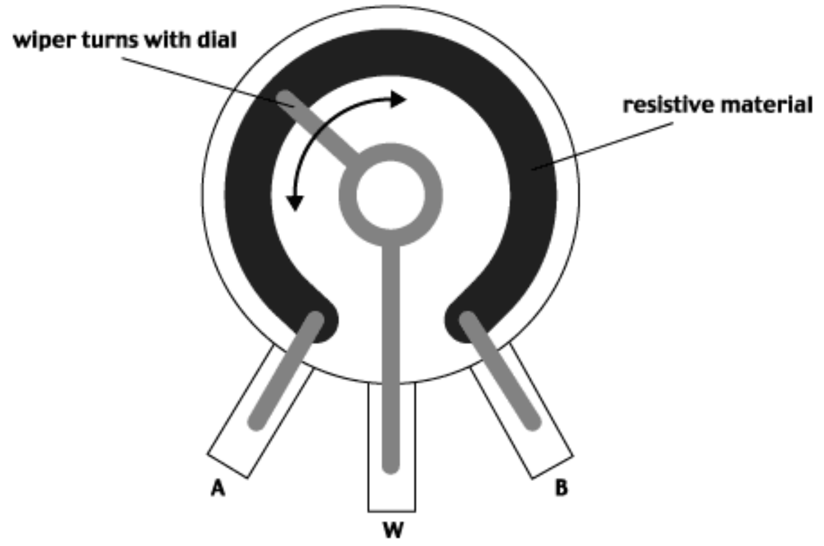
What we need:

1 x Arduino
1 x Protoboard
1 x Potentiometer
1 x LED
1 x 220 Ω resistor

Ex. 3: What's a potentiometer?

A potentiometer is a variable impedance device that provides an analog value. In our case, using an Arduino board, the range will be between 0 and 1023.

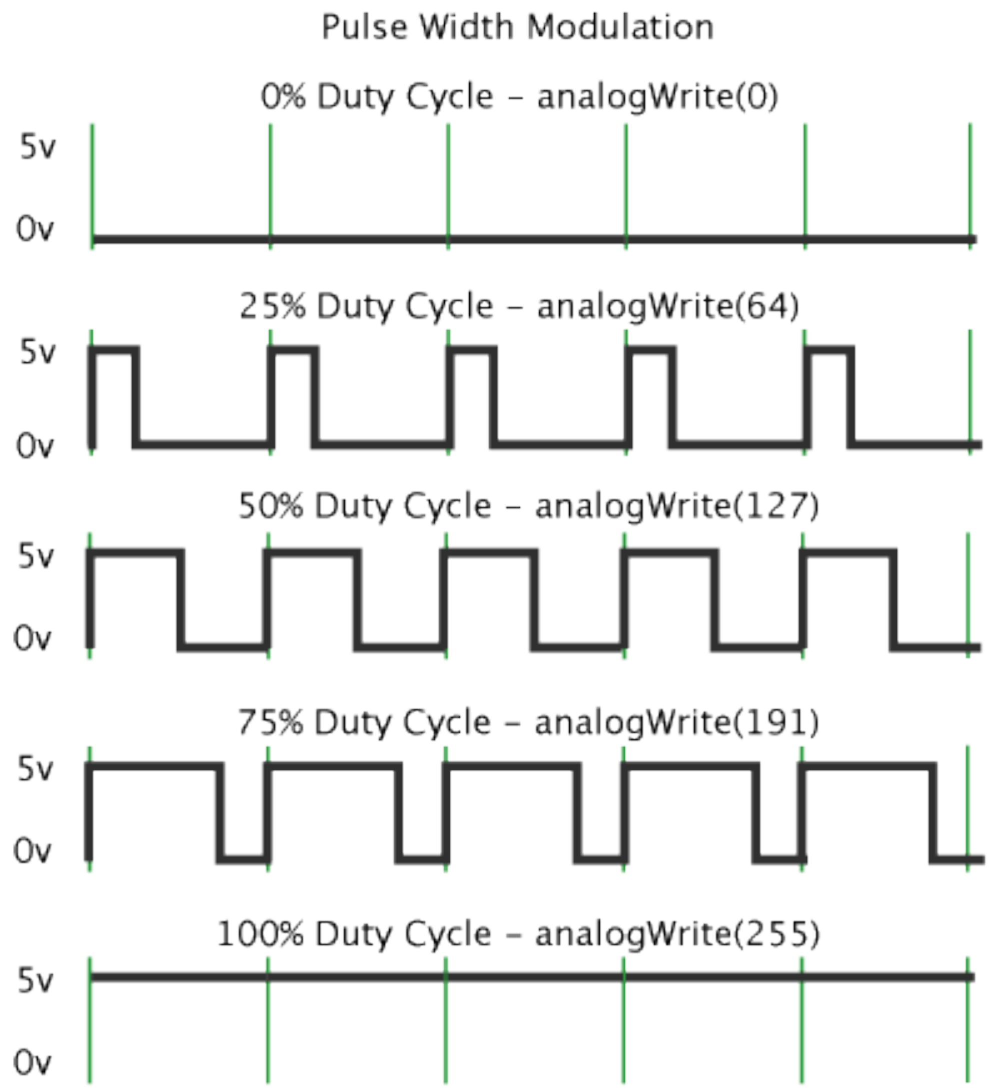
Its resistance is controlled by turning its axis, as shown in the picture:



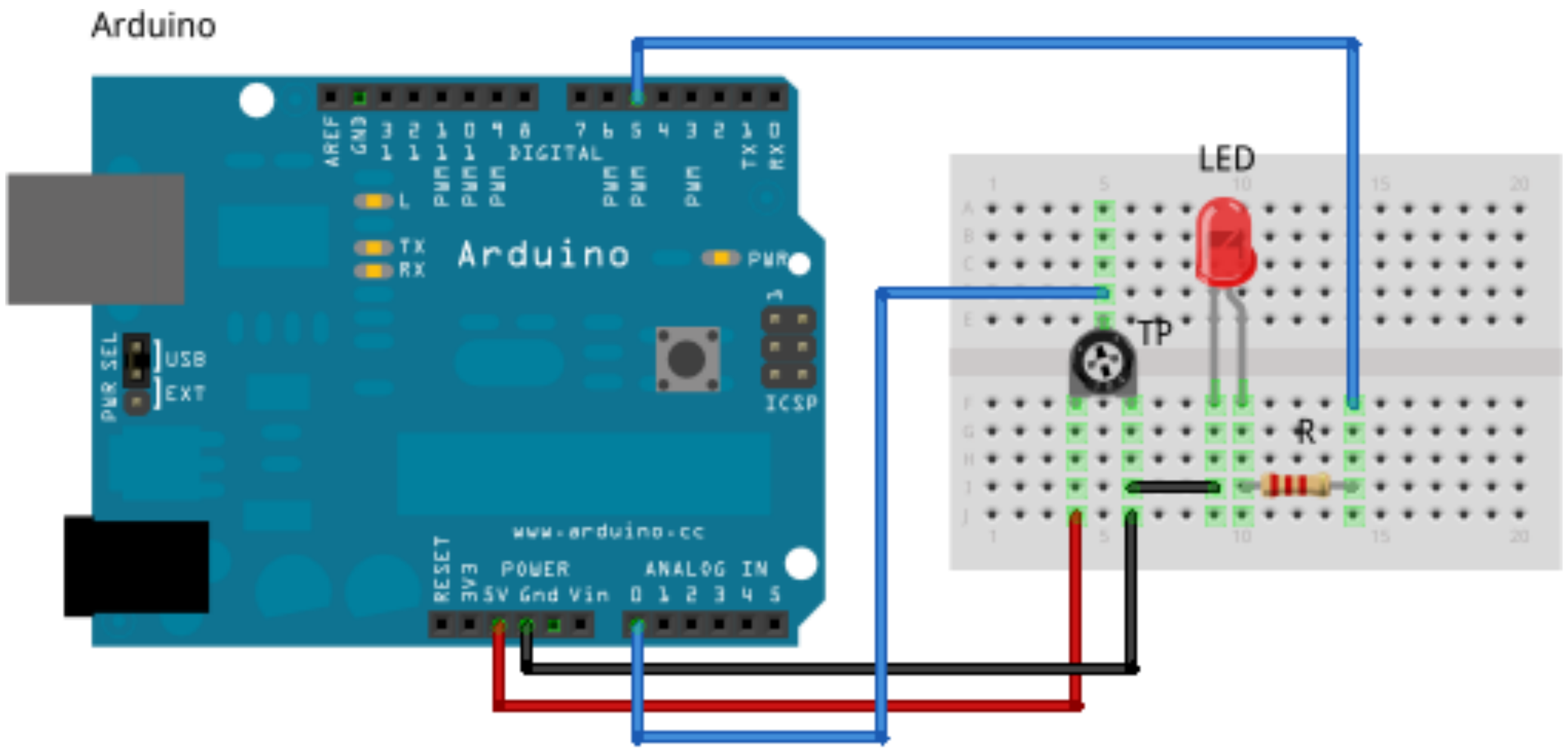
NOTE: *The value of the resistance rise or drop in one direction or the other depending on how you connect the power pins.*

Ex. 3: What does PWM mean?

The pulse width modulation (PWM) is a technique used to simulate an analog output with a digital one, creating a square wave that constantly switches between on and off. The time when the wave is 5V (ON) is called pulse width, which is modified to change the analogue value.


Pulse Width Modulation
0% Duty Cycle - analogWrite(0)
25% Duty Cycle - analogWrite(64)
50% Duty Cycle - analogWrite(127)
75% Duty Cycle - analogWrite(191)
100% Duty Cycle - analogWrite(255)
5v
0v

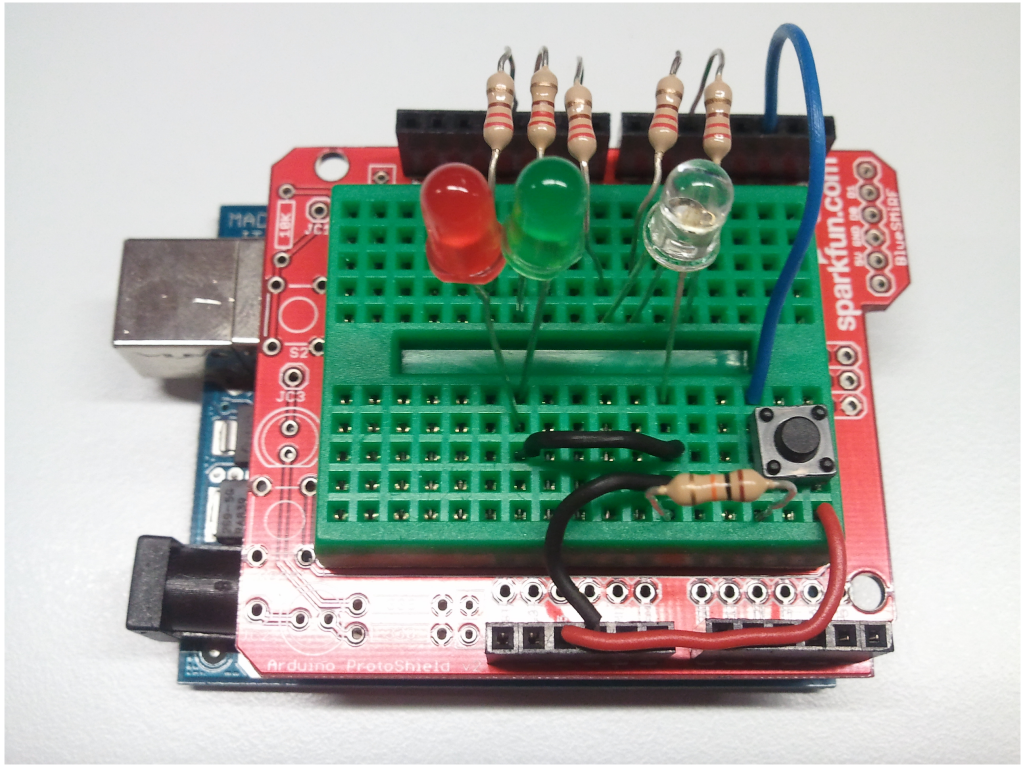
Ex. 3: Circuit scheme



TP = 4,7 KΩ
R = 220 Ω

Ex. 4: Semaphore

In this exercise we'll try to simulate the operation of a traffic light, those which have a button to facilitate the passage of pedestrians. The goal is to simulate the behavior of these lights with a simple program.

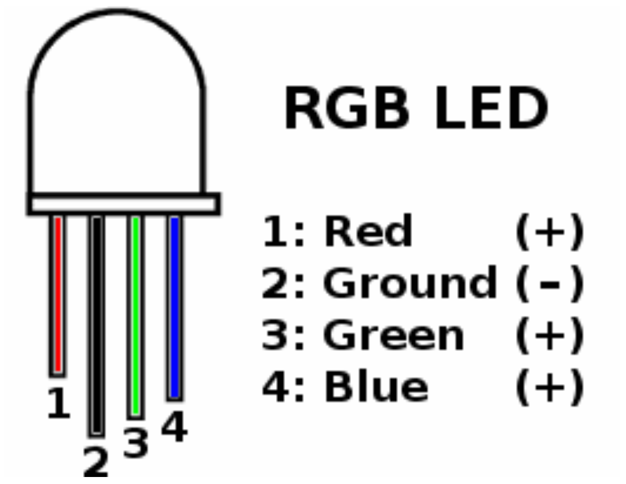


What we need:

1 x Arduino
1 x Protoboard
1 x Switch
2 x LEDs (red and green)
1 x LED RGB
4 x 220 Ω resistors
1 x 10k Ω resistor

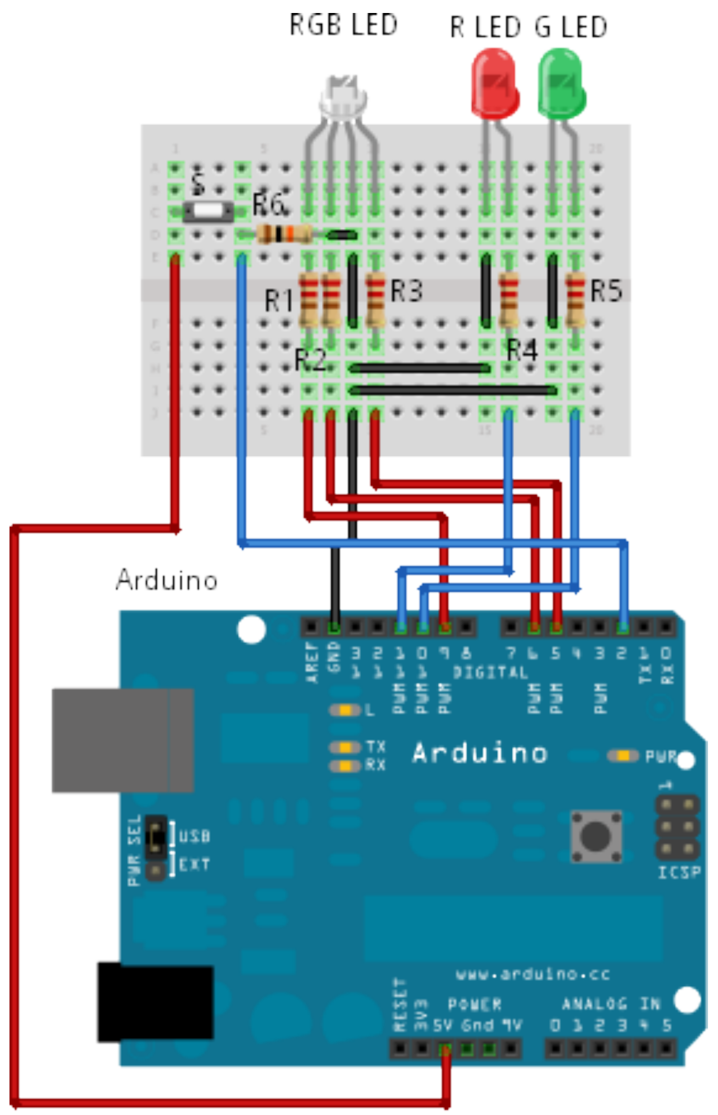
Ex. 4: RGB LED

The RGB LEDs provide a full spectrum of color (in contrast to normal ones, which give only one color), resulting in a light composed of three primary colors. Depending on the intensity of each color, we can get a lot of different shades. It is very important to know wich color controls each LED leg:



Remember that we must use PWM outputs (5, 6 and 9) instead of digital (as in the previous exercise).

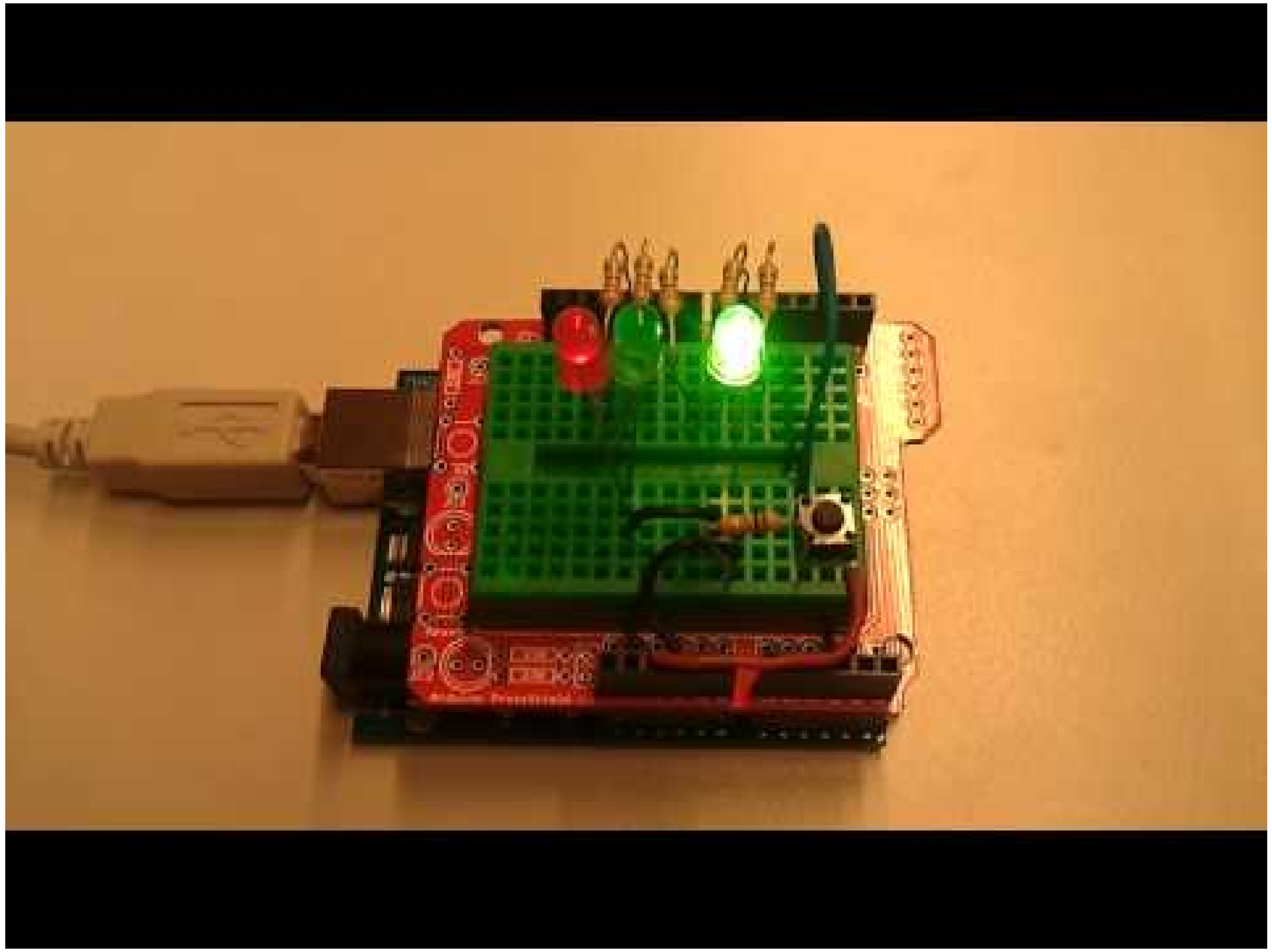
Ex. 4: Circuit scheme



R1 = R2 = R3 = R4 = R5 = 220 Ω
R6 = 10 kΩ

Ex. 4: Video demonstration

Here we can see a video demonstration of this exercise:



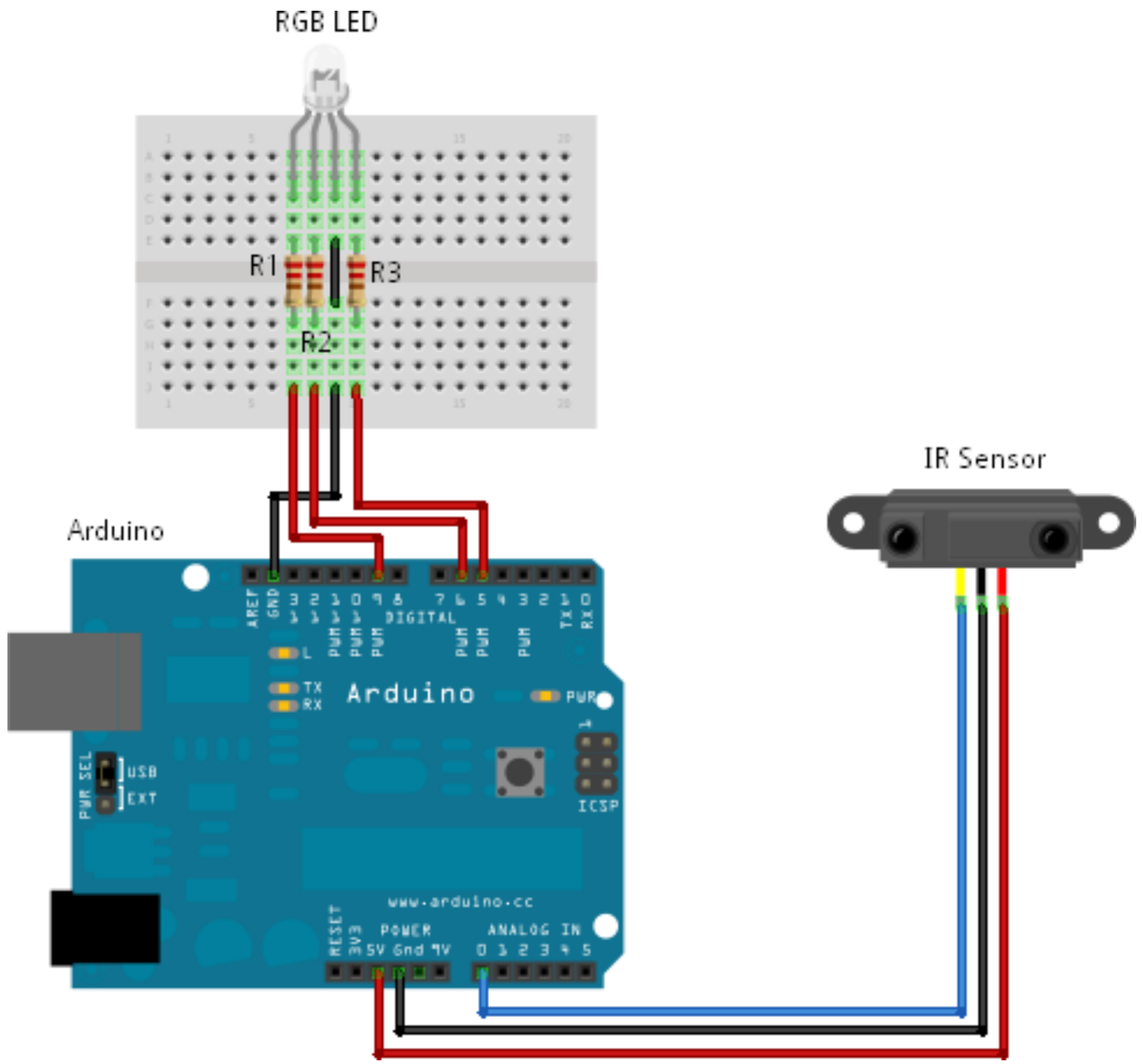
Ex. 5: RGB LED + Infrared sensor

The goal of this exercise will be the interaction between objects, using messages or variable changes. We'll also see and use the lists for the first time, and the elements in programming as **flags**.

To do so entertaining we will create a "virtual musical instrument."

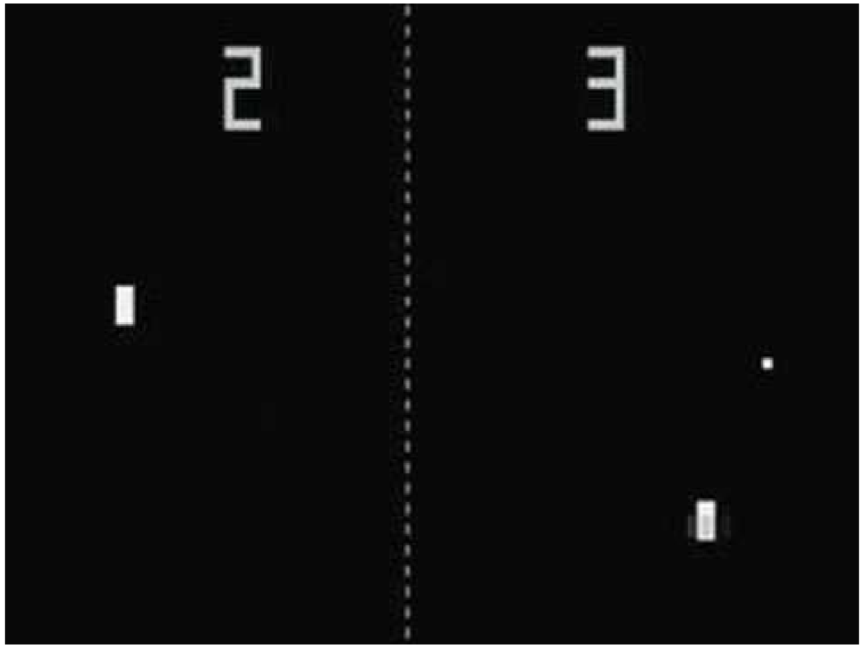
This will be the first time we program more than one object. The goal is to split the program into different objects and that each has a different function.

Ex. 5: Circuit scheme



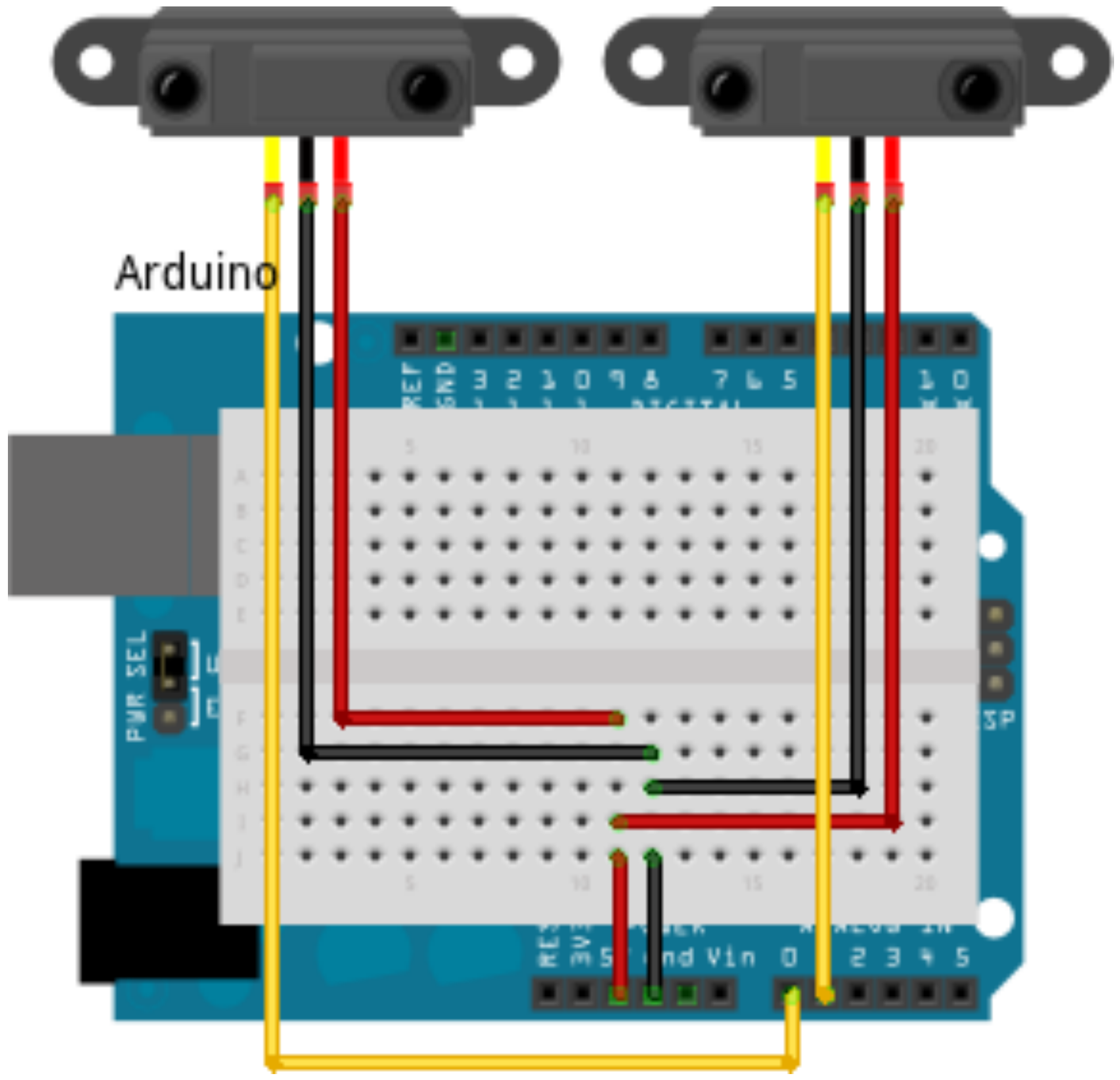
Ex. 6: Pong with sensors

Now we'll try to remake one of the first video game in history, the **PONG:**



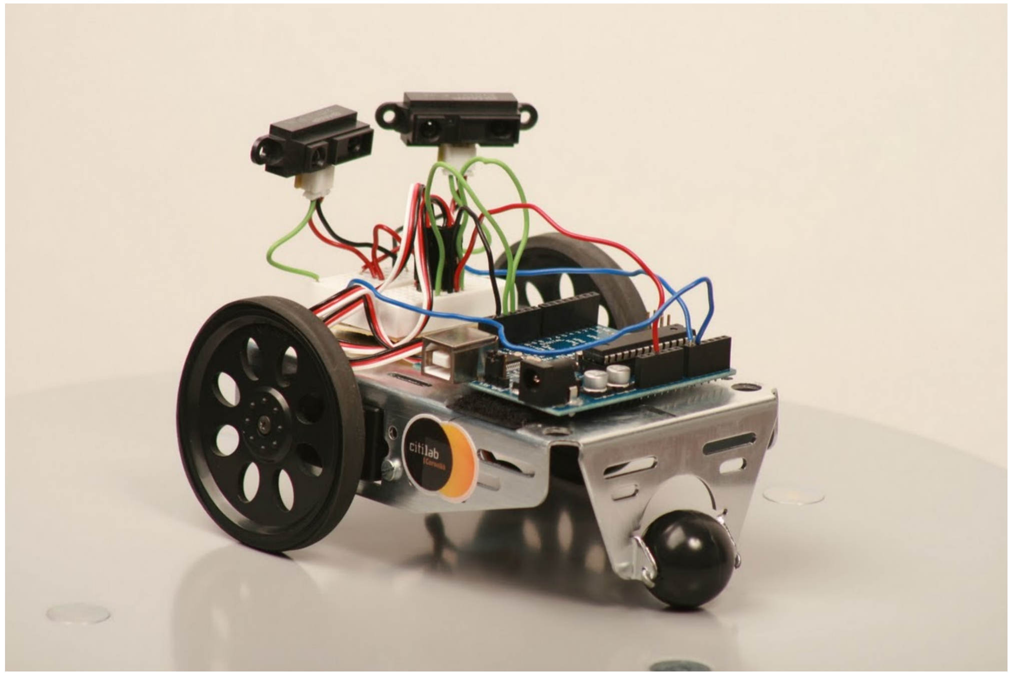
The goal is to use two sensors on the same board, and assign one to each player. In addition, we can incorporate a button to start the game, LEDs that light every time the players score a point, and so on...

Ex. 6: Circuit scheme



Ex. 7: Robot with CR servos

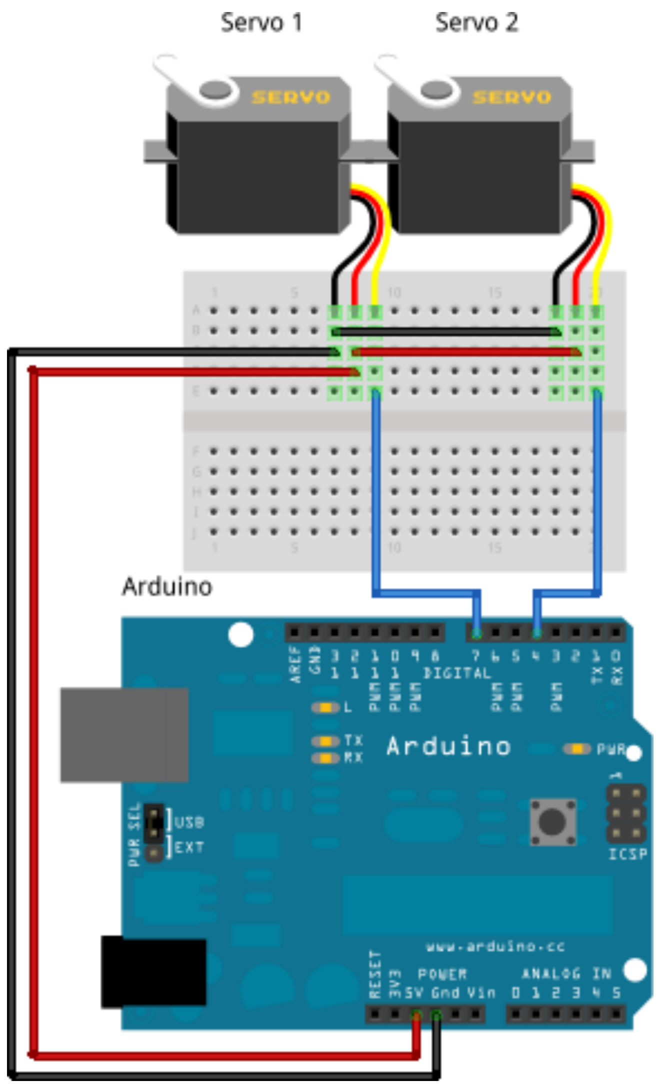
In this session we'll use servos for the first time. The first part of the exercise is to create a robot controlled by the computer keyboard (with the direction arrows, for example). The second is to make this robot autonomous, and capable to go alone and avoid objects with the help of sensors.



What we need:

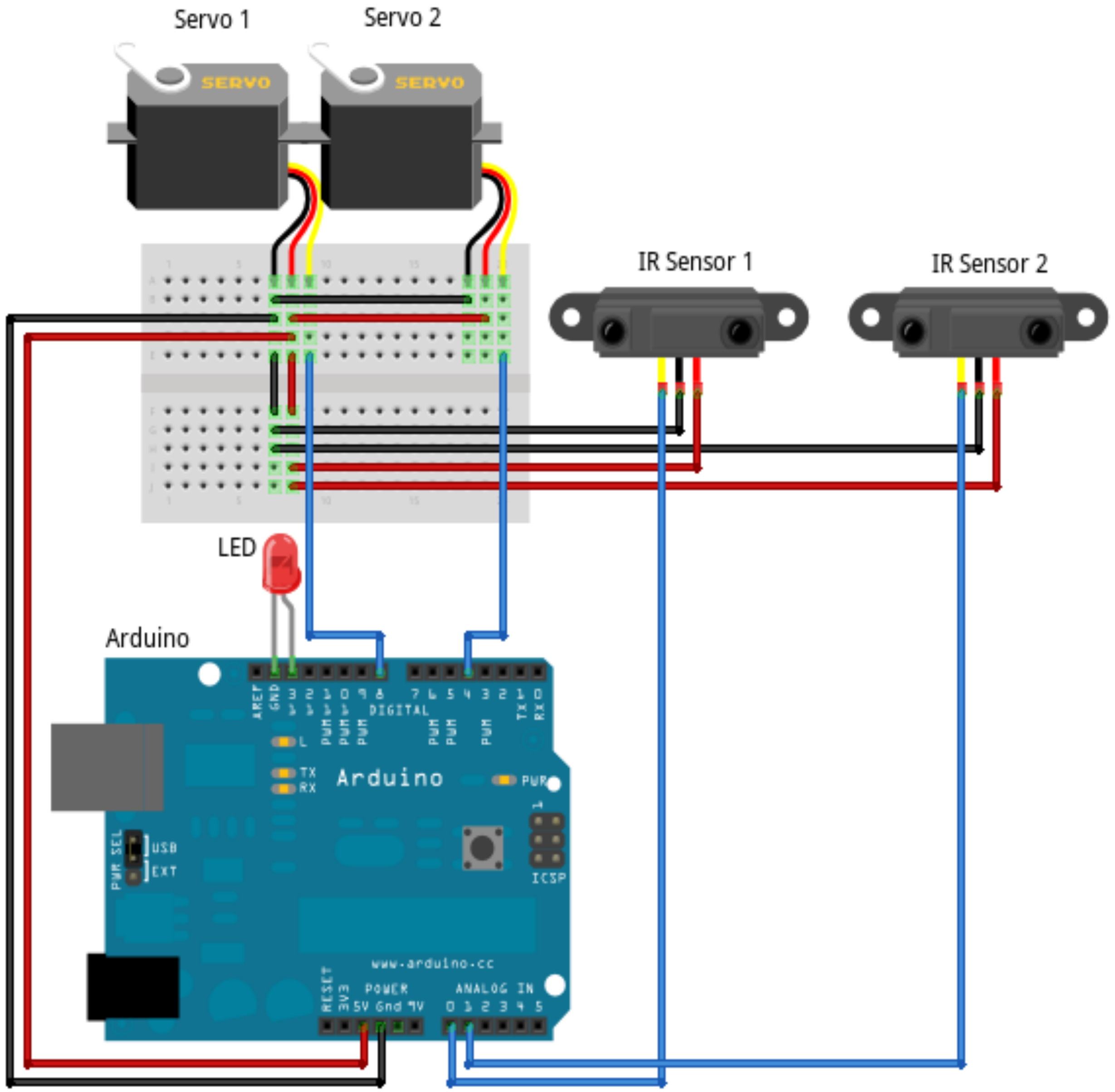
1 x Arduino
1 x Protoboard
2 x CR Servomotors (Futaba)
2 x Infrared sensors
1 x Chassis, wheels and ball
1 x LED (optional)

Ex. 7: Circuit scheme 1



As shown in the diagram, each servo has 3 wires: one red (5V), one black (GND) and one yellow, which will be connected to the pin we want to use to send the order to the servomotor. For this we have pins 4 and 7. As you can see the assembly is quite simple, just need a common point on the board to connect the power of each servo (red and black wires). The diagram in next page includes a pair of infrared sensors and an LED to make it autonomous.

Ex. 7: Circuit scheme 2


Servo 1
Servo 2
IR Sensor 1
IR Sensor 2
LED
Arduino