Co-funded by the
Erasmus+ Programme
of the European Union

Robots Go Green!

...OR HOW ROBOTICS CAN PRESERVE BIODIVERSITY AND HELP ENVIRONMENT AGAINST CLIMATE CHANGE.

EDUCATIONAL ROBOTICS MANUAL

INITIAL TRAINING FOR STUDENTS WITH ARDUINO UNO BOARD

MARCH 2020

AUTHORS:
Amparo Aparisi, Pablo Andrés, Paula Fernández, Pablo Gorbarán, Omaira Górriz, Jacobo Montesinos, Carlos Pozo, Nerea Ros, Joan Ruiz, Sandra Serrano.

LYCEE POLYVALENT JOSEPH ZOBEL RIVIERE SALEE

MARÍTIM IES DISTRICTE

INTRODUCTION TO

THIS MANUAL

This is a manual created as a guideline for students for the Erasmus+ project ROBOTS GO GREEN! 2019-1-DE03-K229-059795_3.

This manual has been written by students to help their international partners to learn basic robotics. Moreover, it has been focused on each student can learn by himself using free simulation software. It is not necessary to have previous knowledge of electronics to start but you can ask for help it is needed to your teacher.

In every practice, you can find a description of one or two electronics components, some references of coding in IDE Arduino, a guided practice, where the components will be connected and programmed and a challenge to solve and consolidate what has been learned.

ALL YOU NEED IS...LINKS TO ONLINE PROGRAMS AND OTHER HELP WEB PAGES

Electronic simulators

www.tinkercad.com/circuits for working online
www.fritzing.org if you prefer to work offline

Programming

Official Arduino webpage https://www.arduino.cc/

- IDE Arduino online or downloadable
  https://www.arduino.cc/en/Main/Software
- Language reference https://www.arduino.cc/reference/en/

Co-funded by the
Erasmus+ Programme
of the European Union

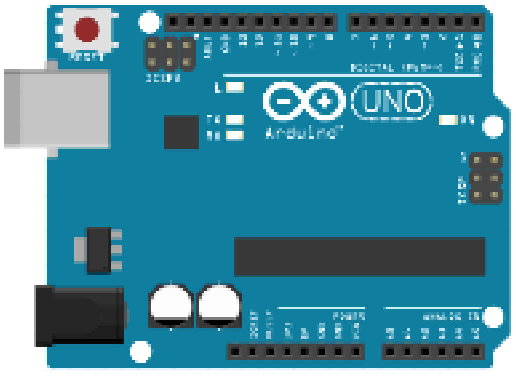
Index

Co-funded by the
Erasmus+ Programme
of the European Union

PART 1:

KNOWING ARDUINO UNO BOARD …

YOU'RE ABOUT TO LEARN…



I. ARDUINO BOARD:

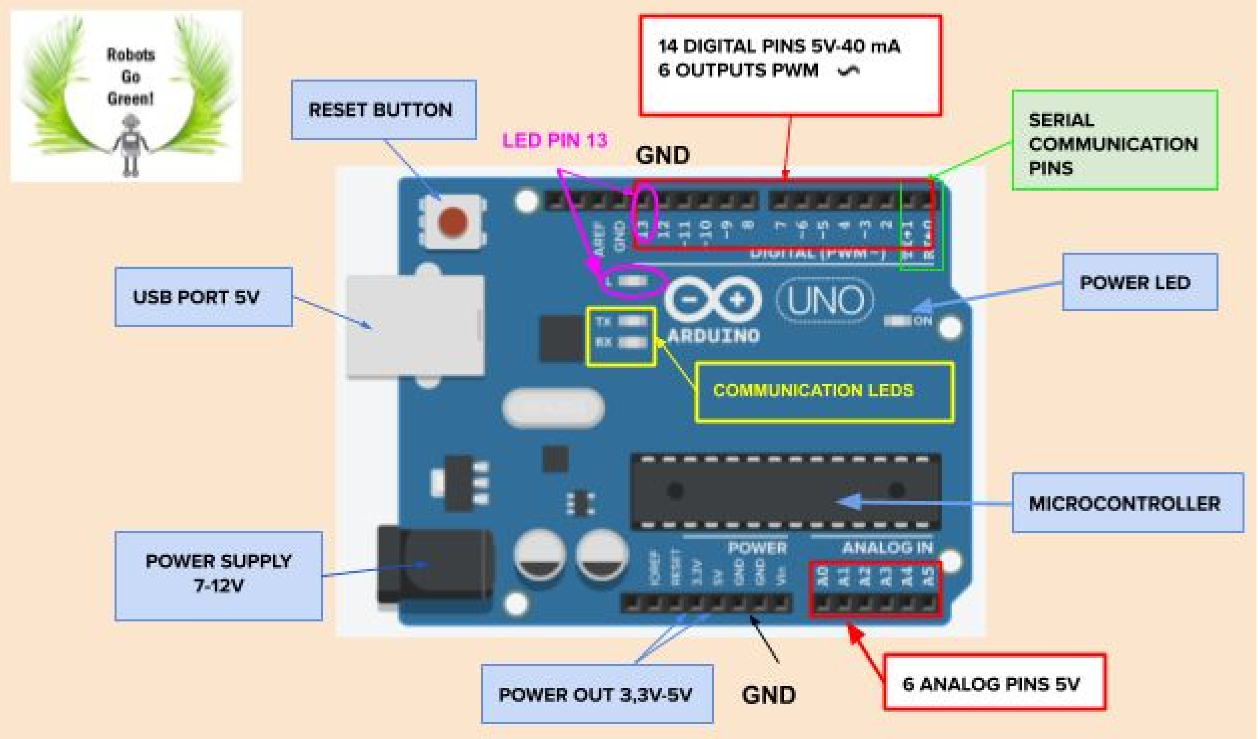
A. Circuit and how to represent by schema

The Arduino is a microcontroller-based board, specifically an ATMEL. A microcontroller is an *integrated circuit* (we could speak of a microchip) in which instructions can be recorded. These instructions are written using a programming language, an adaptation of the **C ++ language** for the ATMEL microcontroller, that allows the user to **create programs that interact with electronic circuits**.

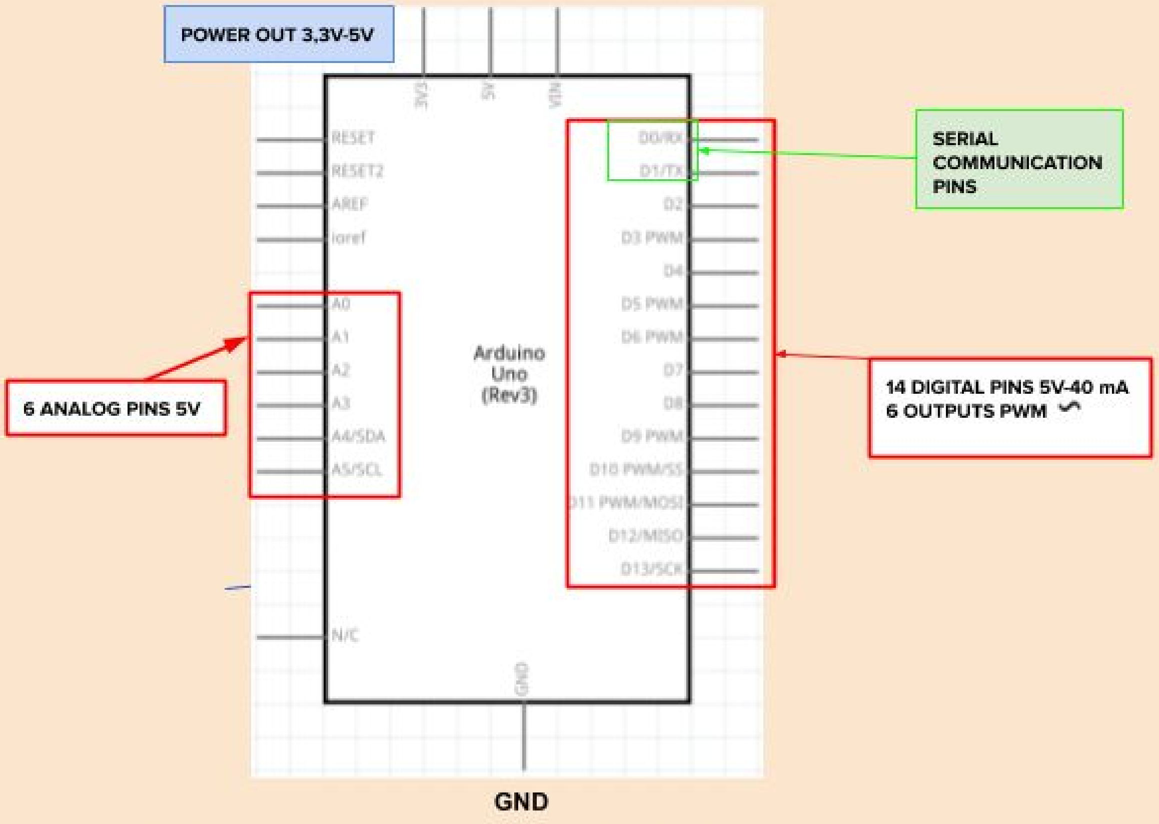
In all the boards the pins are multifunction or multipurpose, that is, depending on the configuration they have one functionality or another.
Digital pins (*from 1 to 13*) represent discrete values, rather than values within a certain range. For example, the light switch can only take two values or states: open or closed, or the same lamp: on or off. **Analog pins** (*from A0 to A5*) allow connecting components with voltage until 5V.
In *Picture 1,* you can see the available connexions in a physical Arduino UNO Board. Meanwhile, in *Picture 2,* you can see an Arduino UNO schema. It is useful to represent in electronics, the connections using a schematic representation that simplifies electronic assembly of each component.

Co-funded by the
Erasmus+ Programme
of the European Union

BOARD & COMPONENTS



Picture 1: PARTS OF ARDUINO UNO R3 BOARD



II. PROGRAMMING ARDUINO BOARD

To program a computer is to give it instructions. You express those instructions with **programming code** that is comprised of human-readable commands. These commands are transformed into programs which computers understand. A computer really only understands 0's and 1's. This is why programming needs to be done in a text editor with special properties. This editor is a software development system called Integrated Development Environment **(IDE)** and can translate programming code into machine language. Translating the code this way is called **compiling**. (You can download Arduino IDE or maybe working online in this page https://www.arduino.cc/ but Tinkercad simulator has Arduino IDE included)

Arduino programming language can be divided in three main parts

- Structure and special language: The elements of Arduino (*C++*) code.
-Functions: For controlling the Arduino board and performing computations.
-Variables: Arduino data types and constants.

CODING

A. STRUCTURE OF A SKETCH:

Void setup() {....} The setup() function will only run once, after each power-up or reset of the Arduino board. Use it when a sketch starts to initialize variables, pin modes, start using libraries, etc. (+inf)

Void loop() {....} The loop() function will run forever (or until you disconnect the power source). The commands in loop() are executed in order, one after another. When reaching the last line, it will start from the beginning again. (+ inf) Note: Use exit(0); to finish loop.

B. CONTROL STRUCTURE:
Operations you can make only using these keywords and the correct syntax.

for (initialization; condition; increment) {...} The *for* statement is useful for any repetitive operation. (+ inf).

initialization: happens first and exactly once.
condition: each time through the loop, the condition is tested; if it's true, the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.
increment: executed each time through the loop when the condition is true.

C. VARIABLES:
Different values, constants or variables, along with the sketch. They are defined before or setup or inside functions.

int var; The variable var is declared as an integer; int var = val; The variable var is declared to hold the value val, the number of the digital pin you are using. You can use **#define** var=val; to define a variable

float var; The variable var is declared as a float point number;

Digital signals:

INPUT/OUTPUT: Digital pins can be used as INPUT, INPUT_PULLUP, or OUTPUT. Changing a pin with pinMode() (*see Functions*)changes the electrical behaviour of the pin.

BE CAREFUL: Pins configured as inputs with either INPUT or INPUT_PULLUP can be damaged or destroyed if they are connected to voltages below ground (negative voltages) or above the positive power rail (5V or 3V). Pins configured as OUTPUT with pinMode() can source (provide current) or sink (absorb current) up to 40 mA (milliamps) of current to other devices/circuits. This makes them useful for powering LEDs because LEDs typically use less than 40 mA. Loads greater than 40 mA (e.g. motors) will require a transistor or other interface circuitry.

HIGH/LOW: When reading or writing to a digital pin there are only two possible values a pin can take/be-set-to: HIGH and LOW.

D. FUNCTIONS For controlling the Arduino board and performing computations

Digital I/O

pinMode(pin, mode); Configures the specified pin to behave either as an input or an output. (+inf)

digitalWrite(pin, value); Write a HIGH or a LOW value to a digital pin .(+inf)

Time

delay(ms); Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)(+inf)

E. SPECIAL LANGUAGE:

Comments: Lines in the sketch that are used to help you understand (or remember), or to inform others about how your program works. They are ignored by the compiler.

/*... */ block comment or a multi-line comment is marked by the symbol /* and the symbol */ marks its end. (+inf)

... // single-line comment. This comment ends automatically at the end of a line. (+ inf)

III. PRACTICE

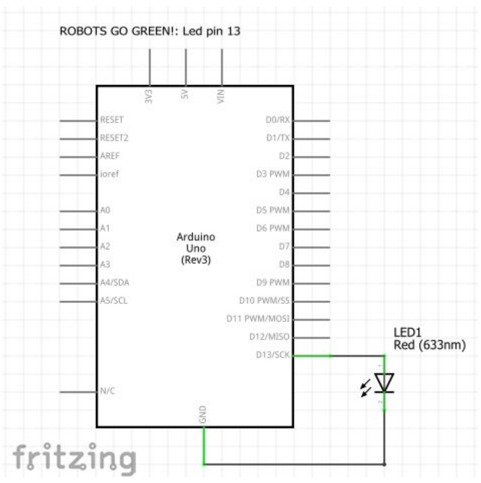
Practice I: Lighting the board light (BLINK)

We are about to learn how to create the first program, lighting the led included in the board (PIN 13)

PRACTICES: SCHEMA & CODE

PRACTICE I: Lighting the board light (BLINK)

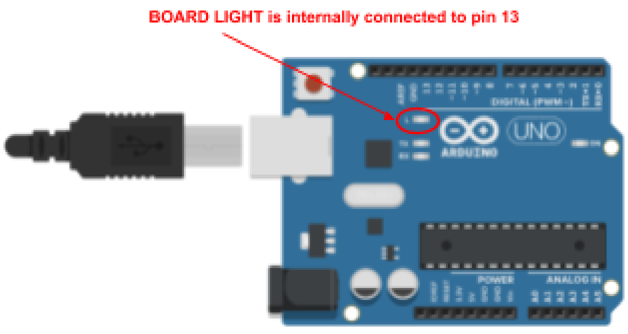
```
  // Robots go green!
  /*Practice I: "Lighting the board light" We are about to learn how to
  create the first program, lighting the led included in the board*/
void setup()   //
{
  pinMode(13, OUTPUT); // Declare pin 13 as an OUTPUT
}
void loop()
{
  digitalWrite(13, HIGH); // Write in the pin 13 a HIGH value...switch on
  delay(1000);       // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW); // Write in the pin 13 a LOW value...switch off
  delay(1000);       // Wait for 1000 millisecond(s)
}
```



PRACTICE II: Lighting the board light only 3 times (for)

Co-funded by the Erasmus+ Programme of the European Union

```
  // Robots go green!
  /*Practice II: "Lighting the board light only 3 times"
  We are about to learn how to use the function for*/

void setup()   //
{
  pinMode(13, OUTPUT); // Declare pin 13 as an
OUTPUT
}
void loop() // The statement will be done
continuously unless exit(0) instruction was written.
{
for (int i=1; i <= 3; i++){ //The following statement will
be done only 3 times
  digitalWrite(13, HIGH); // Write in the pin 13 a HIGH value...switch on
  delay(1000);       // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW); // Write in the pin 13 a LOW value...switch off
  delay(1000);       // Wait for 1000 millisecond(s)
}
exit(0); // It finishes the loop
}
```



IV. CHALLENGE :

I.1 Blinking faster

Using the code of PRACTICE 1, modify the program to make blinking faster.
SOLUTION: CHALLENGE I.1

I.2 SOS in Code Morse

Program a code blinking the board LED to make SOS signal in Morse code (THREE SHORT BLINKS, THREE LONG BLINKS, THREE SHORT BLINKS). Try to do it using the sentence **for**

SOLUTION: CHALLENGE I.2

PART 2:
PLAYING WITH THE LIGHT: DIGITAL INPUTS AND OUTPUTS

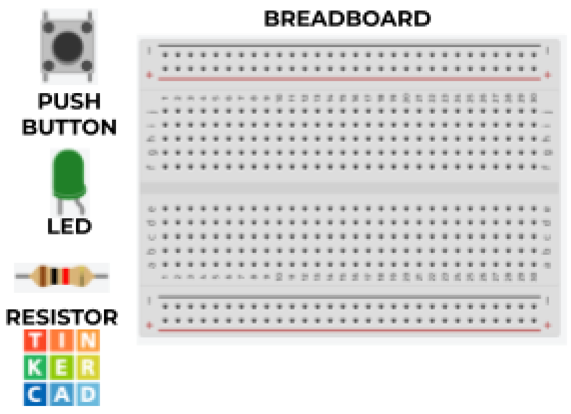
YOU'RE ABOUT TO LEARN...

I. **COMPONENTS:**
   - A. Breadboard and how to connect components
   - B. LED
   - C. Resistors
   - D. Pushbutton

II. **PROGRAMMING:**
   - A. Control structure:
     1. ***if., else***
     2. ***Comparison expressions***
   - B. Functions:
     Digital I/O: digitalRead(pin);

III. **PRACTICE:**
   Practice 2.I: Two Flashing LEDs
   Practice 2.II: Light a LED with a push button

IV. **CHALLENGE:**
   Challenge II.1: Traffic lights
   Challenge II.2: Traffic lights at a crossroad



I. COMPONENTS:

BOARD & COMPONENTS

A. Breadboard and how to connect components:

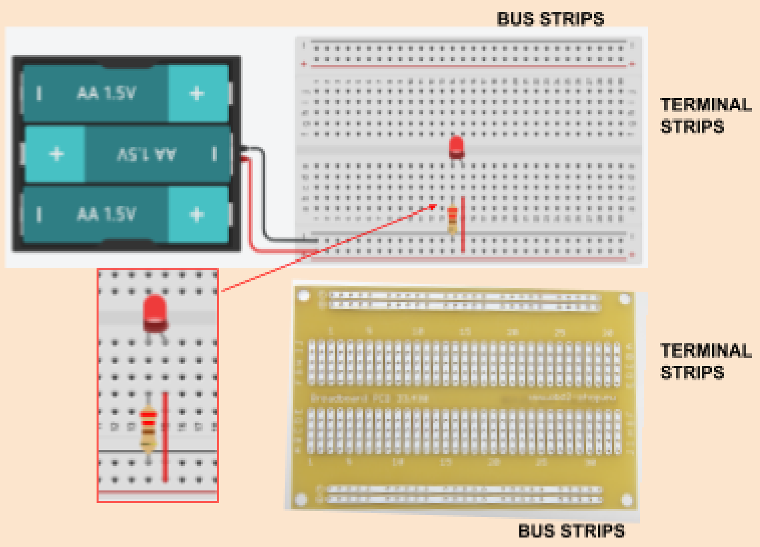
A **breadboard** socket is a quick circuit connection very useful for learning Electronics. It consists of a perforated block of plastic connect the pin to pin by metal strips inside the breadboard. The layout of a typical breadboard is made up of two types of areas, called strips.

Strips consist of interconnected electrical terminals.

Terminal strips: The main areas, to hold most of the electronic components.

Bus strips: To provide power to the electronic components.

(ref. https://en.wikipedia.org/wiki/Breadboard#)



Breadboard socket aspect and how it is inside. Components have to be connected between differents stripes.

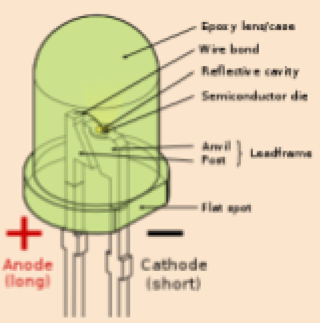
B. LED

A **LED** is a particular type of diode that emits light when it is traversed by an electric current. It is necessary to connect LEDs following its polarity, so, the correct way to connect a LED is shown in the picture.

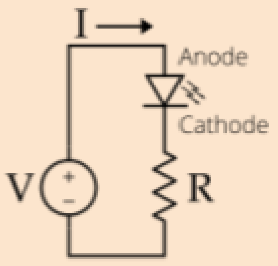
Attention: The current in an LED rises exponentially with the applied voltage, so a small change in voltage can cause a large change in current.
Current through the LED must be regulated by a current-limiting resistor to prevent damage.

ref. https://en.wikipedia.org/wiki/Light-emitting_diode

Parts of a LED



Symbol of a LED and connexion in a circuit with resistor for current limiting

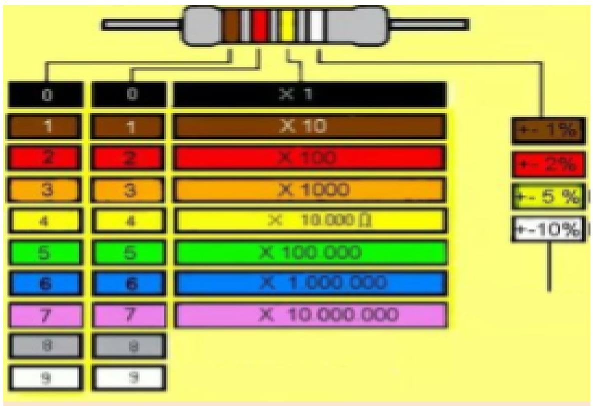


Parts of a LED and its electronic symbol. Circuit of basic connexion of a LED.

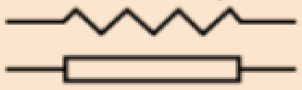
C. RESISTORS

A **resistor** is a component that implements electrical resistance (**Ω**) to a circuit. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, among other uses.
An **electronic colour code** is used to indicate the values of resistance only reading the colour of the stripes drawn on the component.(+inf)

ref: https://en.wikipedia.org/wiki/Resistor



Colors code for resistors.(*Picture ref. Flickr*)

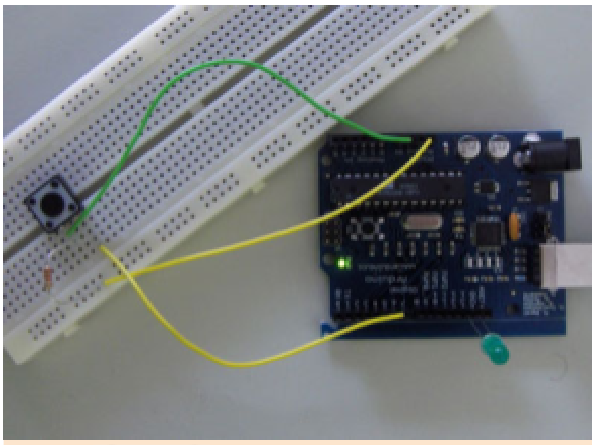


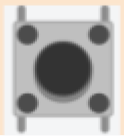
Typical electrical symbol

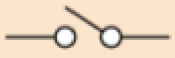
D. PUSHBUTTON

The **pushbutton** is a component that connects two points in a circuit when you press it.
We have to connect three wires to the Arduino board. The first goes from one leg of the pushbutton through a pull-up resistor (e.g. 2.2 kΩ)to the 5V supply.
The second goes from the corresponding leg of the pushbutton to ground.
The third connects to a digital i/o pin which reads the button's state defined as DIGITAL INPUT in the code.

ref. https://www.arduino.cc/en/Tutorial/Pushbutton







Pushbutton symbol

II. PROGRAMMING:

CODING

A. CONTROL STRUCTURE: Operations you can make only using these keywords and the correct syntax.

- **if** (condition){...} (initialization; condition; increment) {...} The **if** statement checks for a condition (comparison expressions) and executes the proceeding statement or set of statements if the condition is '**true**'. (+ inf).
- **else**: The if...else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped. An else clause (if at all exists) will be executed if the condition in the if statement results in false. The else can proceed another if test, so that multiple, mutually exclusive tests can be run at the same time.

```
if (condition1) {
  // do Thing A
}
else if (condition2) {
  // do Thing B
}
else {
  // do Thing C
}
```

- **Comparison expressions:** Useful for conditions

  x == y (x is equal to y)
  x != y (x is not equal to y)
  x < y (x is less than y)
  x > y (x is greater than y)
  x <= y (x is less than or equal to y)
  x >= y (x is greater than or equal to y)

B. FUNCTIONS For controlling the Arduino board and performing computations

Digital I/O

digitalRead(pin); Reads the value from a specified digital pin, either HIGH or LOW. (+inf)

III. PRACTICE:

Practice I: Lighting the board light (BLINK)

PRACTICES: SCHEMA & CODE

PRACTICE 2. I:Two flashing LEDs

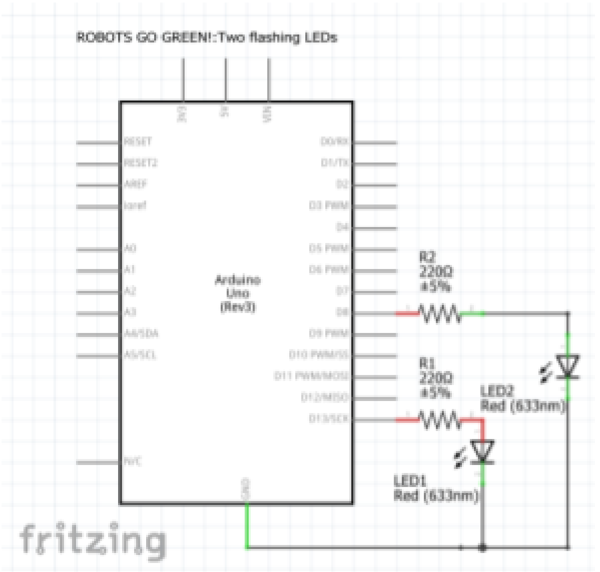
Alternately blink two LEDs:

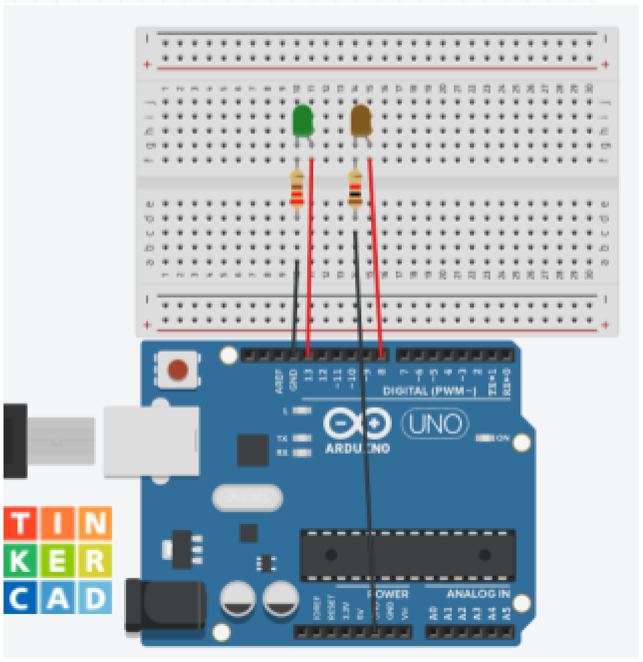
- One LED on and the other off.
- Two LEDs on and two LEDs off at the same time.

Components:
LED1 (D13)
LED2 (D7)
2 Resistors 220Ω

```
// Robots go green!
/*Practice 2.I: "Two flashing LEDs: One led on and the other
off. "*/

void setup() {
    pinMode(13, OUTPUT); //define the green led as an output
    pinMode(8, OUTPUT); //define the yellow led as an output
  }

  void loop() {
    digitalWrite(13, HIGH);//the green will light up
    digitalWrite(8, LOW); // the yellow will be off
  delay(1000);//wait 1000 ms
    digitalWrite(13, LOW);//the green will be off
  digitalWrite(8, HIGH);//the yellow will light up
    delay(1000);//wait 1000 ms
  }
```

```
// Robots go green!
/*Practice 2.I: "Two flashing LEDs:Two leds on, the two off..
"*/
  void setup(){
  pinMode(13, OUTPUT);//define the green led as an output
  pinMode(8, OUTPUT);//define the yellow led as an output
  }
  void loop(){
  digitalWrite(13, HIGH);//the green will light up
  digitalWrite(8, HIGH);//the yellow will light up
  delay(1000);//wait 1000 ms
  digitalWrite(13, LOW);//the green will be off
  digitalWrite(8, LOW);//the yellow will be off
  delay(1000);//wait 1000 ms
  }
```



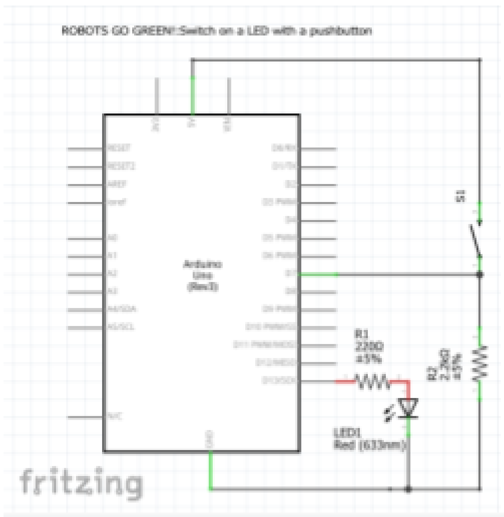


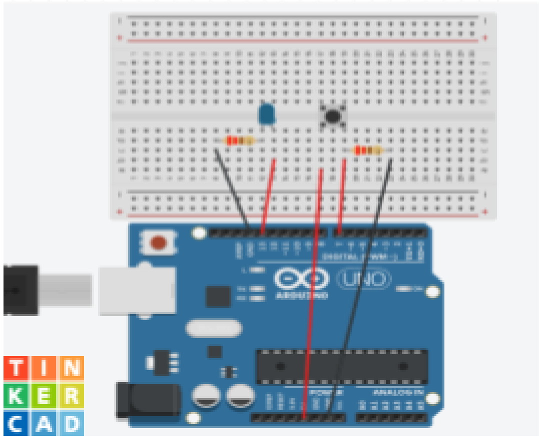
PRACTICE 2. II :Turn on a LED with a push button

Switch on a LED pushing a pushbutton

Components:
LED (D13)
pushbutton(D7)
Resistor 220Ω
Resistor 2,2kΩ

```
// Robots go green!
/*Practice 2.II: "Turn on a LED with a pushbutton "*/

const int LED=13;//define the led
const int BUTTON=7;//define the push button
int val;//define the variable val
void setup()
{
pinMode(LED, OUTPUT);//define the led as an output
pinMode(BUTTON, INPUT);//define the push button as an input
}
void loop()
{
val=digitalRead(BUTTON);
if  (val==HIGH)
{
digitalWrite(LED,HIGH);//the led will be on
}
else { digitalWrite(LED,LOW);//the led will be off
}
}
```





IV. CHALLENGE:

II.1 Traffic lights

With 3 LEDs (red, yellow, green) has to simulate the sequence of traffic lights. Don't forget 220Ω resistors to limit current throughout the LEDs

SOLUTION: CHALLENGE II.1

II.2 Traffic lights at a crossroad

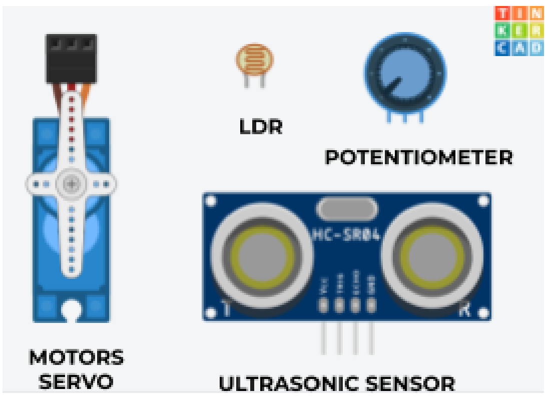
Here, Try to simulate the sequence of traffic lights at a crossroad. While a traffic light is red, the other is green

SOLUTION: CHALLENGE II.2

Co-funded by the Erasmus+ Programme of the European Union

PART 3:
THINGS ARE NOT BLACK AND WHITE: ANALOG INPUTS AND OUTPUTS

YOU'RE ABOUT TO LEARN...

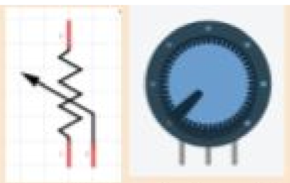


I. COMPONENTS

BOARD & COMPONENTS

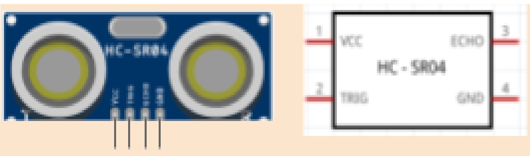
A. Potentiometer

A potentiometer is a device that provides a variable resistance, so we can read on the Arduino board as an analog value.
ht Has three terminals, the first and third one correspond to + 5V and GND respectively and the middle one is the cleaner.
This cleaner is connected and controlled through an analog input on the Arduino board. In this way, by turning the shaft of the potentiometer, we change the resistance. This would allow us, for example, to vary the intensity of the light from an LED.
ref, https://www.arduino.cc/en/Tutorial/Potentiometer


The **potentiometer** is used to regulate resistance and thereby control other devices

B. Ultrasonic sensor

Ultrasonic sensor measures distance, emitting an ultrasound wave which travels through the air. If there is an object on its pat, il will bound back to the module. Considering the travel time and the speed of the sound, you can calculate the distance.
It has 4 terminals: VCC(-5V), GND TRIGG(transmitter), ECHO (receiver)



Ultrasonic sensor uses sonar to determine distance to an object like bats do.

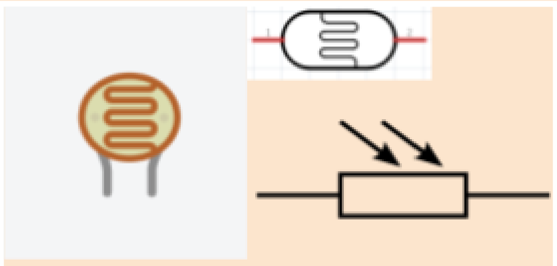
ref.
https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6

Co-funded by the Erasmus+ Programme of the European Union

C. LDR

An **LDR** (Light Decreasing Resistance is a component that decreases resistance with respect to receiving luminosity on the component's sensitive surface: This property is called photoconductivity and it can be applied in light-sensitive detector circuits.

ref :https://en.wikipedia.org/wiki/Photoresistor



LDR is a resistor that variess its value of electrical resistance depending on the amount of light that falls on its surface.

D. Servomotor

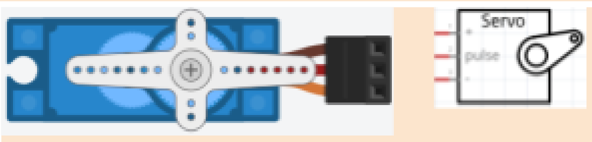
A **servomotor** is a suitable motor coupled to a sensor for position feedback that allows for precise control of angular or linear position, velocity and acceleration

The standard servo can rotate between 0-180º, but there are continuous rotation servos which can rotate between 0-360º.

Servo motors have three wires: power, ground, and signal.

- Power (red): connected to the 5V pin
- GND (black): connected to a ground pin
- Signal (yellow, orange or white): connected to a PWM digital pin (with a hyphen before i.e. pin -9).

ref.https://en.wikipedia.org/wiki/Servomotor



The standard **servomotor** can rotate between 0-180º

II. PROGRAMMING:

CODING

A. FUNCTIONS For controlling the Arduino board and performing computations

Analog I/O

AnalogRead(pin); Reads the value from a specified analog pin (A0-A5).
Note: Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit. (+inf)

AnalogWrite(pin, value); Reads the value from a specified analog pin (A0-A5).
Note: Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit. (+inf)

B. LIBRARIES: The Arduino environment can be extended through the use of libraries, just like most

programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from *Sketch > Import Library.*

#include <library.h>**:** is used to include outside libraries in your sketch. This gives the programmer access to a large group of standard C libraries (groups of pre-made functions), and also libraries written especially for Arduino. The function has to be included at the beginning of the program. (+inf)

C. COMMUNICATION:

Used for communication between the Arduino board and a computer or other devices. Arduino UNO Board has one serial port in Digital pins 0(RX) and 1(TX) that are used for communication with the computer.

***Serial*.begin(**speed**):** Sets the data rate in bits per second (baud) for serial data transmission. In Arduino UNO board this speed is 9600. This instruction must be written in setup. (+inf)

Serial*.print(**value**) & *Serial*.println(**value**):** Prints data to the serial port as human-readable ASCII text., moreover ***println writes the value in the following line. These instructions have to be written in loop(). The value is written as a number or “text” in the monitor serial. e.g. Serial.print(78) gives "*78*" and Serial.print("Hello world.") gives "*Hello world.*" in the monitor serial. (+inf)

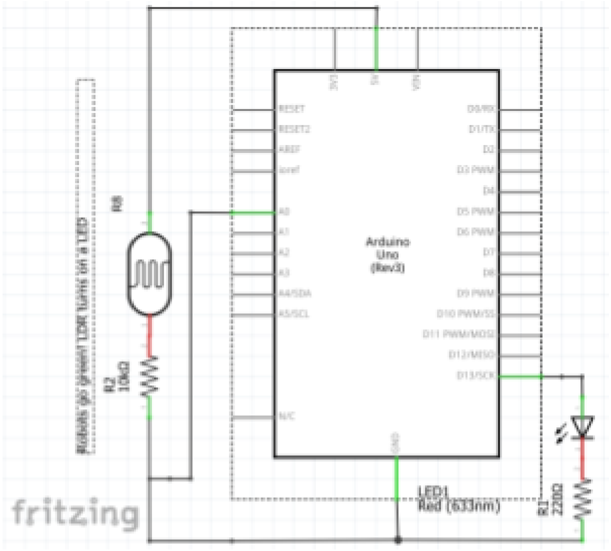
IV. PRACTICE:

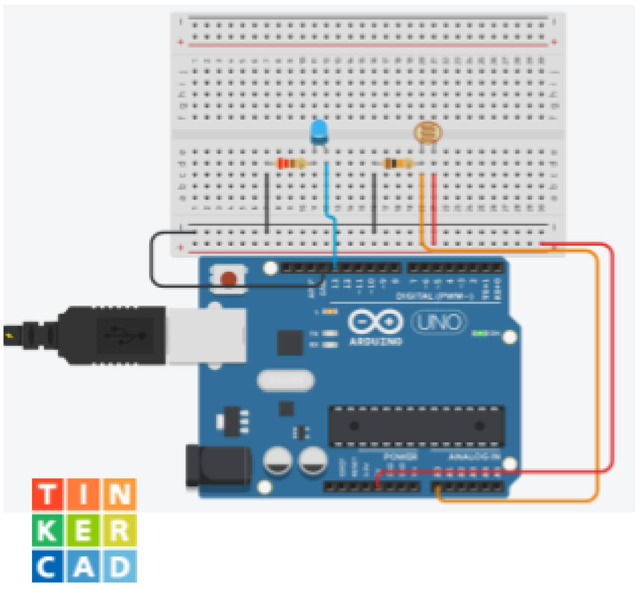
PRACTICES: SCHEMA & CODE

Practice 3.1: Turn on a LED with LDR

When the luminosity decreases until a level, LED turns on.
Components:
LED (D13)
LDR (A0)
1 Resistor 220Ω
1 Resistor 10KΩ

```
// Robots go green!
/*Practice3.1: "Turn on a LED with an LDR. "*/

void setup() {
  pinMode(13, OUTPUT);    //declare the led as an output
Serial.begin(9600);//initialise serial communication

}

void loop() {
  int d;
  d=analogRead(0);
 if (analogRead(0) > 150) { //value 0 to 150
    digitalWrite(13, HIGH); // LED turns on
 Serial.print("Darkness: ");
 Serial.print(d);//Print d in monitor serial
 Serial.println();
  }
  else { // if is higher
    digitalWrite(13, LOW);// led switch off
  }
}
```

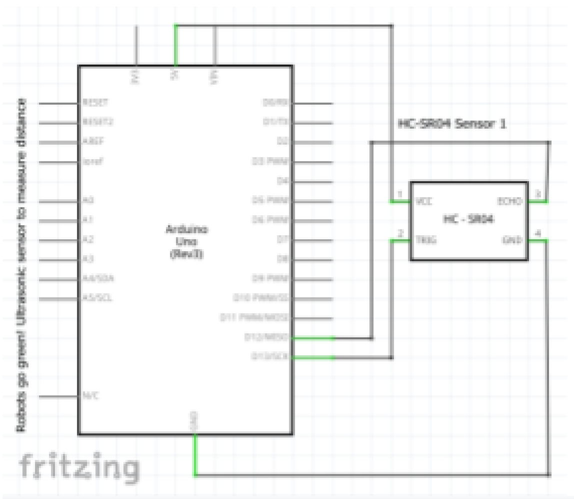


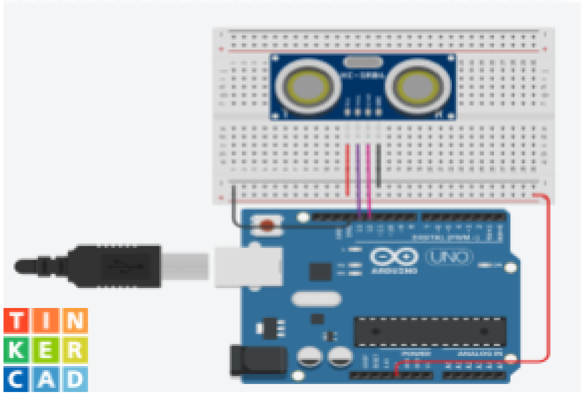


Practice 3.2: Ultrasonic sensor: measurement of distance

Measurement of distance with an ultrasonic sensor writing its value in the monitor serial.
<u>Components:</u>
Ultrasonic sensor H

```
// Robots go green!
/*Practice 3.II: "Measurement of distance"*/

const int Trigger = 13;   //Define Pin digital 13 for Trigger
const int Echo = 12;   //Define Pin digital 12 for Echo

void setup() {
  Serial.begin(9600);//initialise serial communication
  pinMode(Trigger, OUTPUT); //Trigger is an OUTPUT
  pinMode(Echo, INPUT);  //Echo is an INPUT
  digitalWrite(Trigger, LOW);//at the beginning Trigger is LOW
}

void loop()
{

  long t; //variable t t is the time for the Echo to arrive
  long d; //distance in cm

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);//Trigger sends a pulse for 10 us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //pulse width
  d = t/59;           //final formula (v=s/t v(sound)=340m/s d=go&return)

  Serial.print("Distance: ");
  Serial.print(d);      //Print d in monitor serial
  Serial.print("cm");
  Serial.println();
  delay(100);          //Pause  100ms
}
```





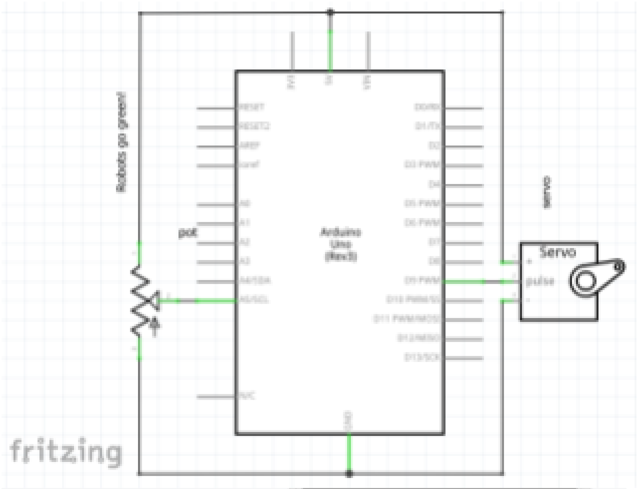
Practice 3.3: Moving a servomotor with a potentiometer

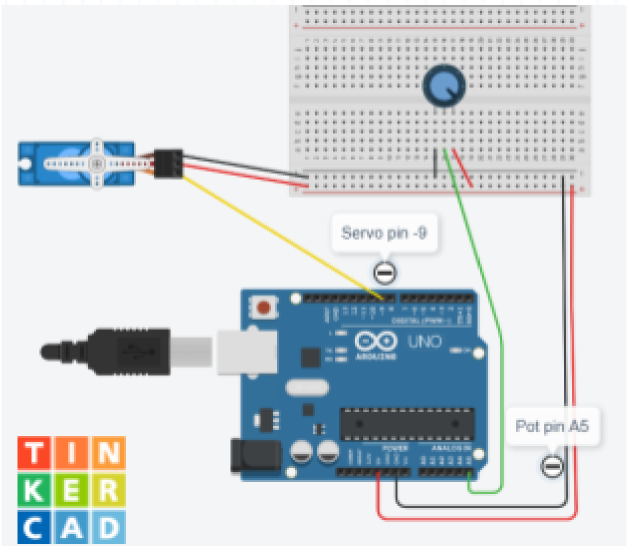
A servomotor is controlled by the movement of the potentiometer

<u>Components:</u>

servomotor

potentiometer





```
// Robots go green!
/*Practice 3.IC: "Moving a servomotor with a potentiometer"
 We are about to learn how Arduino can control the
movement of a servomotor throughout a potentiometer"*/

#include<Servo.h> // Include servo library
Servo myServo; // Define servo object
int const PotPin=A5; //Declare the potentiometer pin
int PotVal; //Declare the potentiometer value
int angle; //Angle value for the servo

void setup(){
myServo.attach(9);  //Declare the servo pin
Serial.begin(9600); //Initialize el monitor serial
}
void loop(){
PotVal=analogRead(PotPin);// Read the potentiometer value
Serial.print("PotVal:");  //Print Potval in monitor serial
Serial.println(PotVal); //Print the potentiometer value Potval
angle=map(PotVal,0,1023,0,179); //Translate PotVal in
Degrees
Serial.print("angle= "); //Print angle= in monitor serial
Serial.println (angle); //Print degrees in monitor serial
myServo.write(angle); //Put the degree value in the servo
delay (15); //Wait 15 ms
}
```

V. CHALLENGE:

III.1 Detector of presence

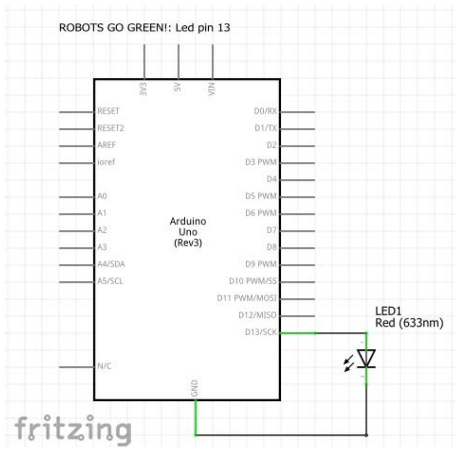
Turn on a LED when the ultrasonic sensor detected an object in a distance below to 60 cm.

SOLUTION: CHALLENGE III.1

CHALLENGE SOLUTIONS

Challenge I.1: “ Blinking faster”(sol.)

```
 // Robots go green!
  /*Challenge I.1: "Blinking faster"

void setup()   //
{
  pinMode(13, OUTPUT); // Declare pin 13 as an OUTPUT
}
void loop()
{
  digitalWrite(13, HIGH); // Write in the pin 13 a HIGH value...switch on
  delay(200);       // Wait for 200 millisecond(s)
  digitalWrite(13, LOW); // Write in the pin 13 a LOW value...switch off
  delay(500);       // Wait for 500 millisecond(s)
}
```

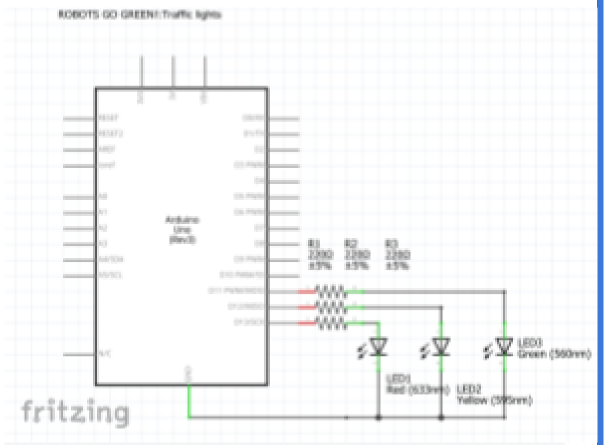


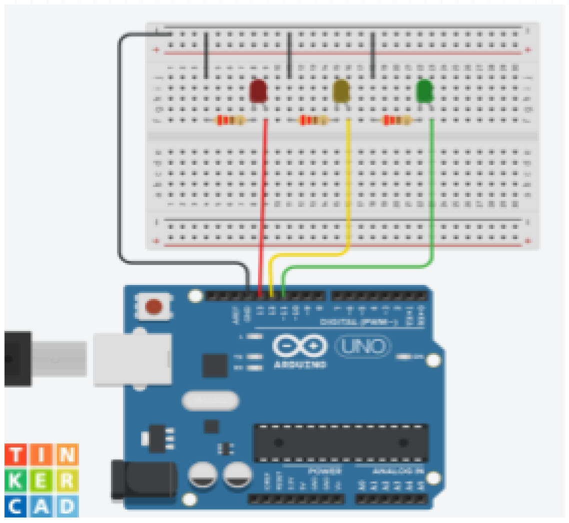
Challenge I.2: “ SOS in Code Morse”(sol.)

```
  // Robots go green!
  /*Challenge I.2: "SOS in Code Morse" In the language of Morse code, the letter “S” is three short dots and the letter “O” is
  three longer dashes. DOTS & DASH have been turned into different durations of the brightness of the board light "*/

int LED=13; //Declare variable LED in pin 13.
int DOT=400; //Declare variable duration DOT as 400 ms.
int DASH=1000; //Declare variable duration DASH as 1000 ms.
void setup(){
pinMode(LED,OUTPUT); //Variable LED is declared as an OUTPUT.
}
void loop()
{
for (int i=1; i <= 3; i++){ //DOT, DOT, DOT
digitalWrite(LED,HIGH);
delay(DOT);
digitalWrite(LED,LOW);
delay(DOT);
}
delay(DOT); // A delay between change DOT and DASH
for (int i=1; i <= 3; i++){ // DASH, DASH, DASH
digitalWrite(LED,HIGH);
delay(DASH);
digitalWrite(LED,LOW);
delay(DOT);
}
delay(DOT); //A delay between change DOT and DASH
for (int i=1; i <= 3; i++){ //DOT, DOT, DOT
digitalWrite(LED,HIGH);
delay(DOT);
digitalWrite(LED,LOW);
delay(DOT);
}
delay(2000); //A delay between SOS signals
}
```

Challenge II.1: “ Traffic lights”(sol.)

```
// Robots go green!
 /*Challenge II.1: "Traffic lights"

void setup() {
  pinMode(13, OUTPUT);//define the red led as an output
  pinMode(12, OUTPUT);//define the yellow led as an output
  pinMode(11, OUTPUT);//define the green led as an output
  digitalWrite(13, LOW);//the red starts off
  digitalWrite(12, LOW);//the yellow starts off
  digitalWrite(11, LOW);//the green starts off
}

void loop() {
  digitalWrite(13, LOW);//the red will be off
  digitalWrite(11, HIGH);//the green will light up
  delay(2000);//wait 2000 ms
  digitalWrite(11, LOW);//the green will be off
  digitalWrite(12, HIGH);//the yellow will light up
  delay(1500);//wait 1500 ms
  digitalWrite(12, LOW);//the yellow will be off
  digitalWrite(13, HIGH);//the red will light up
  delay(2000);//wait 2000 ms
  }
```

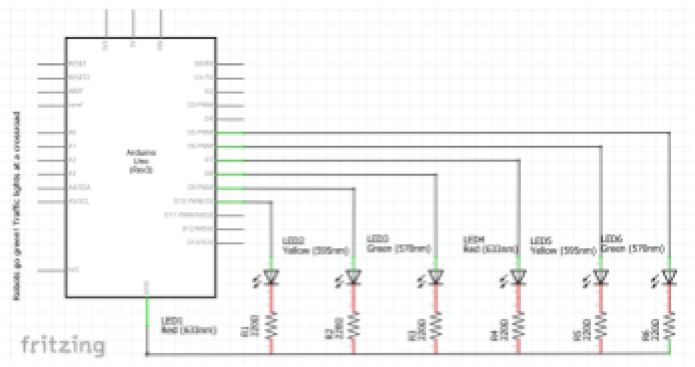


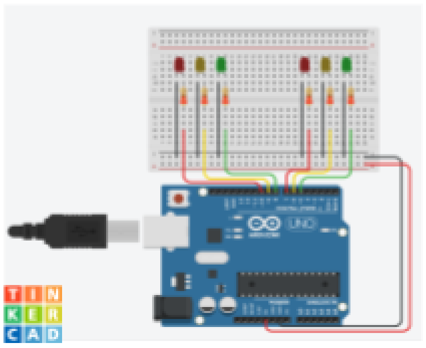


Co-funded by the
Erasmus+ Programme
of the European Union

Challenge II.2: “ Traffic lights at a crossroad”(sol.)

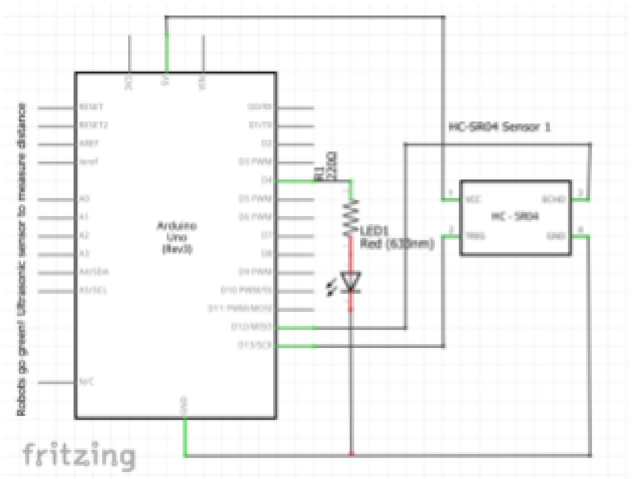
```
 // Robots go green!
  /*Challenge II.2: "Traffic lights at a crossroad"

int leds[] = {5,6,7,8,9,10}; //Green 1=8, Yellow 1=9, Red 1=10, Green 2=5, Yellow 2=6,Red 2=7
int i;

void setup() {
  for (i = 0; i < 8; i++)
  {
    pinMode(leds[i], OUTPUT);//define the leds as an output
  }
}

void loop() {
  digitalWrite (leds[10], HIGH);// to put on Red 1
  digitalWrite (leds[5], HIGH);// to put on Green 2
  delay(2000);//wait 2000 ms
  digitalWrite (leds[5], LOW);// to put off Green 2
   digitalWrite (leds[10], LOW);//to put off Red 1

  digitalWrite (leds[6], HIGH);// to put on Yellow 2
  digitalWrite (leds[7], HIGH);// to put on Red 1
  delay(1000);//wait 1000 ms
  digitalWrite (leds[6], LOW);// to put off Yellow 2
  digitalWrite (leds[10], LOW);// to put off Red 1

  digitalWrite (leds[8], HIGH);// to put on Green 1
  digitalWrite (leds[7], HIGH);// to put on Red 2
  delay(2000);//wait 2000 ms
  digitalWrite (leds[8], LOW);//to put off Green 1
  digitalWrite (leds[7], LOW);//to put off Red 2

  digitalWrite (leds[9], HIGH);// to put on Yellow 1
  digitalWrite (leds[7], HIGH);// to put on Red 2
  delay(1000);//wait 1000 ms
  digitalWrite (leds[9], LOW);// to put off Yellow 1
  digitalWrite(leds[7], LOW);// to put off Red 2
}
```

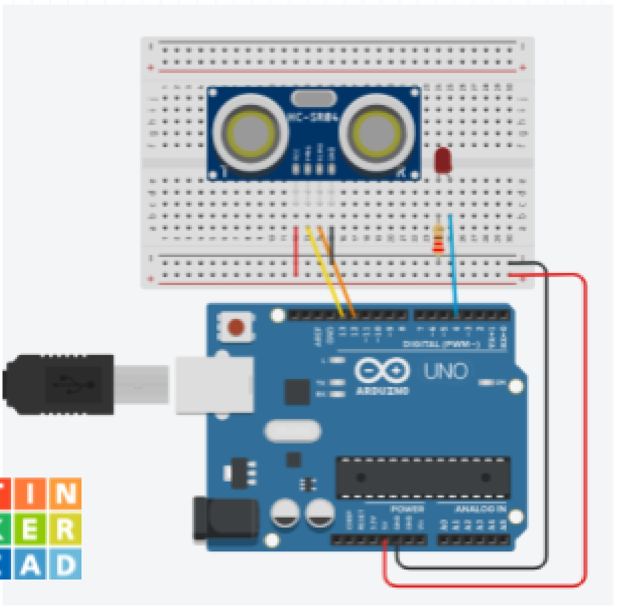




Challenge III.1: “ Sensor of presence”(sol.)

```
// Robots go green!
  /*Challenge III.1: "Sensor of presence"

const int Trigger = 13;   //Define Pin digital 13 for
Trigger
const int  Echo = 12;   //Define Pin digital 12 for Echo
int LED=4;// Define a LED in pin 4
void setup() {
  Serial.begin(9600);//initialise serial communication
  pinMode(Trigger, OUTPUT); //Trigger is an
OUTPUT
  pinMode(Echo, INPUT);  //Echo is an INPUT
  digitalWrite(Trigger, LOW);//at the beginning
Trigger is LOW
  digitalWrite(LED, LOW);
  }
void loop()
{
  long t; //variable t t is the time for the Echo to arrive
  long d; //distance in cm
  if (d<=60) {
    digitalWrite(LED,HIGH);// if distance is below 60 cm
switch on the LED
  }
  else {
    digitalWrite(LED,LOW); // if not switch it off
  }
  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);//Trigger sends a pulse for 10
us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //pulse width
  d = t/59;           // Transform time in distance  (v=s/t
v(sound)=340m/s.Sound go and back

  Serial.print("Distance: ");
  Serial.print(d);      //Print d in monitor serial
  Serial.print("cm");
  Serial.println();
  delay(100);         //Pause  100ms
}
```


Robots go green! Ultrasonic sensor to measure distance
Arduino Uno (Rev3)
R1 220Ω
LED1 Red (633nm)
HC-SR04 Sensor 1
HC - SR04
fritzing


HC-SR04
UNO
ARDUINO