Level crossing with barriers

1. Introduction.

In this Project, we are going to simulate a level crossing with barriers within a main road which will be activated when detecting the proximity of a train. We will be using robot Ozobot`s programming yo simulate the train and, on the other hand, the programming of Arduino to simulate the physical part of the barriers.

The main objective of this Project is to determine the Capability of Arduino as a tool for electronic control.

Arduino´s command implies a great degree of time reduction in electronic design and also in its microcontrolator`s programming.

Its characteristics make of it a great chance for business in terms of its implementation in technical fields like cars-making and and industrial communications. That will also create many job posts as far as these sectors are concerned.

2. Objectives.

This project´s objectives can be grouped as follows.

General objectives.

- Working on social abilities.
- Writing drafts and use adequate tools.
- Get familiar with Arduino
- Get familiar with Ozobot.

Specific objectives.

- Knowing how to make physical connections.
- Making circuit´s diagrams.
- Measuring distances through the rebound effect of sounds.
- Operate on sexagesimal system.
- Programming digital automats.

3. Components

We will be using;

- Arduino´s kit.
- Robots Ozobots
- Felt pens Ozobots
- BreadBoard PCB.
- 2 servo Motores
- 1 ultrasonic sensor
- 12 wire dupont

4. Knowing Ozobot.

Ozobot is a smart robot designed for STEM¨s language learning which is able to read its program on color lines drawn in a paper.

In our case, we will be using the code red-black-red

To slow down the speed and, on the other hand, we will be using the code blue-black-blue to accelerate and getting faster-

5. Knowing Arduino

In this project we are going to work with Arduino Uno board, more specifically, R3.

***Hardware*:**

The main features of this board are three parts that are clearly differentiated. The first part makes reference to the energy o power. The values of this energy will be 5v, 3.3v as well as its corresponding GND connection. The second part is composed of 13 digital pins, that, as its names implies, they simulate binary code (0 or 1). Finally, there are 5 analogue pins that work in a similar way as the digital ones, therefore, their assembly is similar. However , their voltage value can be between 0V and 5v.

Software:

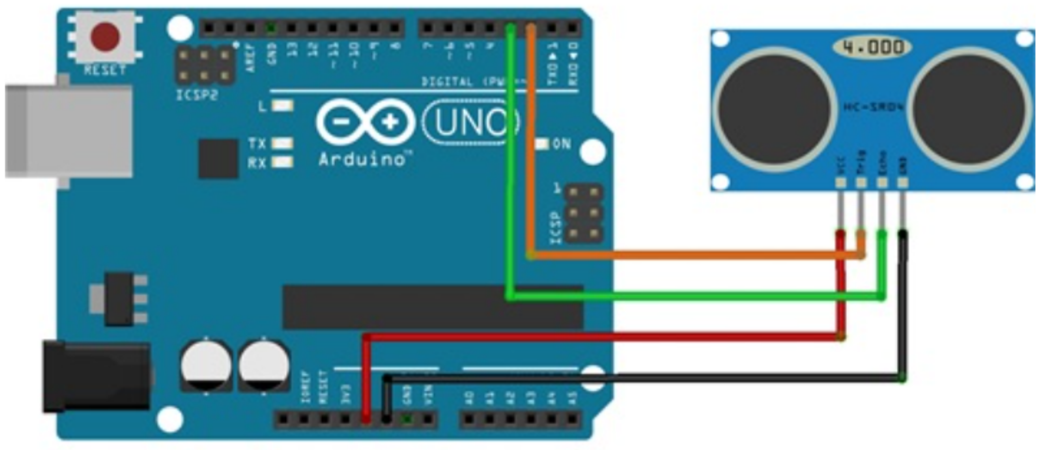
In order to configure Arduino, we will work from the programming environment or IDE. These are composed of two main functions: Setup() and loop().

The Setup function initializates and sets the inicial values and pins. The Loop function will be repeated constantly. This is where we will configure and programme whatever is needed.

6. Mechanics

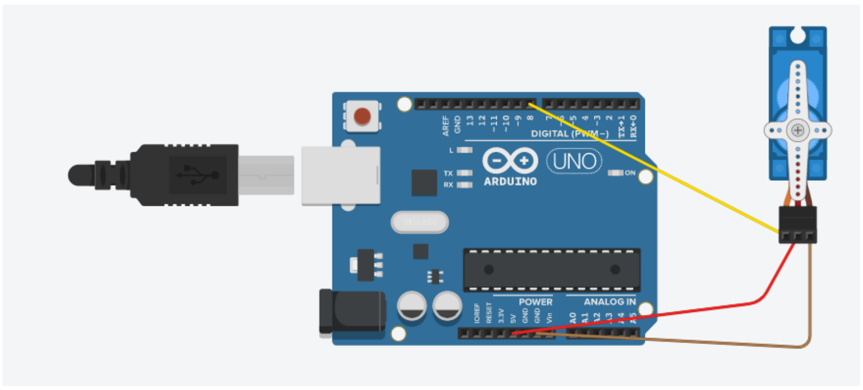
Creation of Grafcet as self-help to clarify the steps for the assembly of the project.

We will start our project with the connection of the ultrasonic sensor.



As we can see, we need 4 connections. The endings will be Power, specifically, the positive pole will connect to 5V and the negative pole will connect with GND. On the one hand, we need a Trigger pin to trigger the electrical impulse. On the other hand, an Echo pin, that will produce a pulse when the signal is received.

Next, we will explain the assembly of *servos*



As we can see, we will need three cables for the functioning of the servo. Two of them will be used for energy (one end 5V and the other GND), while the other cable will be connected to a digital pin that will allow programming the turning angle.

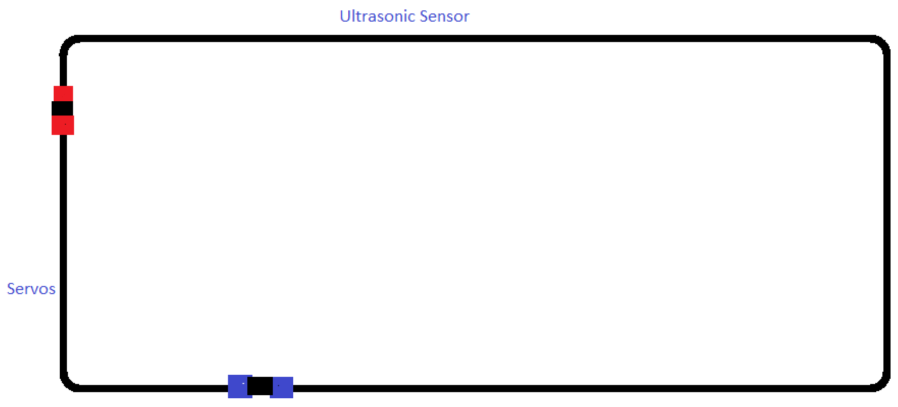
We would like to remind that a servo is basically a DC motor (direct current) with reduction gear that can only turn 180 degrees. It is controlled by electric pulses of Arduino. These pulses indicate the servo where to move.

Our project will mix both concepts to make the servos work from what we can detect from the ultrasonic sensor. The assembly drawing is as follows:

If we look at it as a component:

7. Logic Step: Programming

As commented in previous sessions, we are going to build a circuit for an Ozobot that will be controlled by Arduino sensors in order to simulate a level crossing with barriers. When the Ozobot passes by a certain area (where the ultrasonic sensor will be set), the servos will be activated to close the barriers. After, when the Ozobot approaches the level crossing, the speed will be reduced by using the colour codes mentioned before. Once the Ozobot crosses the barriers, they will open back again and the Ozobot will accelerate.



A positive pole and GND are needed for the programming of the ultrasonic sensor. The Trigger pin will trigger the electric pulses and Echo pin will receive the pulse.

First, we configure the pines and the serial communication up to 9800 bauds in order to see, when wanted, the needed values (in our case, distance).

```
const int Trigger = 2;   //Pin digital 2 for the Trigger
const int Echo = 3;   //Pin digital 3 for the echo

void setup() {
  Serial.begin(9600);//start comunicate
  pinMode(Trigger, OUTPUT); //pin output
  pinMode(Echo, INPUT);  //pin as input
  digitalWrite(Trigger, LOW);//start the pin at 0
}
```

As we can see, the Trigger will trigger the pulses and the Echo will receive the pulse. It must be taken into consideration to start the project with a turned off Trigger to avoid the pulse sending.

Now we have to programme the Loop() part. Our objective is to send the pulses from the Trigger pin so the Echo pin can receive them in order to calculate the time that it takes to bounce.

We are going to send a pulse of 10 microseconds to the Trigger of the sensor

```
digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //send un pulso de 10us
  digitalWrite(Trigger, LOW);
```

After, we register the pulse of the Echo using the function pulseIN (pin, value).

```
  t = pulseIn(Echo, HIGH); //obtain the pulse
```

The pulse of 10 microseconds is send, as it is empirically demonstrated that it works from that amount of time. This way we store the time that the ultrasonic echo takes to arrive. The later, the further.

To calculate the distance we use the formula Speed= distance / time

We know the speed as it is the same as the sound 340m/s, we are interested in measuring small units, therefore we need to work with cm and as the pulse is sent in µs, we will use that unit of time.

s= d / t As we already know, giving that it is the distance of bouncing; we consider a double distance (back and forth) for a specific time.

The formula will be 340ms= 2 d / t therefore, if we solve the distance (that will be the value that we desire to calculate with the sensor) the result will be d= 340ms·t/2

It is important to take into account that the time is expressed in µs and that it is required that the distance is expressed in cm (given that we will measure short distance).

$$170\,\frac{m}{s}\cdot\frac{100\,cm}{1m}\cdot\frac{1s}{10^6\mu s}=\frac{17}{1000}\,\frac{cm}{\mu s}=\frac{1}{59}\,\frac{cm}{\mu s}$$

After having done the conversion factor the distance expressed in centimetres, depending on the time value, expressed in microseconds, will be:

d=1/59·t cm

Finally, we will send the value of the distance to the serial port (to print) and we will make a pause of 100 ms that will exceed the 60 ms recommended between measurements by ultrasonic manufacturers.

The arduino code will be:

```
const int Trigger = 2;
const int Echo = 3;

void setup() {
  Serial.begin(9600);
  pinMode(Trigger, OUTPUT);
  pinMode(Echo, INPUT);
  digitalWrite(Trigger, LOW);
}

void loop()
{

  long t;
  long d;

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH);
  d = t/59;

  Serial.print("Distancia: ");
  Serial.print(d);
  Serial.print("cm");
  Serial.println();
  delay(100);
}
```

After that, distance is controlled. Afterwards, we must programme the servos, which will be activated as soon as a shorter distance than the one stipulated, is detected through the ultrasonic.

Once the ultrasonic sensor is programmed, we programme the servo.

As we have already explained, the connection pines will be connected to 5V and GND respectively, while the other pin will be connected to a digital one. It must be taken into consideration that for the servo to work, a library servo.h. must be included.

#include <Servo.h>

Servo servo1;

In the Setup() function we will set servo1.attach("port connected");

To activate the servo. When we want to modify the parameters for the servo to turn we have to use servo1.write(º*Degrees*). Remember that it can only turn around from 0º to 180º.

Our project will control two servos when the distance in the ultrasonic sensor is less than the one considered for the size of the model.

The final code will be:

It must be highlighted that we use the port 7 as a power source for the ultrasonic sensor. If all the elements are connected to the same source we will lose difference of power

```
#include <Servo.h>

Servo servo1;
Servo servo2;
int power = 7;
const int Trigger = 2;
const int Echo = 3;

void setup() {
  Serial.begin(9600);
  pinMode(Trigger, OUTPUT);
  pinMode(Echo, INPUT);
digitalWrite(Trigger, LOW);
servo1.attach(10);
 servo1.write(90);
 servo2.attach(11);
 servo2.write(90);
 pinMode(power, OUTPUT);
 digitalWrite(power, HIGH);

}

void loop()
{

  long t;
  long d;

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH);
  d = t/59;

  Serial.print("Distancia: ");
  Serial.print(d);
  Serial.print("cm");
  Serial.println();
delay(100);
  if(d <= 10){
    for(int i = 90; i >= 0 ; i--){
      servo1.write(i);
      servo2.write(i);
      delay(10);
    }

    delay(8000);
    servo1.write(90);
    servo2.write(90);
  }
}
```