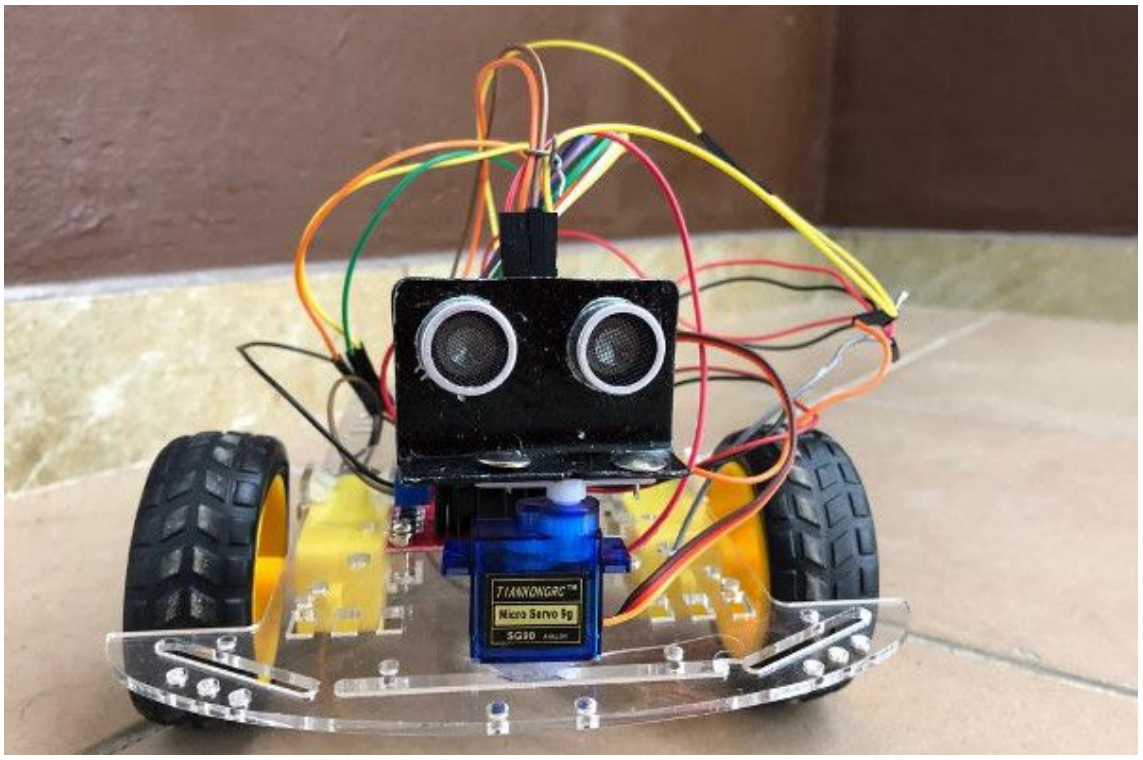
Institut Puig Castellar
Santa Coloma de Gramenet

[1]



Arduino car

CFGS Administració de Sistemes Informàtics i Xarxes

Juan Miguel Segura Fernandez
Alejandro Mallén Gomez

2r SMX

[1] Cover image.

B) GNU Free Documentation License (GNU FDL)

C) Copyright

Abstract (in English, 250 words or less):

In this project we are going to build an Arduino car that will be able to be autonomous. We're going to add a sensor with a servo, which is going to be turning around from 0º to 180º.So It'll figure out how near or far are the obstacles from the car. It'll dodge all the interferences that might be in his way, hence the car will be allowed to be in his own without getting hit by any obstacle. As a consequence the car will be a autonomous.

Keywords(entre 4 i 8):

- Arduino
- Autonomous
- Car
- Obstacles

Index

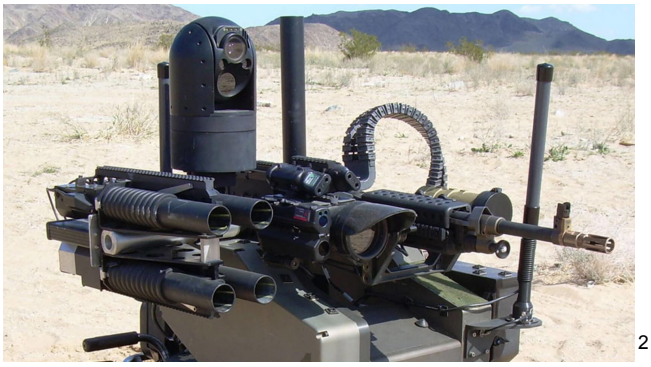
1. Introduction

1.1 Context

- We build an arduino car from the scratch. This is not relevant, but we decided to start this project because we like autonomous robots and we've inspired in others autonomous robots, like the famous vacuum cleaner or militar ones.

- We want our car to dodge any kind of obstacle, changing his path and keep going.

Example of a military robot:

 [2]

1.2 Goals

1. Work like a team to create an Arduino car.
2. Build a car (Hardware)
3. Programme an Arduino board (Software)
4. Initiate us into the programmation world (Arduino IDE)
5. Build an autonomous Arduino car. (Main goal)

[2] Military robot.

Explanation:

- Our main goal is build up a car in our own from scratch, thus we'll get into the hardware assembly. The hardware will be controlled by the arduino motherboard, which will be computed by our code. Being the first time we compute. We want the car to be a autonomous one that will dodge any obstacle on it's way. The car won't follow any path, he will choose a random path, that way, it'll be autonomous. The car will be similar to the famous vacuum cleaner known as "Roomba".

[3]

1.3 Working method

- Our working method consist in building a car from scratch. Thus we can customize our car, either physically or the arduino code. So we can use different materials.

- We didn't want to buy a car and personalize it. Cause we believed that it would be so much easy if we did this. Our mainly job would've been computing the code of the Arduino.

[3] Vacuum cleaner (Roomba).

1.4 Project plan

See diagram

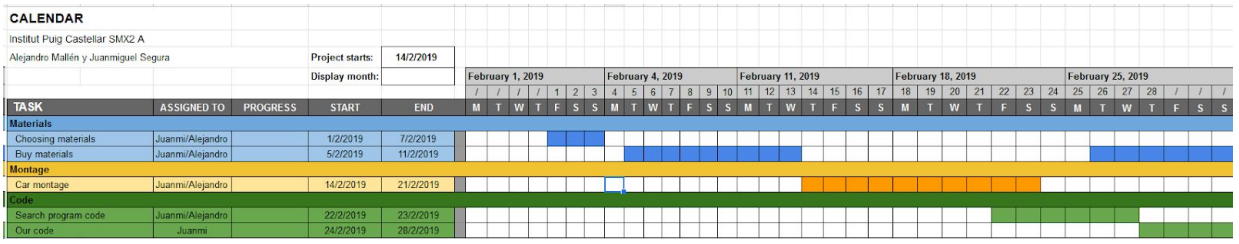
Tasks:

- Materials
  - Choosing materials
  - Buy materials
- Mounting
  - Car montage
- Code
  - Search program code
  - Our code
- Document
  - Doing the TR Document
- Web page
  - Creation of the website

Gantt diagram (*In red the delivery date*)

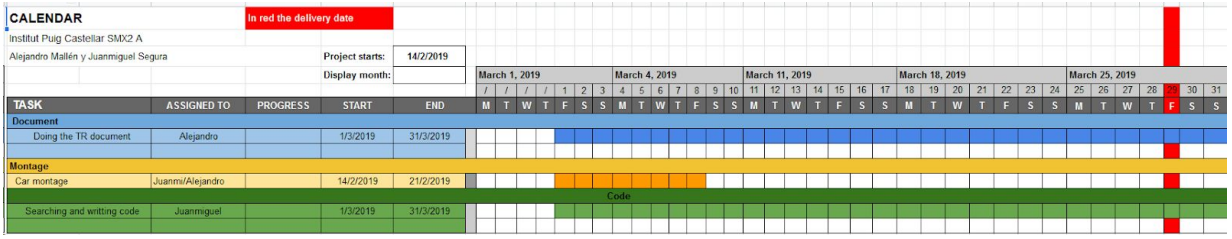
- **February**

4



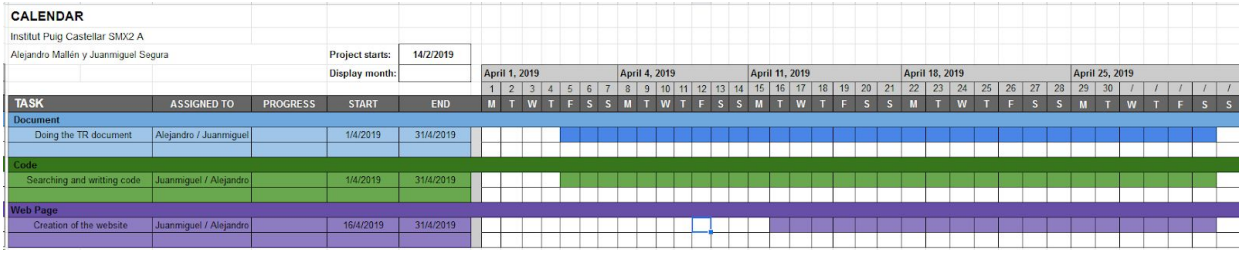
- **March**

5



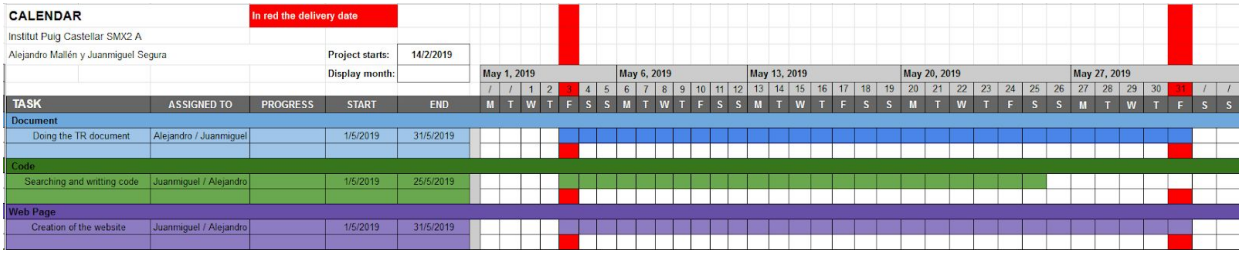
- **April**

6



- **May**

7



[4] February gantt diagram.
[5] March gantt diagram
[6] April gantt diagram
[7] May gantt diagram

1.5 Products obtained

- The products we need are the following ones:
  - Arduino Uno Rev3
  - Smart Robot Car Chassis
  - Motors controller(Neuftech L298N)
  - Ultrasound sensor.
  - Wires from male to female.
  - Wire from female to female.
  - Wires from male to male.
  - One servo
  - Printer 3D

1.6 Memory description

Materials:

- In this section we'll explain what kind of materials we'll need in the future so we can carry out our project. We have to look out for others similar Arduino projects on the internet, so we can find out what materials do we need and the exactly arduino model.

- We'll also explain where did we get all the materials, that is to say, in which web pages we did buy it and why we did choose this web page.

Assembly:

- In this section we'll explain how we did build our car. For it we had to check other Arduino projects and assembly diagrams, so we know how to connect all the wires at the mother board properly. And where to place all the components in the chassis.

Computing:

- In this chapter we'll explain how we did compute the code and why we did that. Cause we compute it so our car can do what we want him to do. Later on we'll explain the code (sentences, variables...)

Website:

- In this section we'll show our web page, where we will upload the documentation and other stuff related with the project. Thus it would be easier to get to our documentation.

2. Materials

2.1 Searching

- We've been looking for for the required materials in differents blogs and web pages, you can check at the annex. We'll buy differents materials depending on what we want the car to do.

- We'll need for the car: a chassis, which it includes motors and tires, the arduino motherboard, the motors controller which will keep an eye of the speed and the wheels, a ultrasound sensor and last but not least the servo and thanks to this one the car will be able to locate all the obstacles. So the car will dodge them.

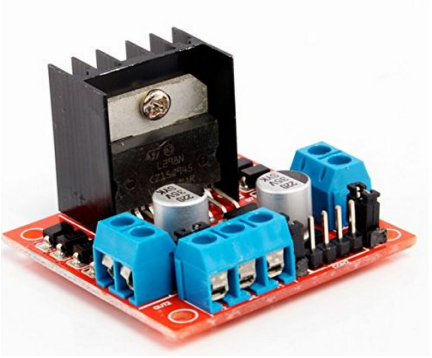
2.2 Budget

Materials and purchase links:

Arduino Uno Rev3 [8]

Smart Robot Car Chassis [9]

Motor controller (Neuftech L298N) [10]



Ultrasound sensor [11]

Wires [12]

Servo [13]

[8] Arduino motherboard
[9] Car chassis
[10] Motor controller
[11] Ultrasound sensor
[12] Wires
[13] Servo

Index

2.3 Description and use

Arduino Uno Rev 3:

- This is the exactly arduino model that we need to develop our project. We've been consulting differents blogs and web pages looking for other arduino projects and we reached the conclusion that we need this one. We decided to buy the original model, it's more expensive than other chinese models that are cheaper. But there aren't troubles with drivers which the chinese one has. We did buy it at the arduino official page.

Smart Robot Car Chassis

- This is the chassis that we've chosen to build up our arduino car. As we said, we've been looking for others arduino projects and we decided the buy the 3 car tires. By doing so the car will need less energy to work.

- The chassis we decided to buy includes the two necessaries motors, the three tires, bolts and the necessary tools so we can assemble it.

Motors controller

- The motor controller is able to handle the two motors. We've chosen the L298N model which is the best to build up little arduino models. The motor controller is essential to develop our car.

Ultrasound sensor

- The ultrasound sensor is implemented in our arduino car, which will allow him to finger the obstacles on it's way and thanks to that the car will be able to dodge all the obstacles. The ultrasound sensor model is the HC-SR04, and is the most common sensor used to this kind of projects. The sensor will be spinning thanks to a servo, which will spin 180º, as we said, thanks to that the car will be able to identify the obstacles.

Servo

- As we said, we we'll use one servo. In that way, the sensor can scan thanks to the servo that will be spinning 180º.

Wires

- We'll need different kind of wires to build our Arduino car, like: from female to female / from male to male and from female to male. All of these wires will allow us to connect all our materials, as the ultrasound sensor, the servo, the motor controller and the Arduino motherboard.

3. Assembly

3.1 Chassis and motors

- We prepare the motors, chassis and the bolts as you see in the next photo.

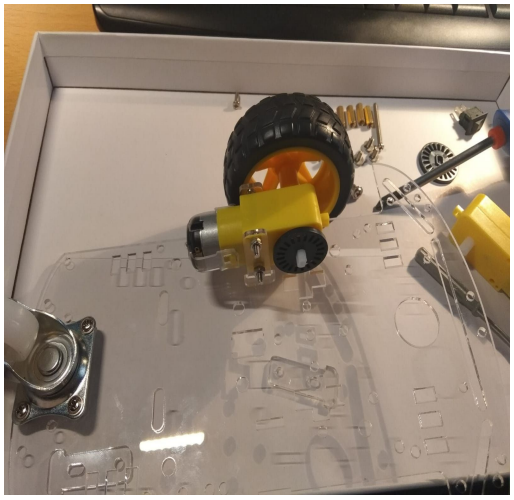
[14]

- Once we get all the materials we get ready to start the assembly.

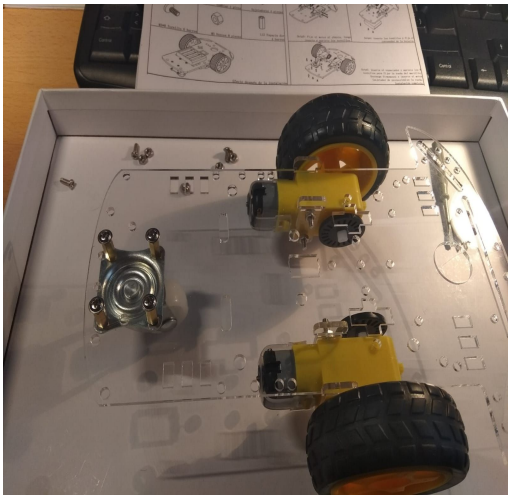
- First, we attach the motors to the chassis and we screw up the bolts. The chassis has holes so you can screw the bolts there. When we have one of the two motors correctly attached to the chassis we do the same with the other motor at the other side.

- When we are done with the motors we fit in the wheels to the motors.

[15]



[16]

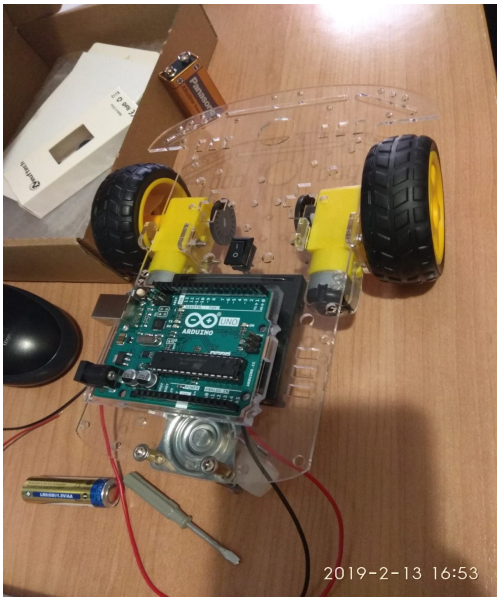


[14] Preparing all the materials to assemble the chassis.
[15] Attaching the left motor.
[16] Attaching the right motor.

3.2 Arduino

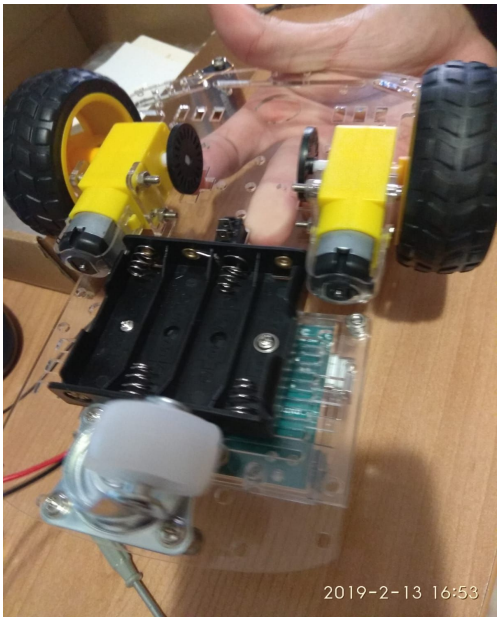
- In the second place we have to put the Arduino motherboard in the chassis without screwing it up.

[17]

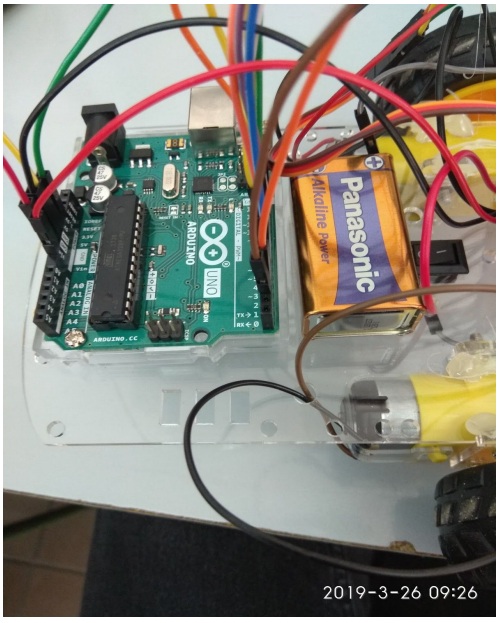


- Now, we put the battery in the chassis too.
- At the start we decided to put a set of batteries, 4 batteries of 2V each one. But it didn't provide the required energy to the car. So we changed it for one battery of 9V and a higher amperage.

[18]



[19]



[17] Putting the arduino motherboard in the chassis.
[18] Putting the set of batteries underneath the chassis.
[19] Changing the set of batteries for one bigger battery.

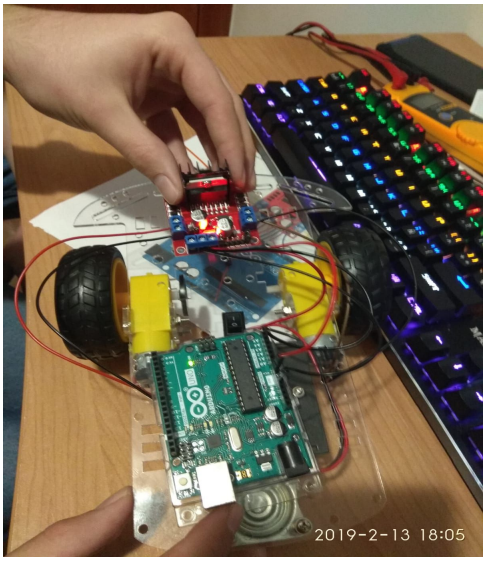
- Later, we had to put two 9V batteries in series so that the voltage doubled, as there was not enough power in a single battery to make all the car's components work.

3.3 Motor controller

- In the third place we put the motor controller in the chassis. And we put it at the front of the chassis, where there are many holes to a anchor it.

- When we have the motor controller ready we attach it with bolts.

20

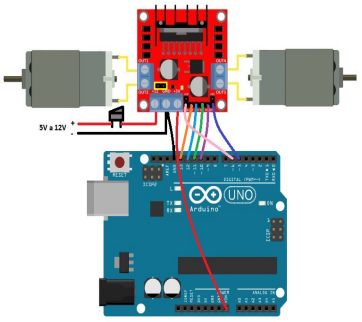


- As you can see at the previous photo we've just connected the wires.
- We'll show you up a diagram that we used of the wire connections,

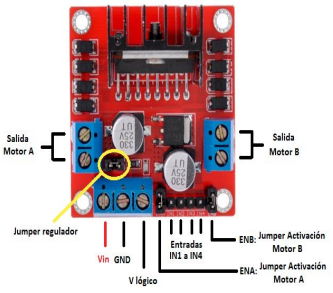
Wire diagram

- In this diagram we can see how to connect the wires from the motherboard to the motor controller and motors correctly.

21



22

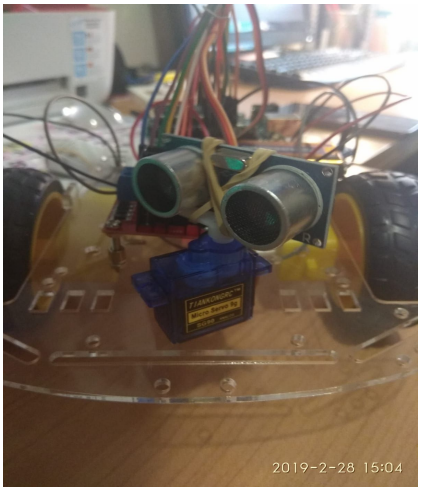


[20] Attaching the motor controller to the front of the chassis.
[21] Wire diagram 1.
[22] Wire diagram 2.

3.4 Servo and ultrasound sensor

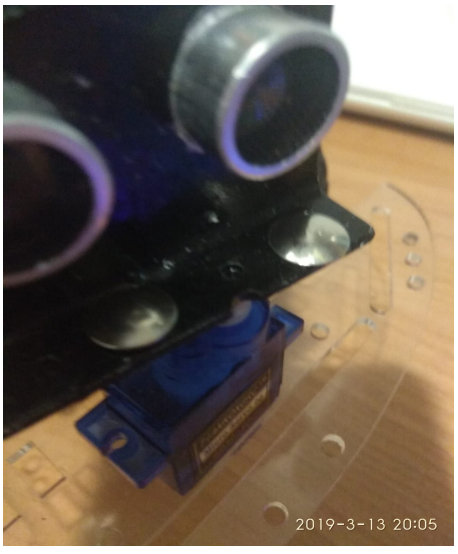
- In the fourth place, we have to put the servo and the ultrasound sensor, these are the last materials that we have left. We put them in the front of the chassis after the motor controller as you can see at the next photo.

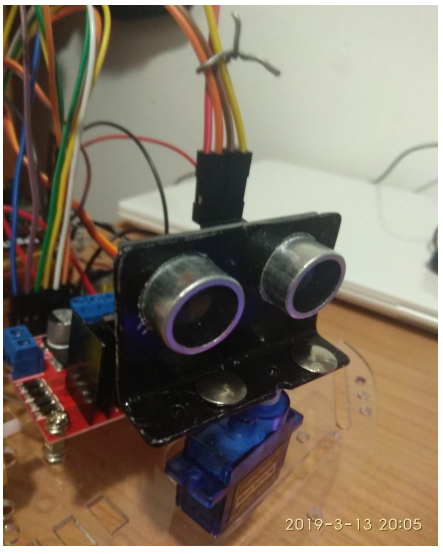
- We need the ultrasound sensor to be above of the servo. So we attach it to the servo with a rubber band, for now.
[23]



- We knew that we couldn't attach the sensor to the servo with a rubber band so we designed a chassis for the sensor. So it could hang in there without falling.
- If we want to build something like that we have to measure the sizes of the sensor. So we cut the sheet correctly. The sensor has two kind of eyes that are protunding so we have to make two holes on the sheet too.
[24]

[25]



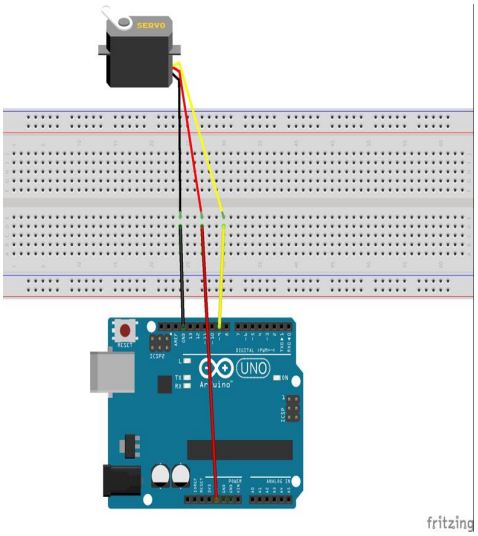


[23] First ultrasensor carcass.
[24] Second ultrasensor carcass 1.
[25] Second ultrasensor carcass 2.

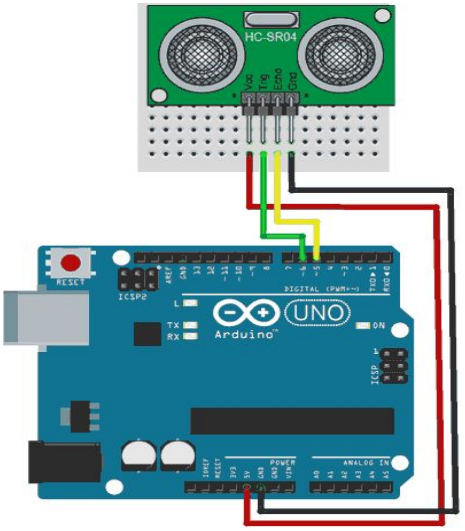
- At the fifth place we connect the wires from the servo and the sensor to the motherboard.

- At first we have some troubles with the connection of the wires. The diagram sais that we have to connect the sensor and the servo to the 5V socket of the motherboard, but there is no need to do that. We plugged in the servo at the 3.3V socket instead of the 5V one and the sensor to the 5V one.

Servo [26]



Sensor ultrasonido [27]



- Finally, when we've decided where to put all the components at the chassis we attach them to it.

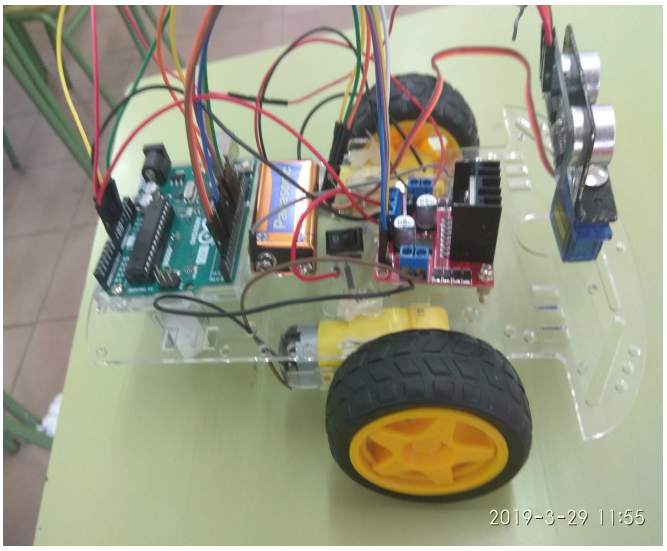
[26] Servo wire diagram.
[27] Sensor wire diagram.

3.5 Distribution

The distribution is the following one, from the back to the front:

- The Arduino motherboard is at the back.
- Following the motherboard we have the battery.
- After the battery we have a switch.
- At the front of the chassis there is the motor controller.
- In the sharp end of the chassis is attached the servo with the ultrasound sensor above.

[28]



3.6 Fastenings

- The Arduino motherboard is fastened for his own carcass. We attach it screwing up the holes from the carcass and holes of the chassis.
- The battery is attached with silicone.
- The motor controller can hold there thanks to the bolts we did screw.
- The servo is fastened for the velcro we put at the front of the chassis.
- The ultrasound sensor, as we explained, is attach to the servo thanks to the metallic sheet we did.

[28] Distribution.

4. Computing

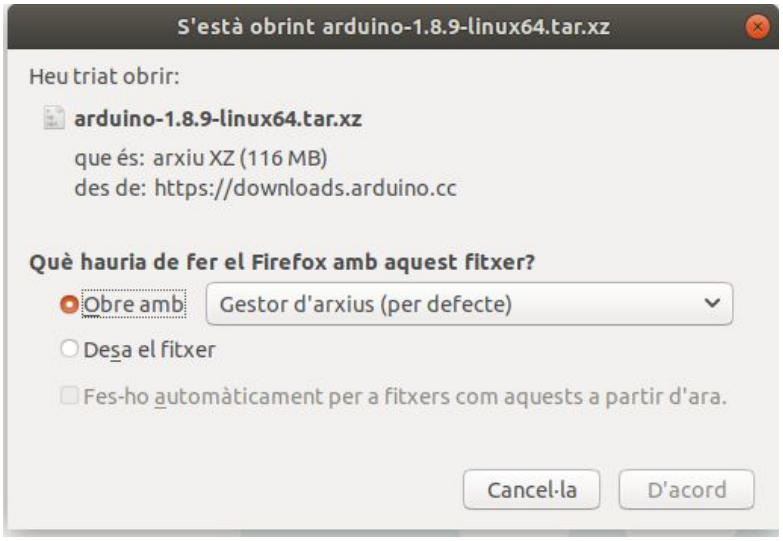
4.1 Arduino IDE installation

- We need the Arduino software called Arduino IDE, so we can program the car.

- We can download Arduino IDE from the official Arduino web page. We have to download and install it. For the installation:
[29]

- We chose Linux 64 bits or 32 bits, it depends on your computer.

- We save the .tar.xz.
[30]



[29] Arduino IDE.
[30] Arduini.tar.xz insllation.

- We decompress it:

```
smx2a@torvalds-113:~/BaixadesDime:  tar -Jxvf arduino-1.8.9-linux64.tar.xz
```

[31]

- Now we have to move the the following directory writing in the terminal the following line:
    - cd arduino-1.8.9/

- We execute it with the command sh install.sh.
- If the installation it's going well, the terminal will print the following lines:

[32]

```
smx2a@torvalds-113:~/Baixades/arduino-1.8.9Dime: sh install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE...
rm: no s'ha pogut eliminar '/usr/local/bin/arduino': El fitxer o directori no existeix
Removing symlink failed. Hope that's OK. If not then rerun as root with sudo.

rm: no s'ha pogut eliminar '/usr/local/bin/arduino': El fitxer o directori no existeix
Removing symlink failed. Hope that's OK. If not then rerun as root with sudo.

ln: no s'ha pogut crear l'enllaç simbòlic '/usr/local/bin/arduino': S'ha denegat el permís
Adding symlink failed. Hope that's OK. If not then rerun as root with sudo.
 done!
smx2a@torvalds-113:~/Baixades/arduino-1.8.9Dime:
```

Finally, we have the Arduino IDE installed.

[33]

[31] Arduino directory.
[32] Arduino IDE installation.
[33] Arduino IDE.

4.2 Code

4.2.1 First code

We found the structure of the code on the Internet, and we adapted it.
In this chapter we are going to explain the code, even it has comments that explains it itself.

- The code:

```
/* Juan Miguel Segura y Alejandro Mallen
Trabajo de sintesi 2n SMX
Licencia GPL
*/

// Incluimos la libreria para controlar el servo y el sensor de ultrasonido
#include <Servo.h>
#include <NewPing.h>


// Aqui se configuran los pines donde debemos conectar el sensor
#define TRIGGER_PIN  2
#define ECHO_PIN     3
#define MAX_DISTANCE 200

// Variables del motor A
int ENA = 6;
int IN1 = 13;
int IN2 = 12;

// Variables del motor B
int ENB = 5;
int IN3 = 11;
int IN4 = 10;

// Variable del servo
Servo servoMotor;

// Indicamos la variable de la velocidad
int vel = 90;
```

[34]

The beginning of the code are just comments.

- As you can see, we added the required libraries for the correct running of the servo and the ultrasound sensor.
- We have to define the pins that are connected to the Arduino motherboard. We define the define the pins connected to the A motor and the B motor.
- The variable of the servo is *servoMotor*.
- We adjust the speed of the car with the variable ***vel,*** thus we can change the speed of the car whenever we want.

[34] Arduino code 1.

```
// Funcion de setup
void setup() {
Serial.begin(9600);
 // Declaramos todos los pines como salidas
 pinMode (ENA, OUTPUT);
 pinMode (ENB, OUTPUT);
 pinMode (IN1, OUTPUT);
 pinMode (IN2, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (IN4, OUTPUT);
 servoMotor.attach(9); // Declaramos el servo para que trabaje con el pin 9
}
```

[35]

- That's the variable where we define the pins and the servo are outputs that we've already declared.

- With the *Adelante* variable we can make the car go forward. With the *Atras* variable we make the car move backguards. With the *Izquierda* and *Derecha* variable we move the car left or right. With the *Parar* variable we make the car stop.

[36]

```
void Derecha (int time)
{
 //Direccion motor A
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, vel); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, vel); //Velocidad motor A

  delay(time);

}

// Funcion para parar el coche
void Parar(int time){
  //Direccion motor A
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, LOW);
  analogWrite (ENA, vel); //Velocidad motor A
  //Direccion motor B
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, LOW);
  analogWrite (ENB, vel); //Velocidad motor B
  delay(time);
}
```

[35] Arduino code 1.
[36] Arduino code 1.

[37]

```
void Izquierda (int time)
{
 //Direccion motor A
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, vel); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, vel); //Velocidad motor B

 delay(time);
}
```

[38]

```
 void Adelante (int time)
 {
  //Direccion motor A
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, HIGH);
  analogWrite (ENA, vel); //Velocidad motor A
  //Direccion motor B
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
  analogWrite (ENB, 97); //Velocidad motor B /

  delay(time);
 }

 void Atras (int time)
 {
  //Direccion motor A
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  analogWrite (ENA, vel); //Velocidad motor A
  //Direccion motor B
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  analogWrite (ENB, vel); //Velocidad motor B

  delay(time);
 }
```

- As you can notice at the speed variables if the car moves forward the two motors (IN2 and IN4) accelerate forward. But if the car has to move backwards the motors (IN1 and IN3) accelerate to the opposite direction.

[37] Arduino code 1.
[38] Arduino code 1.

- If the car moves to the right the motor A move backwards with the IN2 variable and the A motor moves forwards with the IN3 variable. In that way the car will turn right. But if the car has to move left it does the opposite.

- If the car stops it's pretty simple, the motors just stop working and it doesn't move.

Observation

- As you can see, we add to the functions the variable called time. That variable defines the time required to exec the function when they are called in the code.

4.2.2 Second code

- In the second code we've just add the servo function called move_servo.
- With the move_servo function we make the servo always move from left to right and when it gets to the right it moves to the left.

39

```
// Funcion para mover el servo
void move_servo() {
// Vamos a tener dos bucles uno para mover en sentido positivo y otro en sentido negativo
/*
180° = esta en la izquierda
90° = en medio
0° = esta en la derecha
 */

// Para el sentido positivo
  for (int i = 0; i <= 180; i+=5) { // i = i +5
    // Desplazamos el servo al angulo de la variable i, cada vez se ira incrementando.
    servoMotor.write(i);
    // Hacemos una pausa de 25ms por cada cambio de angulo
    delay(1);
    }

  // Para el sentido negativo
  for (int i = 180; i > 0; i-=5) { // i = i - 5
    servoMotor.write(i);
    delay(1);
  }

}
```

[39] Arduino code 2.

- As we can see in the code, in the servo function, we have added two for loops, one in a positive direction, which is fulfilled when the variable *i* is smaller than 180 and, it is moving the servo to the degree of the number of the variable *I*, and another loop in negative direction that does the same but it is always fulfilled that the variable i is greater than 0.

- In our servo, the 0 degree is found when the servo is completely turned to the right, and the degree 180 is found when it is turned completely to the left. Because our servo works by degrees, in our code the *i* variable is a variable that assigns the degree to which the servo has to move, so, we make it increase 5 degrees each time it rotates. So we get the servo to go from right to left increasing 5 consecutively.

4.2.3 Third code

- We add the move_servo function.
- The move_servo function makes the car spin 0º, 90º and 180º.

[40]

```
/ Para el sentido positivo
 for (int i = 0; i <= 180; i+=5) { // i = i +5
   // Desplazamos el servo al angulo de la variable i, cada vez se ira incrementando.
   servoMotor.write(i);
   // Hacemos una pausa de 25ms por cada cambio de angulo
   delay(1);
   // Cuando el servo llega a 90°, llama al sensor y escanea
   if ( i == 90 ) {
     sensor_ultrasonido(i);
     }
   }
```

- We add a new sentence so when the servo is at 90º (the middle), it calls the servo ultrasound sensor if there are obstacles.

- When the loop for ends in a positive way, the servo will be at 180º, then it will call the ultrasound sensor function.

[40] Arduino code 3.

[41]

```
for (int i = 0; i <= 180; i+=5) { // i = i +5
  // Desplazamos el servo al angulo de la variable i, cada vez se ira incrementando
  servoMotor.write(i);
  // Hacemos una pausa de 25ms por cada cambio de angulo
  delay(1);
  // Cuando el servo llega a 90°, llama al sensor y escanea
  if ( i == 90 ) {
    sensor_ultrasonido(i);
    }
  }
 // Cuando acacava de hacer el bucle que esta en 180° llama al sensor
 // Le pongo 180 porque el servo esta en el grado 180
 sensor_ultrasonido(180);
```

- Now, we do the same loop *for* but in a negative way, it calls the servo when it's at 90º and it goes to 0º.

[42]

```
  // Para el sentido negativo
  for (int i = 180; i > 0; i-=5) { // i = i - 5
    servoMotor.write(i);
    delay(1);
    // Cuando el servo llega a 90°, llama al sensor y escanea
    if ( i == 90 ) {
      sensor_ultrasonido(i);
      }
  }

  // Cuando acacava de hacer el bucle que esta en 0° llama al sensor
  // Le pongo 0 porque el servo esta en el grado 0
  sensor_ultrasonido(0);
}
```

- We call the sensor_ultrasonido every time that the servo is at 0º, 90º or 180º. And we add to it the *i* variable, which is the degrees where the servo is when calls the function.

[41] Arduino code 3.
[42] Arduino code 3.

[43]

```
void sensor_ultrasonido(int i) {
  NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
  delay(250);  // Esperar 1 segundo entre mediciones
  int uS = sonar.ping_cm();  // Obtener medicion del objeto en cm


  // Si hay un objeto en una distancia menor a 15
  if ( uS && uS < 25) {

    // Esta en la derecha, hago que pare y que gire a la izquierda
    if (i == 0) {
       Parar(1000);
       Izquierda(500);
       Adelante(0);
      }

    // Si hay un obstaculo en medio hago que vaya girando y vaya escaneando hasta que no hay
    else if (i == 90){
      Atras(500);
      Derecha(500 );
      }

  // Hay un obstaculo en la izquierda, hago que gire a la derecha
    else if ( i == 180) {
      Atras(500);
      Derecha(500);
      Adelante(0);
      }
  }

  // Si no hay ningun objeto sigue adelante
  else {
   Adelante(0);
   }

}
```

- At the beginning of this new function we call the NewPing library which we import at the start of the NewPing.h code. Then we declare a new variable and it will be the mediation between the obstacles and the ultrasound sensor, measured in centimeters. We add to the function a 250 milliseconds delay.

- As you can see, there is a sentences that if the ultrasound sensor spots an object within 25 cm it scans in which position the servo is, the variable *i*.

- If the *i* variable is at 0º it means that there is an obstacle on his right, so it calls the *Izquierda* function so the Arduino can turn left.

- If *i* is at 90º, it means that there is an obstacle right in front of the car. In that case it calls the Derecha function and the arduino turns right.

- If *i* is at 180º it means that the obstacle is on his left, so the arduino turns right.

- In the event that the sentence is not complied with an object within 25cm, the arduino always moves forward and the ultrasound sensor keeps scanning.

[43] Arduino code 3.

- Later, doing tests with the Arduino, we realized that it is not necessary for the servo to turn completely 180 º, because it also captures the obstacles by turning a little less, so we lower the turns from 180 º to 155 º and, on the opposite side, from 0 º to 25 ° , so the loop of the Move_servo function remains this way:
[44]

```
/ Para el sentido positivo
 for (int i = 25; i <= 155; i+=5) { // i = i +5
   // Desplazamos el servo al angulo de la variable i, cada vez se ira incrementando.
   servoMotor.write(i);
   // Hacemos una pausa de 25ms por cada cambio de angulo
   delay(1);
   // Cuando el servo llega a 90º, llama al sensor y escanea
   if ( i == 90 ) {
     sensor_ultrasonido(i);
     }
   }
  // Cuando acacava de hacer el bucle que esta en 180º llama al sensor
  // Le pongo 180 porque el servo esta en el grado 180
  sensor_ultrasonido(155);

 // Para el sentido negativo
 for (int i = 155; i > 25; i-=5) { // i = i - 5
   servoMotor.write(i);
   delay(1);
   // Cuando el servo llega a 90º, llama al sensor y escanea
   if ( i == 90 ) {
     sensor_ultrasonido(i);
     }
 }
```

- With the previous code there was a fault, and is that, when the servo rotates completely to one side, did not give time to detect the object that was in front, at 90 º, and then collided with this.

- Changing the code and dropping down the degrees to which it arrives to rotate, we get that the servo has to rotate less and therefore gives time to detect if there is any obstacle in the center and avoid it.

[44] Arduino code 3

5. Test

- During the code modifications, although there haven't been many modifications, we have carried out tests to verify the correct functioning of the code.

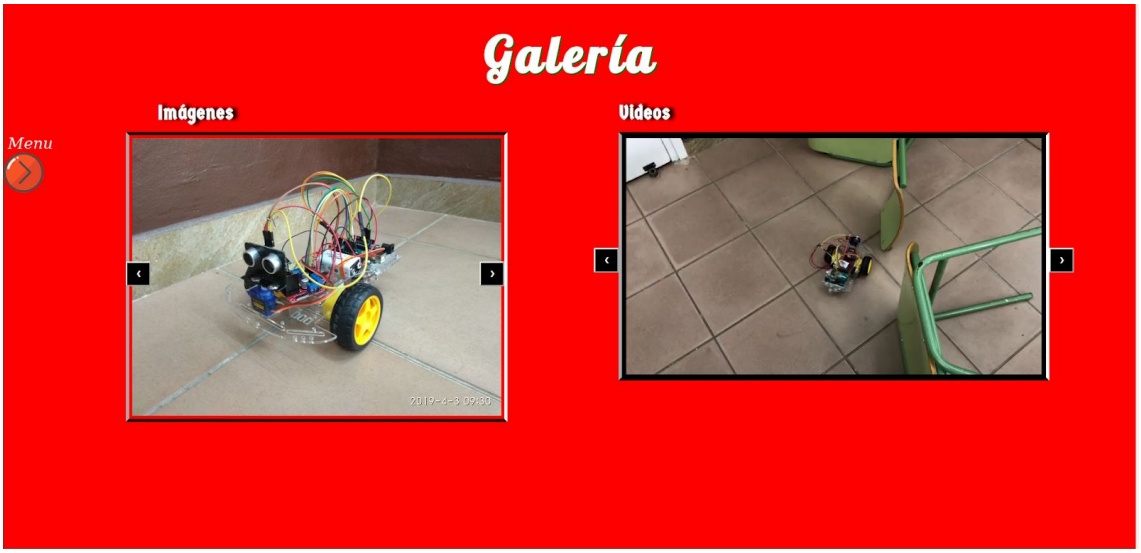
- The first test that we did, we verified that the Adelante, Atras, Derecha and Izquierda worked properly.

- Later on, when we connected the servo and ultrasonic sensor to the Arduino, we proved that the servo turned 180 º, and that the ultrasound sensor, when detecting the objects showed us a signal. For example, a change of direction in the direction of the wheels.

- Once verified that all the components of our Arduino worked and fulfilled the function that wore in the code, we started with the creation of the final code.

- Finally, we recorded the videos of the tests, once reached the final code. The videos of the tests recorded in class can be seen in the gallery section of our website.

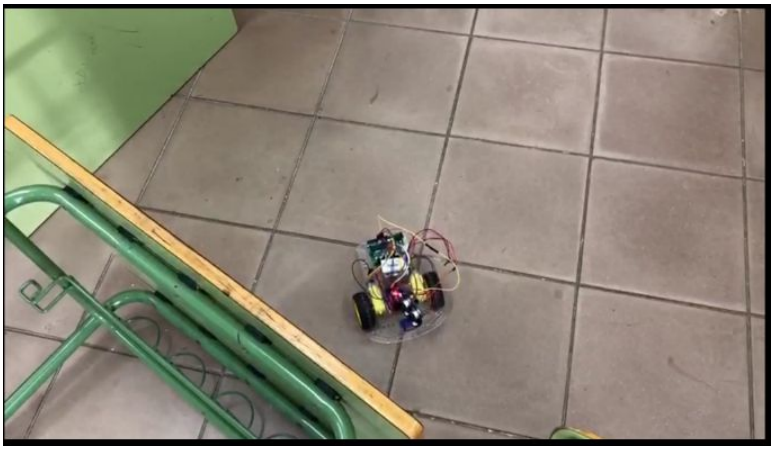
45



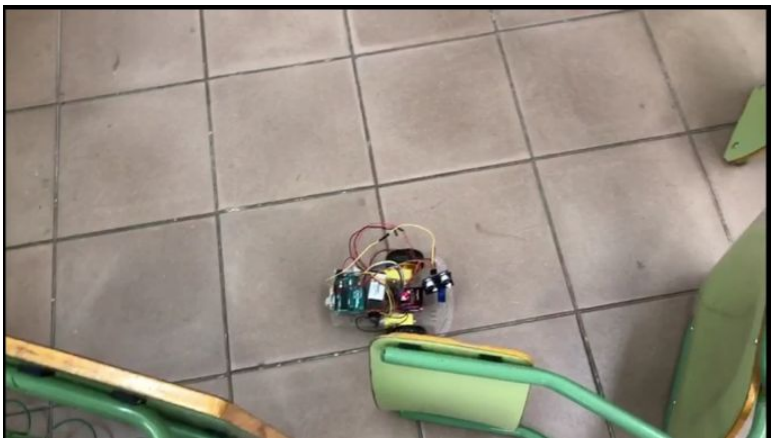
[45] Web Images

46

47



48



[46] Car image 1
[47] Car image 2
[48] Car image 3

- As we can see in the images, we put as obstacles chairs and tables lying down in the class, creating a circle which the Arduino had to dodge all obstacles without colliding with anything.

- In this way, we could verify that the code works as expected and that, when the Arduino detects an obstacle in the right turn to the left, if it detects it on the left it turns to the right and if it detects it in the center it rotates backwards and turns to one side.

6. Webpage

- We've thought about making a Web page to promote our Arduino Car, and at the same time, practice in the subject of WEB applications.

- We've tried to make the website in a modern way, "one page". We will create a menu where you can navigate the page. We've created the page programming with HTML5, CSS and Jquery. We've also used PHP, we will explain in a detailed way later.

6.1. Sections

The page has 4 sections, the sections are the following ones:

- Home.
- Features.
- Gallery.
- Contact.

6.1.1. Home

- This is the home page where we show a photo of our Arduino Car, and put some "posits" where we define our project with three adjectives.

[49]

[49] Arduino web 1

6.1.2. Features

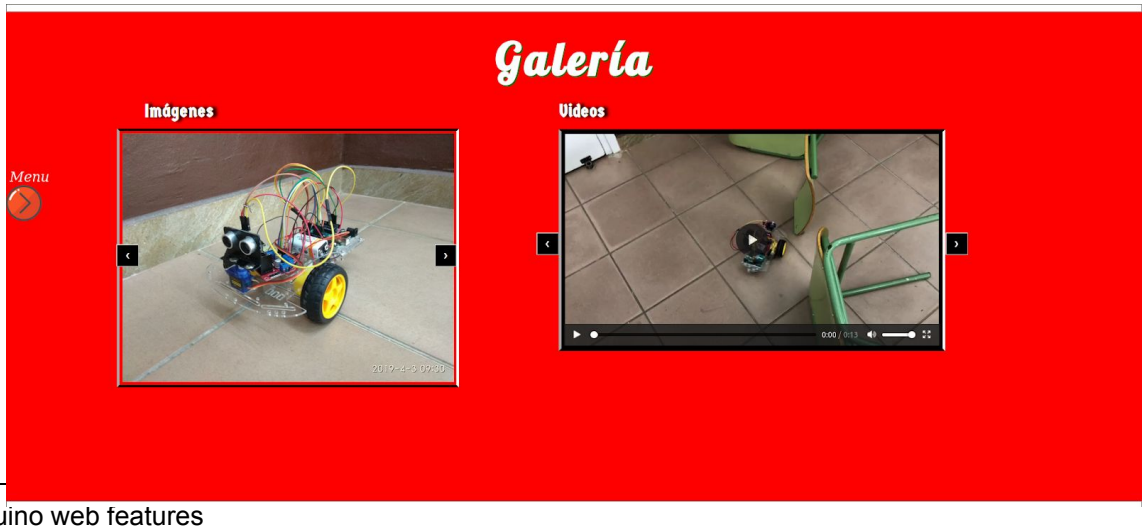
- It is the second page, we will put three circles and in each one of them explain an important characteristic of our Arduino Car.

[50]

6.1.3. Gallery

- In this section, we will put two carousels, one with photos of our car and in the other, we will put the three best videos of the tests.

[51]



[50] Arduino web features
[51] Arduino web gallery

6.1.4. Contact

- Finally, we will have the contact section, where we will find data from the project, and we will have a form to contact us. Later we will explain the function of this formulary.

[52]

6.2. Web Code

- We've created the web page with HTML5, CSS, Jquery and PHP for the contact formulary. Our web page is responsive. The web page is uploaded in Gitlab, like Rusben taught us.

- With that code we can make our web page public.
https://jmseguraf01.gitlab.io/web-arduino/

.gitlab-ci.yml 152 Bytes

```
pages:
  stage: deploy
  script:
  - mkdir .public
  - cp -r * .public
  - mv .public public
  artifacts:
    paths:
    - public
  only:
  - master
```

[52] Arduino web contact

- Thus, every time we edit the page code and upload it to Gitlab, the page is automatically updated on the server, but, as we will explain later, we will end up the Web page to a server on its own.

- You can check our web page code in that link: https://gitlab.com/jmseguraf01/web-arduino

- A screenshot of each programming language we've used:

HTML5:

```html
<!DOCTYPE html5>
<html>
  <head>
   <meta charset="UTF-8">
   <title>Coche Arduino</title>
   <!-- CSS -->
   <link rel="stylesheet" type="text/css" href="css/estilos.css"/>
   <!-- css para pantallas moviles -->
   <link rel="stylesheet" media="(max-width: 1024px)" href="css/movile_style.css" />
   <!-- ICONO -->
   <link rel="shortcut icon" href="images/icono.ico" />
   <!-- Google fonts -->
   <link href="https://fonts.googleapis.com/css?family=Germania+One" rel="stylesheet">
   <link href="https://fonts.googleapis.com/css?family=Lobster" rel="stylesheet">
   <link href="https://fonts.googleapis.com/css?family=VT323" rel="stylesheet">
   <link href="https://fonts.googleapis.com/css?family=Playfair+Display" rel="stylesheet">
  </head>

  <body>
    <!-- La div que engloba todo -->
    <div id="all">
      <!-- Div del titulo -->
      <div id="Inicio"></div>
      <div id="title_one_page">
        <p>Arduino Car</p>

      </div>

      <!-- Div de la primera pagina donde añado la foto en tamaño grande y el menu -->
      <div class="one_page">
        <!-- EL MENU -->
        <div id="menu_left">
          <!-- Si le damos se muestra el menu -->
          <div id="see_menu">
            <p>Menu</p>
            <img src="images/see_menu.png" width="100%">
          </div>
          <!-- Si le damos se oculta el menu -->
          <div id="hide menu">
```

[53]

[53] HTML

CSS:
54

```css
* {box-sizing: border-box; margin:0;}

body{
  background-color: black;
}

#all{
  /* background-image: linear-gradient(to right top, #727881, #565960, #3c3d41, #222224, #040404); */
  background-color: black;

}

/* Primera pagina */
.one_page{
/* height: 1150px; */
background-image: url("../images/coche_arduino.png");
background-repeat: no-repeat;
background-position: center;
background-size: 1024px;
height: 1024px;
width: 100%;
border-bottom: thin solid white;
border-width: thick;
}

#title_one_page{
  text-align: center;
  /* padding-top: 1%; */
  font-family: 'Germania One', cursive;
  color: white;
  font-size: 80px;
  letter-spacing: 3px;
  text-shadow: 2px 2px 0 black, 3px 4px 0 yellow;

}

#img_coche_arduino{
padding-top: 1%;
padding-left: 5%;
```

Javascript (Jquery):
55

```javascript
// Inicio el script de Jquery
$(document).ready(function(){
  // Para que cuando se recargue la pagina suba al inicio
  // $('html, body').animate({scrollTop: ($('#Inicio').offset().top)},500);


  // Cuando se clicka a la div de see_menu se ejecuta la siguiente funcion
  $( "#see_menu" ).click(function() {
    $("#sections_menu").css("display","block");
    $("#see_menu").css("display", "none"); // Para ocultar el div al que le das y sale el menu
    $("#sections_menu").css("animation","menu_animation_entry 0.5s linear both");  // Le añado una animacion defin
    $("#hide_menu").css("animation","entry_hide_menu 1s linear both");  // Le añado una animacion de entrada al bo
    $("#hide_menu").css("display","block");  // Hago que se vea el otro div para ocultar el menu
  });

  // Cuando se clica a la div de hide_menu se ejecuta la siguiente funcion
  $("#hide_menu").click(function(){
    $("#sections_menu").css("animation","menu_animation_exit 0.5s linear both");  // Le añado una animacion para l
    $("#hide_menu").css("animation","exit_hide_menu 0.5s linear both");  // Le añado una animacion de salida al bo
    setTimeout('$("#hide_menu").css("display", "none")', 600); // Hago que desaparezca el div de la pantalla despu
    setTimeout('$("#see_menu").css("display", "block");', 450);  // Para volver a poner el div de mostrar menu en
  });

  // Si le damos click a inicio, ejecutamos una funcion para que vaya a la parte de inicio
  $("#section_inicio").click(function(){
    $('html, body').animate({scrollTop: ($('#Inicio').offset().top)},500);
  });

    // Si le damos click a caracteristicas, ejecutamos una funcion para que vaya a la parte de caracteristicas
  $("#section_caracteristicas").click(function(){
    $('html, body').animate({scrollTop: ($('#caracteristicas').offset().top)},500);
  });

  // Si le damos click a galeria, funcion para que vaya a la parte de galeria
  $("#section_galeria").click(function(){
    $('html, body').animate({scrollTop: ($('#galeria').offset().top)},500);
  });

  // Si le damos click a contacto, funcion para que vaya a la parte de contacto en la pagina
```

---

[54] CSS
[55] Jquery

PHP
56

```php
<?php

$destino = "jmseguraf01@elpuig.xeill.net";
$nombre = $_POST["nombre"];
$email = $_POST["email"];
$mensaje = $_POST["mensaje"];

echo "Correo de $nombre\n";
echo "Con email: $email\n";
echo "Como mensaje: $mensaje";

mail($destino, "Web Arduino", "Nombre del cliente: $nombre\n, su correo és: $email, y su mensaje és: $mensaje")
?>
```

[56] PHP

6.3. Web server configuration

- Once we have the Web page finished we will host it on a Web server, which we mount using Apache as a Web server..

6.3.1. Apache web server

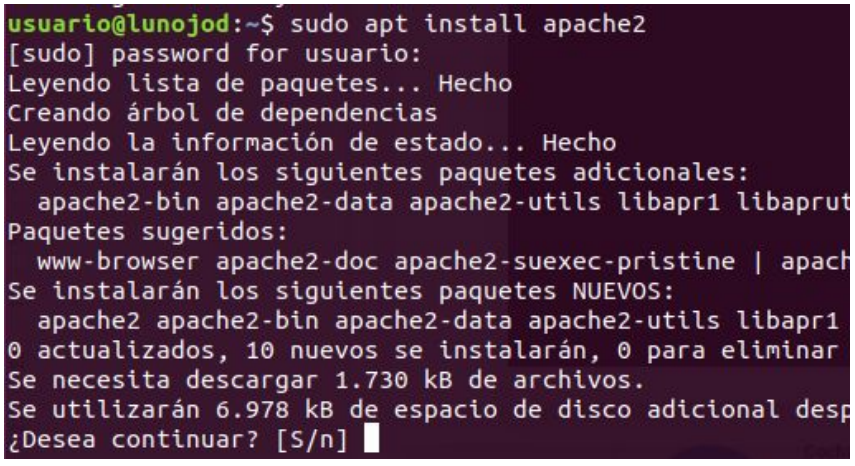
- We will use Apache to host our web page in a virtual machine with an Ubuntu Server 18.04, because it is a LTS version and offers us a longer support time.

- We will use an OVA-linked cloning provided by our teacher Виктор:

[57]

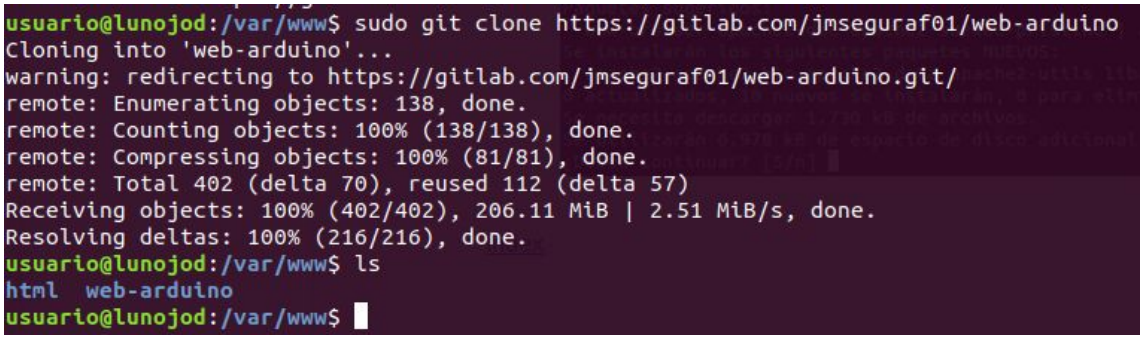
- We install Apache.

[58]

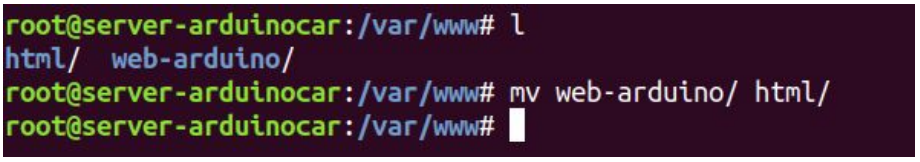


[57] Clone MV
[58] Apache2 installation

- Once Apache is installed, we go to the directory where Apache saves the web (/var/www), and we clone the repository of our page.
[59]



- Once downloaded it, we will move the Web-Arduino directory into /var/www/html.
[60]



- We will not create any VirtualHost, we will only edit the default VirtualHost and we will change the DocumentRoot to point to the directory of the Aduino website.

- We edit the ***/etc/apache2/sites-avaliable/000-default.conf*** file and we add the following lines:

[61]

- As we can see, we change the mail of the administrator of the Web, and the DocumentRoot.

- We reset the Apache service with the command systemctl restart Apache2.

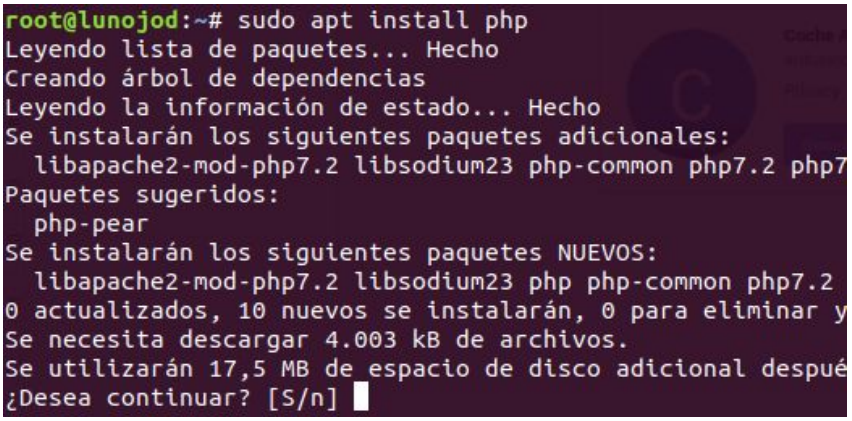
[59] Git clone web
[60] Arduino web.
[61] DocumentRoot

- We can verify by putting the IP of the Web server that, we get directly the Web page of the Arduino.

6.3.2. Mailutils mail server

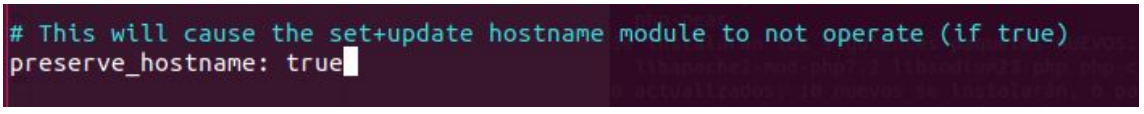
- We will need to have an email server like Mailutils in our web server so when they fill out and send the contact form, it fulfills its function that is to send an email to the administrator of the page thanks to the code of PHP.

  We install PHP:
  [62]



- Now we change the hostname in the /etc/hostname file:

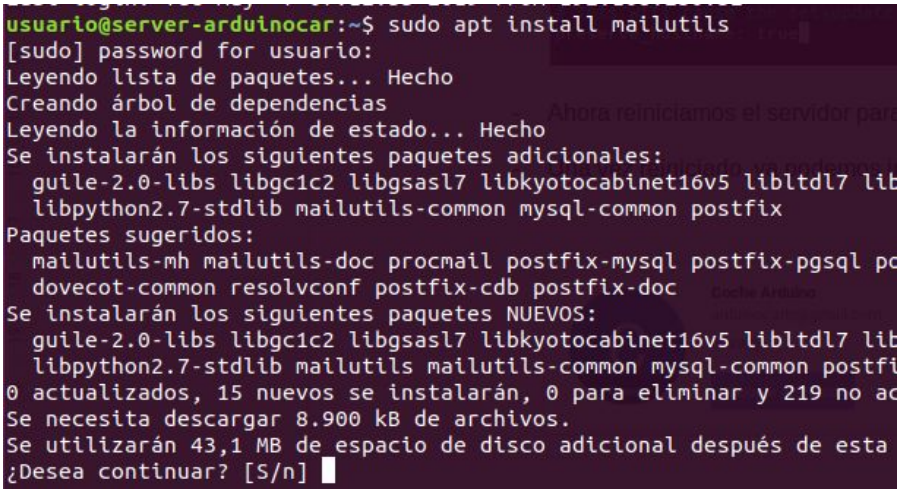
- We edit the /etc/cloud/cloud.cfg fil and we change the following lane:
  [63]



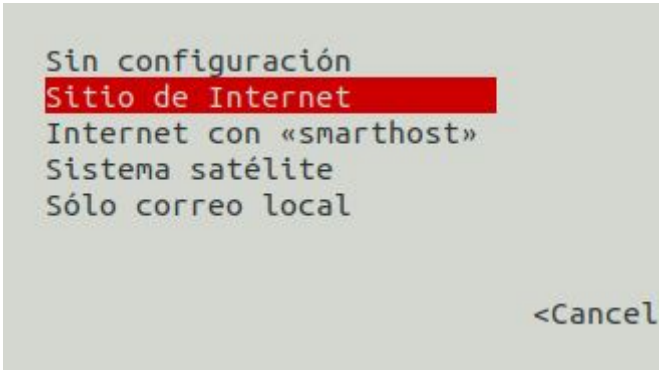
- We restart the service.

[62] PHP installation
[63] Preserve_hostname

- Once we've restarted the service we can install mailutils:
[64]



- We select the default option.
[65]

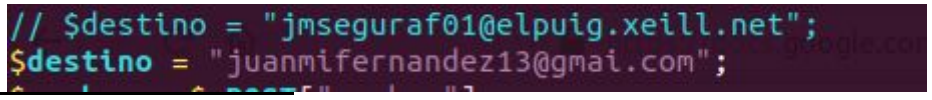


[64] Mailutils installation
[65] Default installation

- Once installed, the form will work and we send a mail so we can check it out by doing a that:
[66]

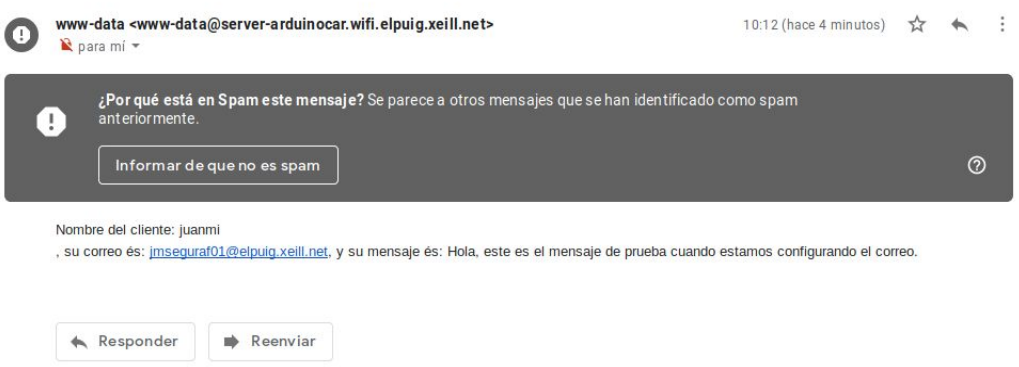
- If we send it and we check it out on the mail of the administrador (jmseguraf01@elpuig.xeill.net), we will see that it has worked.

- If we do the test in the institute it will give us an error, since mailutils can not send emails to the institute's domain from the institute so we will change the mail administrator and put my personal.
[67]



[66] Contact email 1
[67] Contact email 2

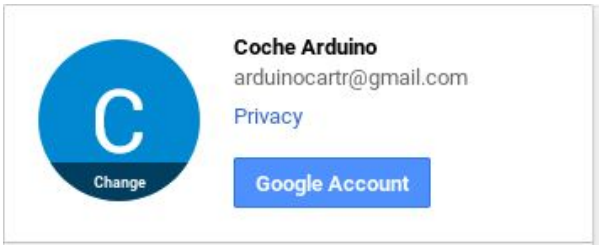
- We make sure it arrives.
[68]



- As we can see, the message arrives well but it arrives as spam. To avoid this, we will have to link a Gmail account to the mail server so that the messages do not arrive as Spam.

6.3.2.1. Link the Gmail Account

- To link a Gmail account to the mail server (Mailutils), the first thing we will do is create an account for the project:
[69]



(Arduinocartr1?!)

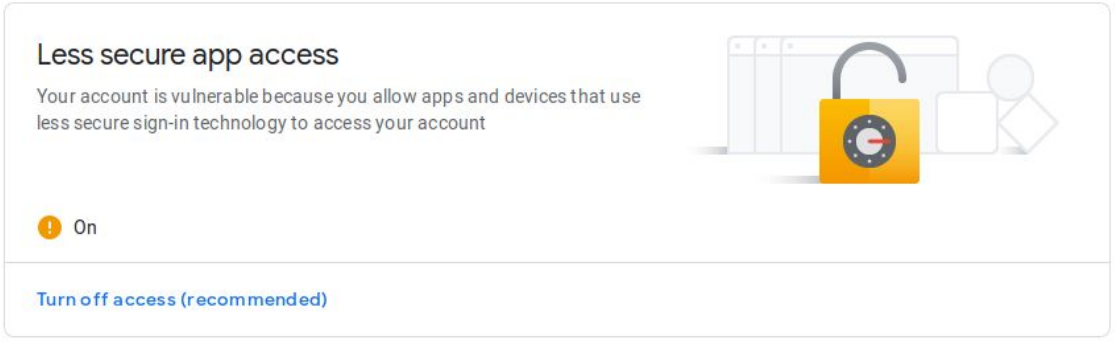
- Once we have the account created, as we can see in the image above, we have to change the settings to allow less unsafe applications to access your account:
    1. Go to your Google account.
    2. In the Navigation pane on the left, click Security.

[68] Email
[69] Tr email

3. In the Unsafe Application Access pane, located at the bottom of the page, click Enable Access.

Now, we must change the configuration of Postfix, on the mail server.

[70]



- **We have to create the following file**: ***/etc/postfix/sasl/sasl_passwd*** and add this:
  [smtp.gmail.com]:587 arduinocartr@gmail.com:Arduinocartr1?!
  [service]:port mail:password

- After that we create the database with this command:
  sudo postmap /etc/postfix/sasl/sasl_passwd

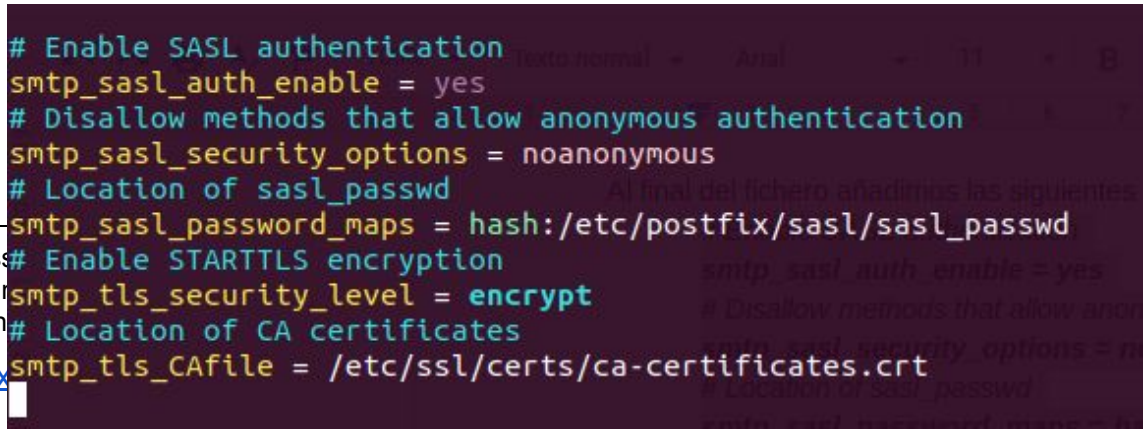
- We assign the permissions 660, to the file that we've just created.

- We add the following line in this file ***/etc/postfix/main.cf***.

[71]

- Finally, at the end of the file we must add the following lines:

[72]



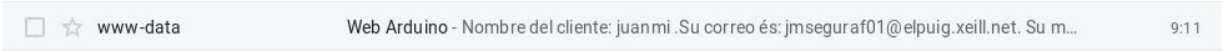
[70] Less
[71] Mai
[72] Con

Index

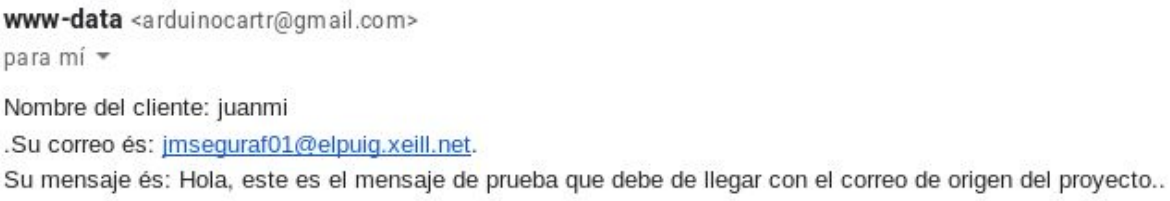
- We restart the postfix service:
  sudo systemctl restart postfix

- Now, we try to fill and send the contact form, and as a result, we will have to get to our mail a message with the data of the form and the origin of the mail account created above (*arduinocartr@gmail.com*).

73



74



- And finally, as we can see, the message does not leave us as spam and arrives as an email.

6.3.3. Domain configuration

- In the previous points, we've managed to upload the Web page to an own server created with Apache, and on the same server, install Mailutils so the contact form works. But this server can only be viewed from the local network. And we want that our server could be seen from outside the local network so that anyone can visit our project website.

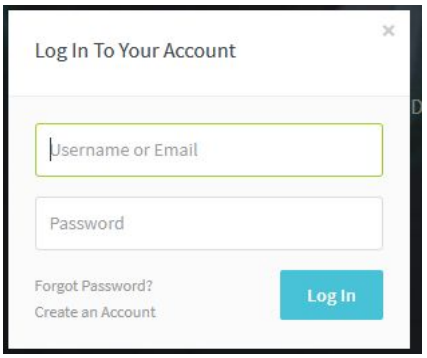
- We must link a web domain to the router where the server is located, then open a port on the router to indicate that any connection that goes to port 80, redirect to the IP address of our server.

- We start by getting a domain that takes us to the public IP of our router. This, we'll do it with No-Ip.

[73] Email
[74] Email content

- If this is the first time we use the service, we will have to register with an email account. In my case I'm already registered so we logg in.

[75]

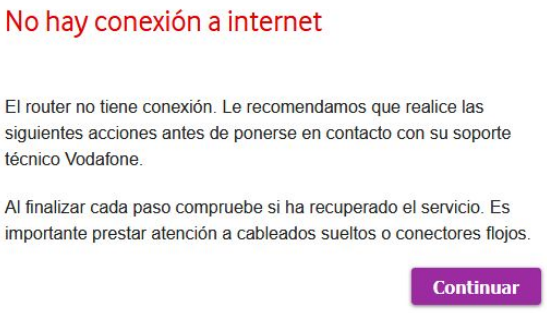


- We create the free domain.

[76]

- As we can see, this domain share us directly to our public IP address, in this way, every time we write the domain will take us to our router .

[77]



[75] Log in
[76] Free domain
[77] Error

- In this case, with our Vodafone router we can not access from within the network to the domain because it gives an error. But if we are out of the local network, we can access without any problem.

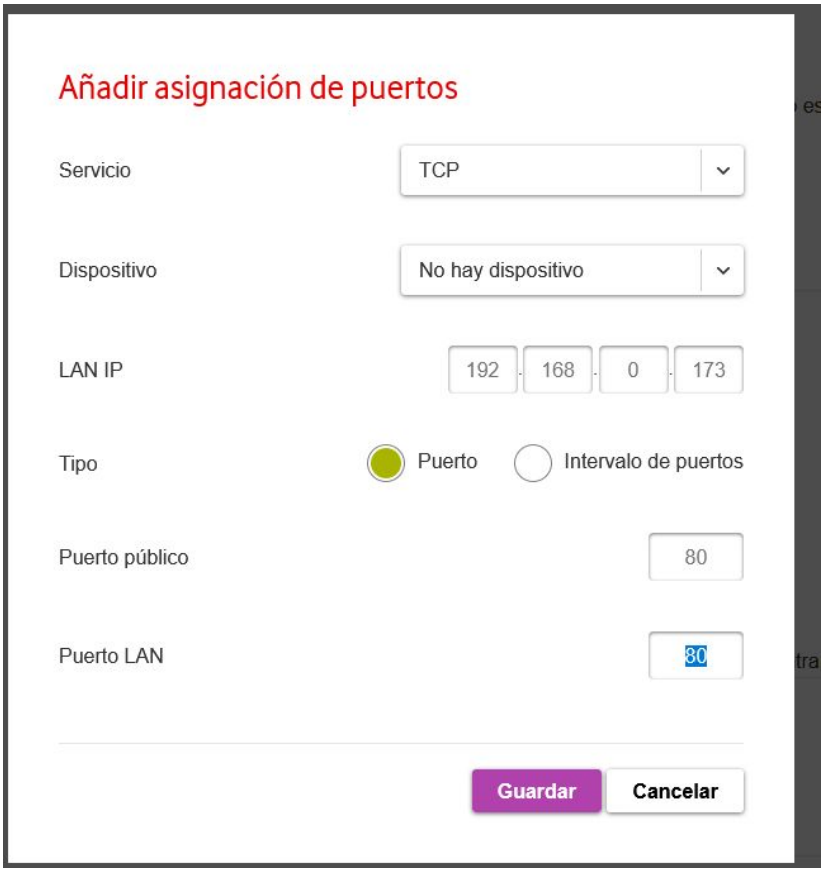
- The next step is to access to our router, open the 80 port and redirect the connection to the IP address of our web server.

- This step on each router is different as each has a different web interface configuration. In the case of Vodafone we seek port redirection..[78]

[78] Port

- We add a redirect here.

79



- LAN IP is the IP address of the server, the public port is the port to which will be accessed from outside the LAN and Port LAN is the port, with which the server is accessed within our local network.

[79] Port redirection

- Now, if we put the domain arduinocar.ddns.net, it will take us directly to the DocumentRoot of the file 000-default.conf, the directory of the Arduino website.

80

[80] Arduino web

6.3.4. Configuración HTTPS

- As we can see, we can access without any problem to the Web that is uploaded on our server. But this is an HTTP connection, it's not encrypted. In this chapter we will show how we encrypt our Web page with HTTPS from Let's encrypt.

- To start, we go to the Let's encrypt page, select that we have access to the shell and send us to https://certbot.eff.org. Here we select the software we use and the distribution of Linux.

[81]

- When selecting the parameters, we will get a guide with all the commands that we must run on our server to install Cerbot..
[82]

```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository universe
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ sudo apt-get install certbot python-certbot-apache
```

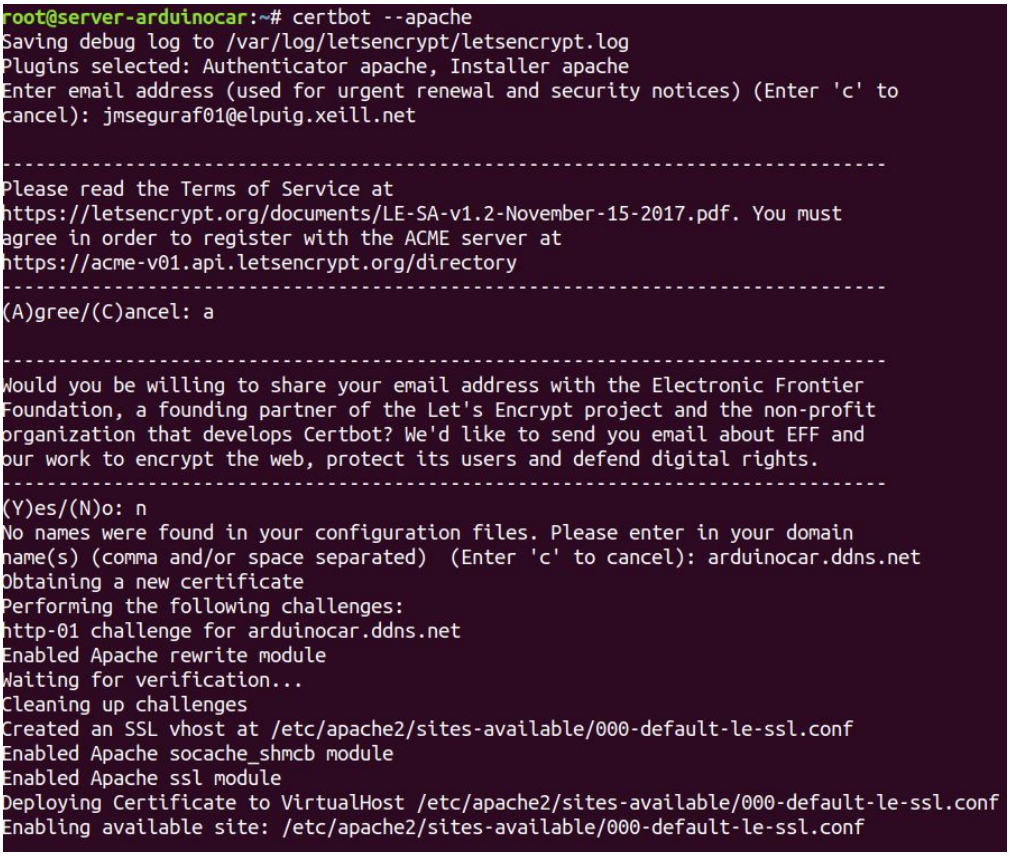
- We execute all the commands.

- Once the installation is finished, we execute the following command to automate the installation of the certificate.

[81] Installation
[82] Installation process

- Then it will ask a series of questions. The following image asks for the mail of the administrator, we accept the terms and conditions of the Let's encrypt service.

[83]



- Now, it will ask us for a domain name, we'll have to put the domain that the non-IP service has given us.

- Finally, it asks if we want to use HTTP, we redirect directly to HTTPS, in our case we say no.

[83] Installation

[84]

```
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
-------------------------------------------------------------------------------
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-------------------------------------------------------------------------------
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 1

-------------------------------------------------------------------------------
Congratulations! You have successfully enabled https://arduinocar.ddns.net

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=arduinocar.ddns.net
-------------------------------------------------------------------------------

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/arduinocar.ddns.net/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/arduinocar.ddns.net/privkey.pem
   Your cert will expire on 2019-08-12. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

root@server-arduinocar:~#
```

- Then we must open a router port that points to the LAN port 443, cause this is the insurance to connect by HTTPS

[84] Installation answers

- Our Vodafone router, does not let us Open the port 443 face to outside the network, so we open the port 4443 and add a redirection in Apache.

[85]

- Once we've opened the port, if we put the domain address https://arduinocar.ddns.net:4443, it will take us to the Web page in HTTPS with the Let's Encrypt certificate.

- But we want it to take us directly when we put the domain name, without having to put the port and HTTPS, for this we will add a redirect on the Apache server.

- We have enable the rewrite module with this command:
  sudo a2enmod rewrite

[86]

```
root@server-arduinocar:/etc/apache2/sites-available# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

[85] Configuration
[86] Rewrite module

- We edit the default VirtualHost (000-default. conf) and we add a rewrite rule:
[87]

```
# Redirección al puerto 4443 para el https
RewriteEngine On
RewriteRule ^/(.*) https://arduinocar.ddns.net:4443
```

- We restart the apache service with systemctl restart apache2.

- Now, if we search arduinocar.ddns.net in our browser, it takes us to the HTTPS version automatically.

[88]

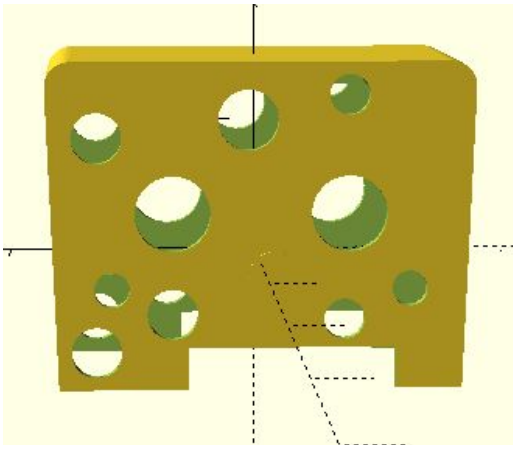
[87] Rewrite rule
[88] HTTPS web

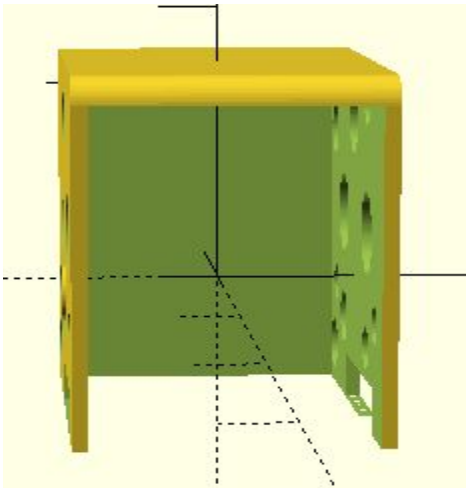
7. 3D Housing

- So that our Arduino car is more beautiful aesthetically, we decided to create a casing. We thought about creating it with wood, but our teacher recommended us to use the 3D printer that we have in high school and so we did.

  In the creation of the case we use the following steps:
  - To begin with, lake in the measures across and across our car.
  - Then we started to create the casing with simple geometric shapes as cubes, fused a small cube into another larger cube to get the casing will be hollow. All this, following the measures taken earlier.
  - Then, we decided to make a hole in the front, so that, just ahead you could see the interior of the car Arduino.
  - Later, in the two side walls, we made holes of "random" shape so that it would be a more beautiful and at the same time, more original aesthetic form.
  - Finally, we decided to put a rounded edge to the cube to make it look prettier.

- Once the case is finished, the 3D visualization of the program is as follows:





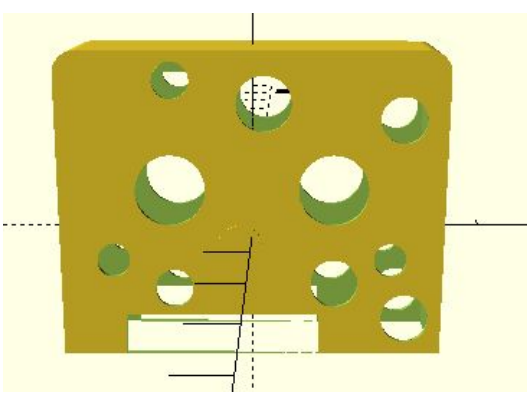
89

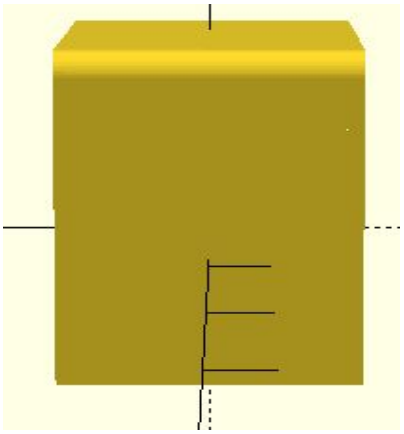
90

[89] Housing 1
[90] Housing 2

[91]

[92]





- Once the case is finished, we put it to print, but we had several problems with the printer because, it did not get the print thread well. We tried to print it several times until we hit the trick, put the print thread the fattest size possible.

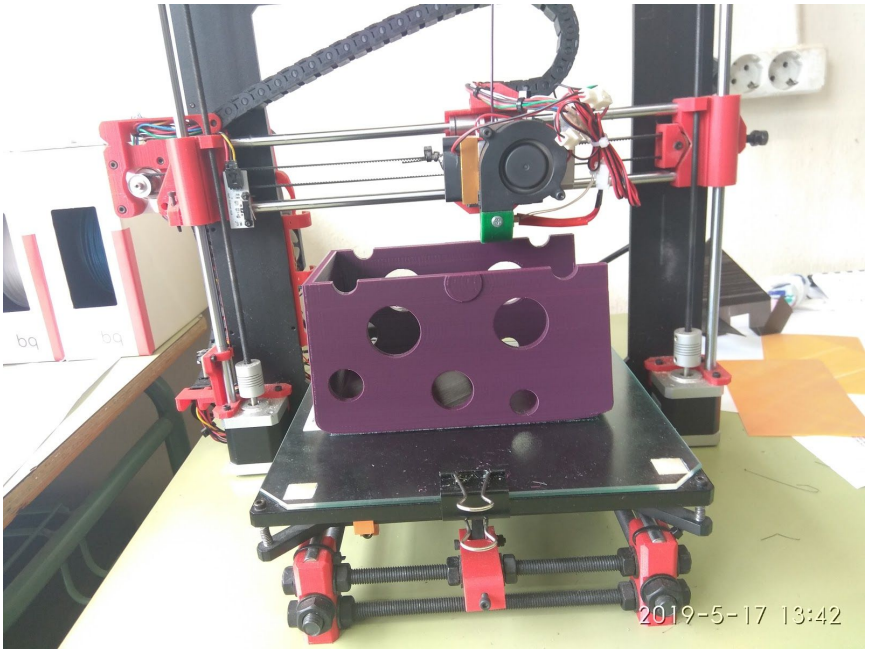
- Printer indicated 24h and 9m printing.

- These are some photos of the printing process:

[93]

[91] Housing 3
[92] Housing 4
[93] Impression process 1

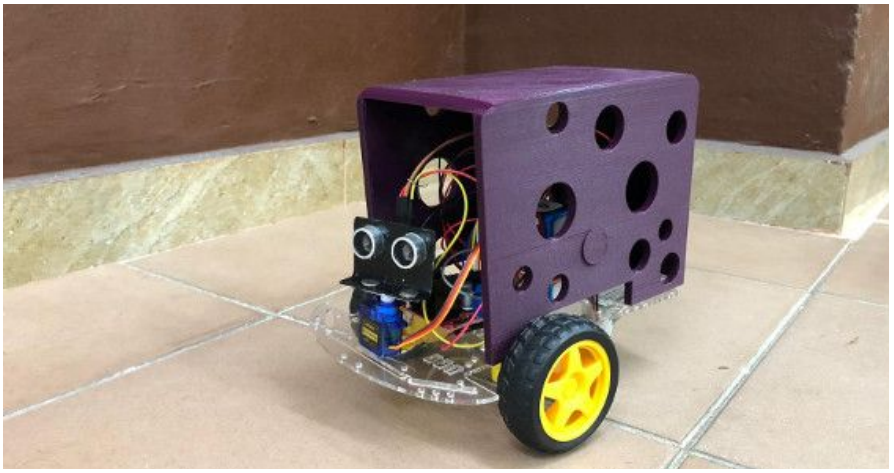
[94]



- Here we can see a video of the housing impresion.

- Finally, with the housing put on, the car looks like that :

[95]



[94] Impression process 2
[95] Arduino car with the housing

8. Conclusions

Juan Miguel Segura and Alejandro Mallén have reached the following conclusion. We have acquired many new knowledge on Arduino electricity and assembly. A field we've never tried before. As for the programming part of the Arduino we have learned to program in C.

With the website we have been able to develop a Web page from 0. That is hosted on a laptop at Juan Miguel's house and an email server so they can contact us from the website. All this we have been able to realize applying the knowledge acquired in the subjects of services and WEB applications.

We have also had the opportunity to create a car casing with the OpenSCAD software and print it on the 3D printer of our center.

In conclusion, we could not have done most of this work without the knowledge acquired in class.

9. Bibliography

- Search for required materials:
  https://leantec.es/blog/13_Robot-autonomo-esquiva-objetos.html
  https://programarfacil.com/tutoriales/fragmentos/servomotor-con-arduino/

- Code, assembly and diagrams:
  https://programarfacil.com/tutoriales/fragmentos/servomotor-con-arduino/
  https://leantec.es/blog/13_Robot-autonomo-esquiva-objetos.html
  https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/
  https://www.prometec.net/coche-arduino-l298n/

- Help with the Web page (CSS/HTML5/jquery) and PHP:
  https://code.tutsplus.com/es/tutorials/create-a-sticky-note-effect-in-5-easy-steps-with-css3-and-html5--net-13934
  https://www.youtube.com/watch?v=Smkw2vBizIY
  https://www.w3schools.com/bootstrap/bootstrap_carousel.asp

10. Annexos

Code Arduino

```
/* Juan Miguel Segura y Alejandro Mallen
Trabajo de sintesi 2n SMX
Licencia GPL
*/

// Incluimos la libreria para controlar el servo y el sensor de ultrasonido
#include <Servo.h>
#include <NewPing.h>


// Aqui se configuran los pines donde debemos conectar el sensor
#define TRIGGER_PIN  2
#define ECHO_PIN     3
#define MAX_DISTANCE 200

// Variables del motor A
int ENA = 6;
int IN1 = 13;
int IN2 = 12;

// Variables del motor B
int ENB = 5;
int IN3 = 11;
int IN4 = 10;

// Variable del servo
Servo servoMotor;

// Indicamos la variable de la velocidad
int vel = 80;

// Funcion de setup
void setup() {
Serial.begin(9600);
 // Declaramos todos los pines como salidas
 pinMode (ENA, OUTPUT);
 pinMode (ENB, OUTPUT);
 pinMode (IN1, OUTPUT);
 pinMode (IN2, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (IN4, OUTPUT);
 servoMotor.attach(9); // Declaramos el servo para que trabaje con el pin 9
}

// Funcion 'loop' es lo que se ejecuta
void loop() {
  move_servo();
//sensor_ultrasonido(90);
}
```

```
void Adelante (int time)
{
 //Direccion motor A
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, vel); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, 87); //Velocidad motor B / Le pongo mas porque hay diferencia

 delay(time);
}

void Atras (int time)
{
 //Direccion motor A
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, vel); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, vel); //Velocidad motor B

 delay(time);
}


void Derecha (int time)
{
 //Direccion motor A
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, vel); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, vel); //Velocidad motor A

  delay(time);

}
```

```
// Funcion para parar el coche
void Parar(int time){
  //Direccion motor A
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, LOW);
  analogWrite (ENA, vel); //Velocidad motor A
  //Direccion motor B
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, LOW);
  analogWrite (ENB, vel); //Velocidad motor B
  delay(time);
}

void Izquierda (int time)
{
 //Direccion motor A
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, vel); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, vel); //Velocidad motor B

 delay(time);
}
```

```
// Funcion para mover el servo
void move_servo() {
// Vamos a tener dos bucles uno para mover en sentido positivo y otro en sentido negativo
/*
180º = esta en la izquierda
90º = en medio
0º = esta en la derecha
 */

// Para el sentido positivo
  for (int i = 25; i <= 155; i+=5) { // i = i +5
    // Desplazamos el servo al angulo de la variable i, cada vez se ira incrementando.
    servoMotor.write(i);
    // Hacemos una pausa de 25ms por cada cambio de angulo
    delay(1);
    // Cuando el servo llega a 90º, llama al sensor y escanea
    if ( i == 90 ) {
      sensor_ultrasonido(i);
      }
    }
   // Cuando acacava de hacer el bucle que esta en 180º llama al sensor
   // Le pongo 180 porque el servo esta en el grado 180
   sensor_ultrasonido(155);

  // Para el sentido negativo
  for (int i = 155; i > 25; i-=5) { // i = i - 5
    servoMotor.write(i);
    delay(1);
    // Cuando el servo llega a 90º, llama al sensor y escanea
    if ( i == 90 ) {
      sensor_ultrasonido(i);
      }
  }

  // Cuando acacava de hacer el bucle que esta en 0º llama al sensor
  // Le pongo 0 porque el servo esta en el grado 0
  sensor_ultrasonido(25);
}

void sensor_ultrasonido(int i) {
  NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
  delay(250);  // Esperar 1 segundo entre mediciones
  int uS = sonar.ping_cm();  // Obtener medicion del objeto en cm


  // Si hay un objeto en una distancia menor a 15
  if ( uS && uS < 25) {

    // Esta en la derecha, hago que pare y que gire a la izquierda
    if (i == 25) {
       Parar(1000);
       Izquierda(500);
       Adelante(0);
      }
```

```
    // Si hay un obstaculo en medio hago que vaya girando y vaya escaneando hasta que no hay
    else if (i == 90){
      Atras(500);
      Derecha(500 );
      Serial.print("Objeto detectado en medio");

      /*while ( uS < 25 ) {

        // Si el coche esta muy cerca del obstaculo tira para atras
        if (uS < 6 ) {
          Atras(150);
          }

        // Vuelvo a mirar a ver si hay algun obstaculo
        uS = sonar.ping_cm();
        Derecha(150);
        Parar(1000);
        Serial.print("obstaculo");
        }*/
      }

  // Hay un obstaculo en la izquierda, hago que gire a la derecha
    else if ( i == 155) {
      Atras(500);
      Derecha(500);
      Adelante(0);
      }
  }

  // Si no hay ningun objeto sigue adelante
  else {
   Adelante(0);
   }

}
```