**Universitatea Politehnica Timisoara**

|  |
| --- |
| FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE  **Embedded Systems:** |
| Arduino Based Car controlled via WIFI |
|  |
|  |
| **Steleac Raul-Dacian și Roșu Andrei** |
| **2018-2019** |
| Contents  [1.Target of the project 3](#_Toc4347455)  [2. The Microcontroller 4](#_Toc4347456)  [2.1 General Overview 4](#_Toc4347457)  [2.2 Summary 5](#_Toc4347458)  [2.3 Power 5](#_Toc4347459)  [2.4 Memory 6](#_Toc4347460)  [2.5 Input and Output 7](#_Toc4347461)  [2.6 Communication 8](#_Toc4347462)  [2.7 Programming 9](#_Toc4347463)  [2.8 Schematics 10](#_Toc4347464)  [3. Modules 11](#_Toc4347465)  [3.1 PWM 11](#_Toc4347466)  [4. Display System 12](#_Toc4347467)  [5. Sensors 13](#_Toc4347468)  [5.1 Ultrasonic Sensor HC-SRO4 13](#_Toc4347469)  [How does it work? 13](#_Toc4347470)  [How do we use it? 14](#_Toc4347471)  [5.2 ESP8266 NodeMCU 1.0 14](#_Toc4347472)  [8.Bibliography 17](#_Toc4347473) |

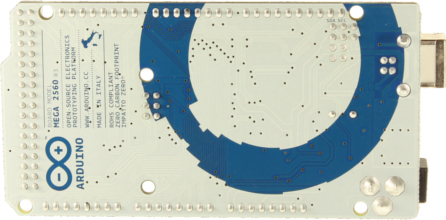
|  |
| --- |
|  |
|  |
| 1.Target of the project The targer set for the project is to assamble a car based on an Arduino MEGA  2560 that can perform the follwing tasks:  - being able to detect obstacles that come in front of the car.  - host a website that allows control over the the movement functionalities of the car.  - being able to sing notes via a speaker. |
|  |

# 2. The Microcontroller

## 2.1 General Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

The Mega 2560 is an update to the Arduino Mega, which it replaces.



## 2.2 Summary

Microcontroller ATmega2560

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limits) 6-20V

Digital I/O Pins 54 (of which 14 provide PWM output)

Analog Input Pins 16

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 256 KB of which 8 KB used by bootloader

SRAM 8 KB

EEPROM 4 KB

Clock Speed 16 MHz

## 2.3 Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 boards) programmed as a USB-to-serial converter.

Revision 2 of the Mega2560 board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

Stronger RESET circuit.

Atmega 16U2 replace the 8U2.

The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator or be supplied by USB or another regulated 5V supply.

3V3. A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND. Ground pins.

## 2.4 Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

## 2.5 Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode (), digitalWrite (), and digitalRead () functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

* Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
* External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt () function for details.
* PWM: 0 to 13. Provide 8-bit PWM output with the analogWrite () function.
* SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
* LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
* TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference () function.

There are a couple of other pins on the board:

* AREF. Reference voltage for the analog inputs. Used with analogReference ().
* Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block

## 2.6 Communication

The Arduino Mega2560 has several facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 (ATmega 8U2 on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.

The ATmega2560 also supports TWI and SPI communication. The Arduino software includes a Wire library to simplify use of the TWI bus; see the documentation for details. For SPI communication, use the SPI library.

## 2.7 Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.

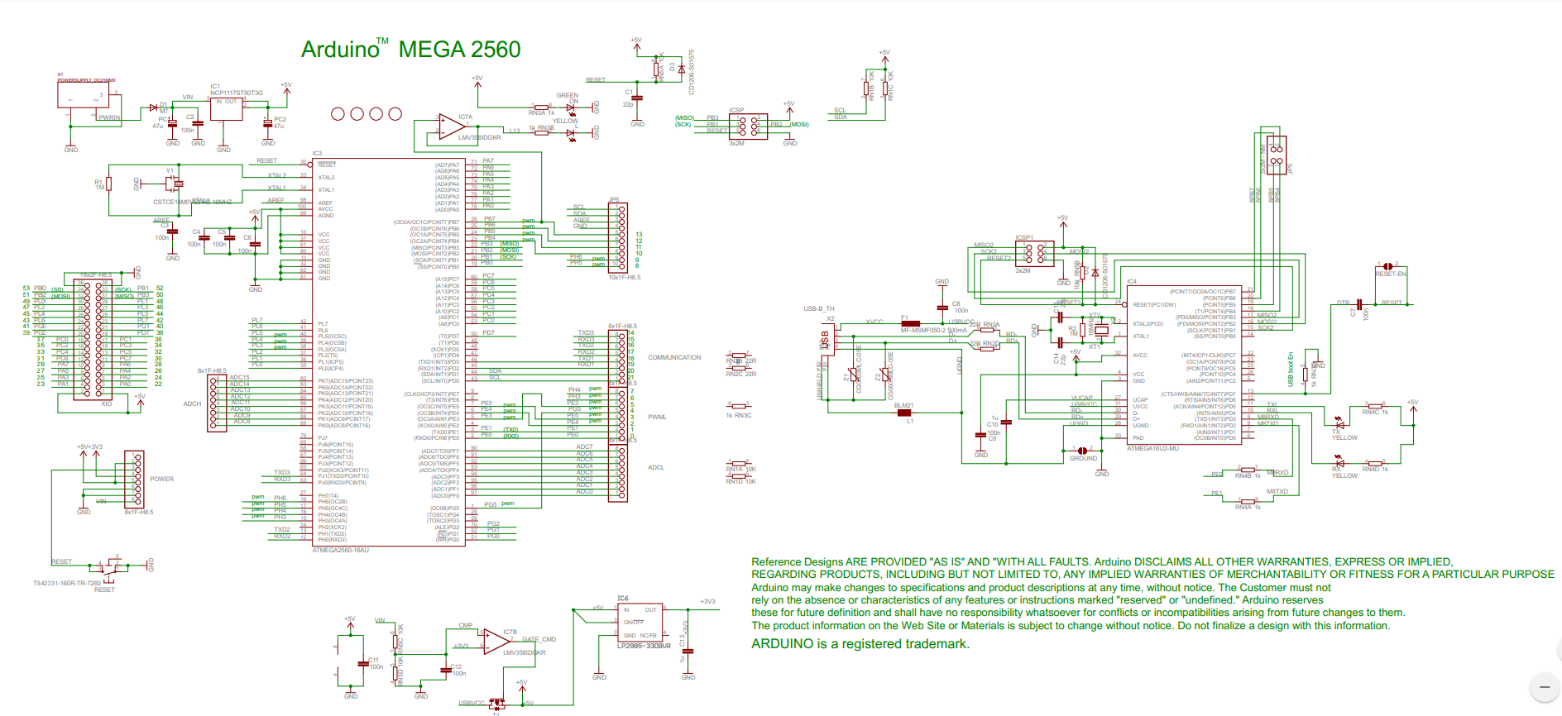
The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

## 2.8 Schematics



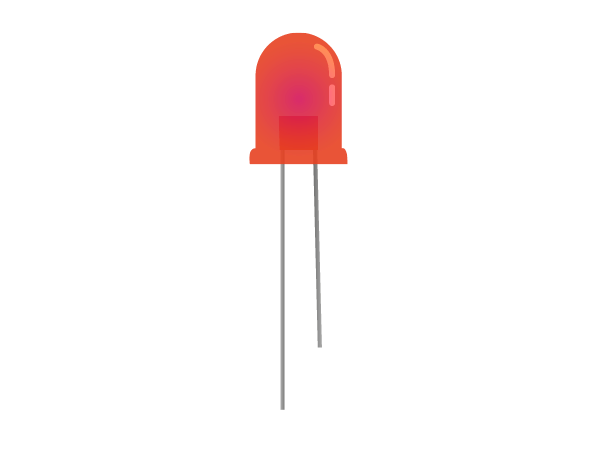
# 3. Modules

The modules of the microcontroller that are used in this project are the PWM.

## 3.1 PWM

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

# 4. Display System

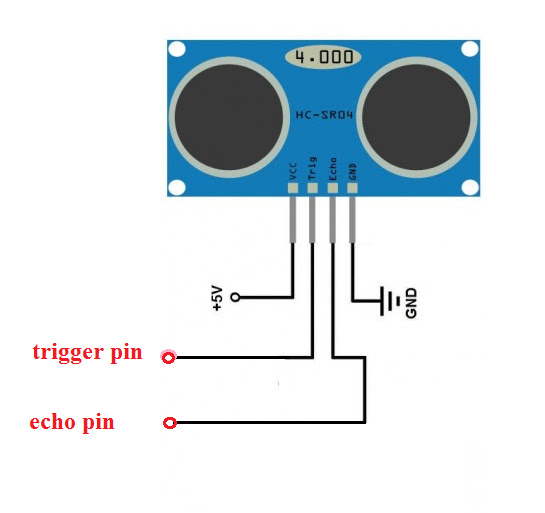
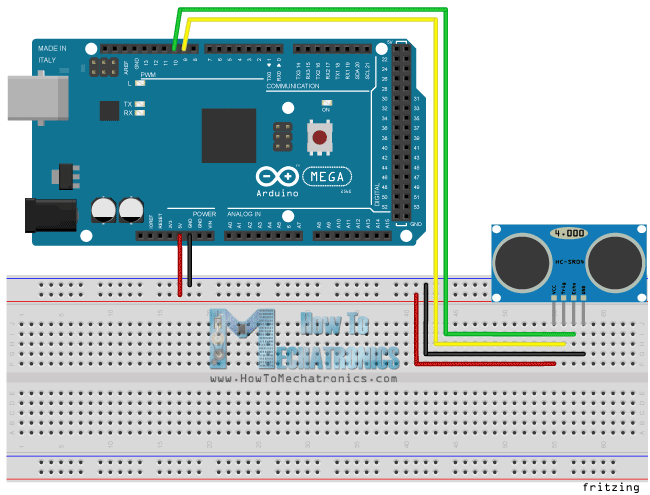
The Display System is based on 4 Mono-colored LEDS that are used to signal turns and the stop state of the car.

The LED is a light-emitting diode is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons can recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

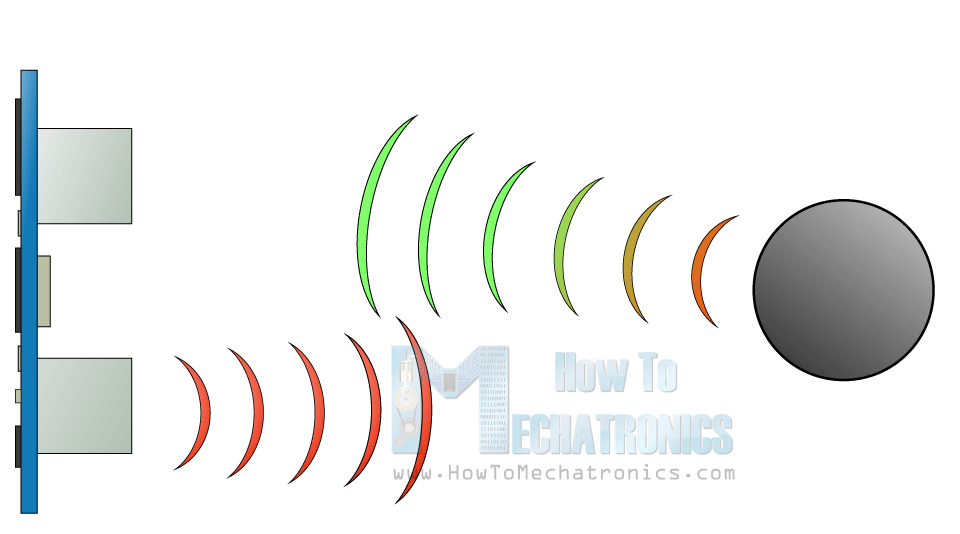
# 5. Sensors

## 5.1 Ultrasonic Sensor HC-SRO4

### How does it work?

It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

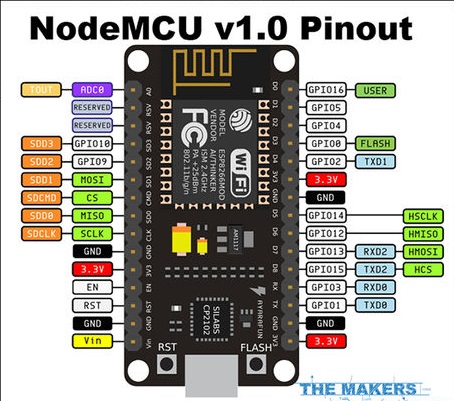
The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

In order to generate the ultrasound, you need to set the Trig on a High State for 10 µs. That will send out an 8-cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.

### How do we use it?

In the loop first you must make sure that the trigPin is clear, so you must set that pin on a LOW State for just 2 µs. Now for generating the Ultra sound wave we must set the trigPin on HIGH State for 10 µs. Using the ***pulseIn ()*** function you must read the travel time and put that value into the variable “duration”. This function has 2 parameters, the first one is the name of the echo pin and for the second one you can write either HIGH or LOW. In this case, HIGH means that the ***pulsIn()***function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously.

## 5.2 ESP8266 NodeMCU 1.0

NodeMCU is an open source [IoT](https://en.wikipedia.org/wiki/Internet_of_Things) platform. It includes [firmware](https://en.wikipedia.org/wiki/Firmware) which runs on the [ESP8266](https://en.wikipedia.org/wiki/ESP8266) [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) [SoC](https://en.wikipedia.org/wiki/System_on_a_chip) from [Espressif Systems](https://en.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1" \o "Espressif Systems (page does not exist)), and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the [Lua](https://en.wikipedia.org/wiki/Lua_(programming_language)" \o "Lua (programming language))scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and [SPIFFS](https://en.wikipedia.org/w/index.php?title=SPIFFS&action=edit&redlink=1).

At the core of the NodeMCU 1.0 there is a ESP8266 chip.

The ESP8266 is a low-cost [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) microchip with full [TCP/IP stack](https://en.wikipedia.org/wiki/TCP/IP_stack) and [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) capability.

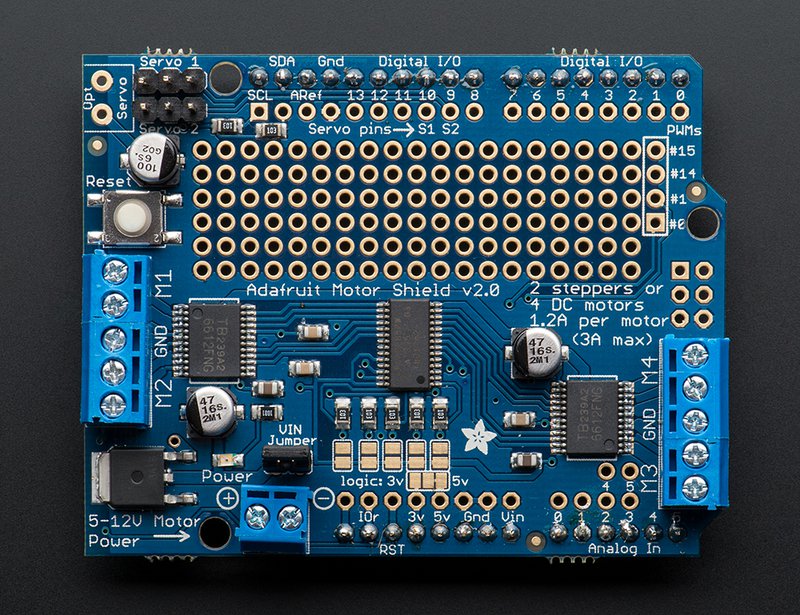
### Features of the ESP8266

* Processor: L106 32-bit [RISC](https://en.wikipedia.org/wiki/Reduced_instruction_set_computing) microprocessor core based on the [Tensilica](https://en.wikipedia.org/wiki/Tensilica" \o "Tensilica) Xtensa Diamond Standard 106Micro running at 80 MHz[[5]](https://en.wikipedia.org/wiki/ESP8266#cite_note-5)
* Memory:
  + 32 KiB instruction RAM
  + 32 KiB instruction cache RAM
  + 80 KiB user-data RAM
  + 16 KiB ETS system-data RAM
* External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
* [IEEE 802.11](https://en.wikipedia.org/wiki/IEEE_802.11) b/g/n [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi)
  + Integrated [TR switch](https://en.wikipedia.org/wiki/Duplexer#Transmit-receive_switch), [balun](https://en.wikipedia.org/wiki/Balun" \o "Balun), [LNA](https://en.wikipedia.org/wiki/Low-noise_amplifier), [power amplifier](https://en.wikipedia.org/wiki/RF_power_amplifier) and [matching network](https://en.wikipedia.org/wiki/Matching_network)
  + [WEP](https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy) or [WPA/WPA2](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access) authentication, or open networks
* 16 [GPIO](https://en.wikipedia.org/wiki/General-purpose_input/output) pins
* [SPI](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus)
* [I²C](https://en.wikipedia.org/wiki/I%C2%B2C) (software implementation)[[6]](https://en.wikipedia.org/wiki/ESP8266#cite_note-EspressifBBS_I2C-6)
* [I²S](https://en.wikipedia.org/wiki/I%C2%B2S) interfaces with DMA (sharing pins with GPIO)
* [UART](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter) on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
* 10-bit [ADC](https://en.wikipedia.org/wiki/Analog-to-digital_converter) ([successive approximation ADC](https://en.wikipedia.org/wiki/Successive_approximation_ADC))

The NodeMCU a microcontroller that is the base of the WIFI communication and server hosting of this project. The MCU hosts a web server that can be accessed from a Smart Phone or a PC via a LAN IP address of the MCU. After the website is accessed we can press the buttons that are at our disposal to control the message that is written on the analog pins by the MCU. Later on the pins will be read by the Arduio MEGA and se the PWM and direction of the motors.

Also from the website the song can be played.

# 5.3 DK Electronics Motor Shield V1.0

 The Motor Shield is based on the L293D which is quadruple high-current half-H drivers.It is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications. All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo- Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

In our project we you this to control the direction of the motors and set the PWM of them.

To do so we digital write the PWM and the 2 direction pins, setting one on HIGH and the other one on LOW depending on the clockwise or counterclockwise direction of the motors.

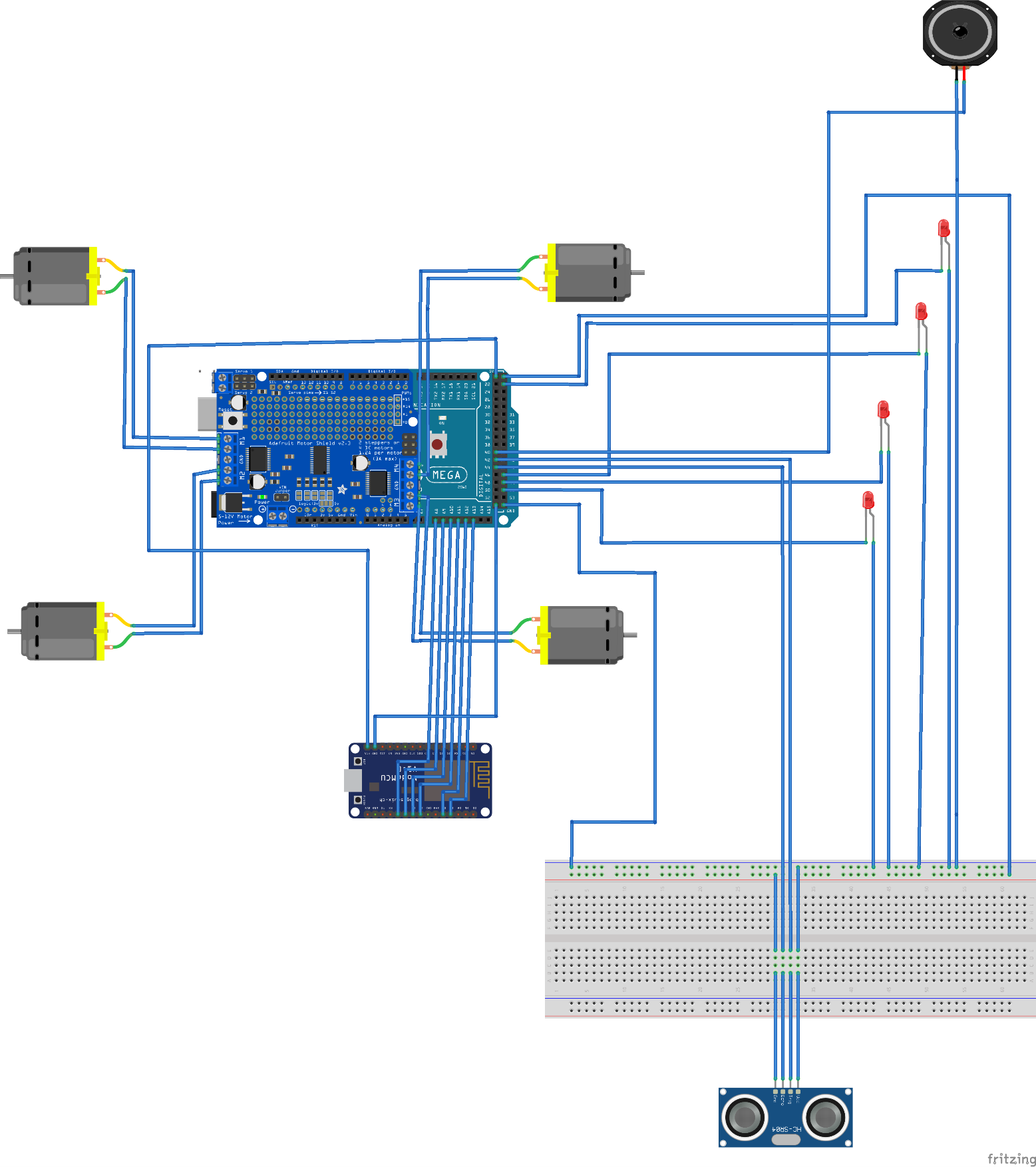
# 5.4 DC motors

We are using 4 DC motors to power the movement of the car. The power of the motors is connected to the H-Bridges from 2 Lithium Accumulators. And so the Arduino has a power separate from the motors to ensure safety of our circuit.

# 5.5 Speaker

We also connected an old speaker to emit musical notes.

# 6. Hardware schematics



# 7.Programming

**Arduino MEGA code:**

#include <AFMotor.h>

#include "music.h"

AF\_DCMotor motorA(1);

AF\_DCMotor motorB(2);

AF\_DCMotor motorD(3);

AF\_DCMotor motorC(4);

enum States{

Idle,

Forward,

Backward,

Left,

Right,

Sing,

Singing

};

int CurrentState = Idle;

//Analogical pins used for sending information from NodeMcu to Adruino Mega

int p1 = A8;

int p2 = A9;

int p3 = A11;

int p4 = A10;

int p5 = A12;

int p6 = A13;

const int trigPin = 42;

const int echoPin = 44;

long duration, inches, cm;

//Initialization of the speed of each motor and pins

void setup()

{

motorA.setSpeed(250);

motorB.setSpeed(250);

motorD.setSpeed(250);

motorC.setSpeed(250);

pinMode(23, OUTPUT);

pinMode(47, OUTPUT);

pinMode(49, OUTPUT);

pinMode(51, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

Serial.begin(9600);// sets the digital pin 7 as input

}

void runCurrentState()

{

switch(CurrentState)

{

case Idle : SM\_Idle();break;

case Forward : SM\_Forward();break;

case Backward : SM\_Backward();break;

case Left : SM\_Left();break;

case Right : SM\_Right();break;

case Sing : SM\_StartSing();break;

case Singing: SM\_Sing();break;

}

}

void SM\_Forward()

{

//Ultrasonic sensor check to avoid crashing

if(cm < 25)

{

sang\_flag = 0;

motorA.run(RELEASE);

motorB.run(RELEASE);

motorD.run(RELEASE);

motorC.run(RELEASE);

for (int thisNote = 0; thisNote < 3; thisNote++)

{

int noteDuration = 1000/noteDurations[thisNote];

tone(40, melody[thisNote],noteDuration);

int pauseBetweenNotes = noteDuration \* 1.30;

delay(pauseBetweenNotes);

noTone(8);

}

}

else

{

sang\_flag = 0;

motorA.run(FORWARD);

motorB.run(FORWARD);

motorD.run(FORWARD);

motorC.run(FORWARD);

digitalWrite(23, LOW);

digitalWrite(49, LOW);

digitalWrite(47, HIGH);

digitalWrite(51, HIGH);

delay(500);

digitalWrite(47,LOW );

digitalWrite(51, LOW);

delay(500);

}

}

void SM\_Backward()

{

sang\_flag = 0;

motorA.run(BACKWARD);

motorB.run(BACKWARD);

motorD.run(BACKWARD);

motorC.run(BACKWARD);

digitalWrite(47, LOW);

digitalWrite(51, LOW);

digitalWrite(23, HIGH);

digitalWrite(49, HIGH);

delay(700);

digitalWrite(23, LOW );

digitalWrite(49, LOW);

delay(700);

}

void SM\_Left()

{

sang\_flag = 0;

digitalWrite(23, LOW);

digitalWrite(47, LOW);

motorA.run(RELEASE);

motorB.run(FORWARD);

motorD.run(FORWARD);

motorC.run(RELEASE);

digitalWrite(49, HIGH);

digitalWrite(51, HIGH);

delay(700);

digitalWrite(49,LOW );

digitalWrite(51, LOW);

delay(700);

}

void SM\_Right()

{

sang\_flag = 0;

digitalWrite(49, LOW);

digitalWrite(51, LOW);

motorA.run(FORWARD);

motorB.run(RELEASE);

motorD.run(RELEASE);

motorC.run(FORWARD);

digitalWrite(23, HIGH);

digitalWrite(47, HIGH);

delay(700);

digitalWrite(23,LOW );

digitalWrite(47, LOW);

delay(700);

}

void SM\_Idle()

{

sang\_flag = 0;

motorA.run(RELEASE);

motorB.run(RELEASE);

motorD.run(RELEASE);

motorC.run(RELEASE);

digitalWrite(47, HIGH);

digitalWrite(49, HIGH);

delay(500);

digitalWrite(51, HIGH);

digitalWrite(23, HIGH);

delay(500);

digitalWrite(47, LOW);

digitalWrite(49, LOW);

delay(500);

digitalWrite(51, LOW);

digitalWrite(23, LOW);

delay(500);

}

void SM\_StartSing()

{

notePointer = 0;

motorA.run(RELEASE);

motorB.run(RELEASE);

motorD.run(RELEASE);

motorC.run(RELEASE);

}

void SM\_Sing()

{

if(notePointer < GOT\_Number)

{

tone(40, GOT\_NOTES[notePointer]);

delay(GOT\_LENGHTS[notePointer]);

noTone(40);

notePointer++;

}

else

CurrentState = Idle;

}

void loop()

{

digitalWrite(trigPin, LOW); //Clearing the trigger pin

delayMicroseconds(2);

digitalWrite(trigPin, HIGH); // Setting the trigger pin to output for 10 ms then set it back to low

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); //Read the echo pin that return the travel time of the pulse in ms

cm = microsecondsToCentimeters(duration);

if(analogRead(p2) > 150) // Reading the pin p2 (A9) related to backward motion

{

CurrentState = Backward;

}

else if(analogRead(p3)>150) // Reading the pin p3 (A10) related to left motion

{

CurrentState = Left;

}

else if(analogRead(p4) > 150) // Reading the pin p4 (A11) related to right motion

{

CurrentState = Right;

}

else if(analogRead(p5)>150) // Reading the pin p5 (A12) for stopping the motors

{

CurrentState = Idle;

}

else if(analogRead(p6) > 150) // Reading the pin p6 (A13) for singing

{

if(CurrentState != Sing && CurrentState != Singing)

{

CurrentState = Sing;

}

else

CurrentState = Singing;

}

else if(analogRead(p1) > 150) // Reading the pin p1 (A14) for fowrwad motion

{

CurrentState = Forward;

}

runCurrentState();

}

long microsecondsToCentimeters(long microseconds)

{ return microseconds / 29 / 2;

}

**Node MCU code:**

#include <ESP8266WiFi.h>

const char\* ssid = "AndroidAP6213";

const char\* password = "26101997";

int pin=D8;

int ledPin = BUILTIN\_LED; // GPIO13

int pin2=D7;

int pin3=D6;

int pin4=D5;

int pin5=D4;

int pin6=D3;

WiFiServer server(80);

String request="/STOP";

void setup() {

Serial.begin(115200);

delay(10);

pinMode(pin, OUTPUT);

pinMode(pin2, OUTPUT);

pinMode(pin3, OUTPUT);

pinMode(pin4, OUTPUT);

pinMode(pin5, OUTPUT);

pinMode(pin6, OUTPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

// Connect to WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

// Start the server

server.begin();

Serial.println("Server started");

// Print the IP address

Serial.print("Use this URL to connect: ");

Serial.print("http://");

Serial.print(WiFi.localIP());

Serial.println("/");

}

void loop() {

// Check if a client has connected

WiFiClient client = server.available();

if (!client) {

return;

}

// Wait until the client sends some data

Serial.println("new client");

while(!client.available()){

delay(1);

}

// Read the first line of the request

request = client.readStringUntil('\r');

Serial.println(request);

client.flush();

// Match the request

int value = 0;

if (request.indexOf("/FATA") != -1)

{

value = 1;

analogWrite(pin,200);

analogWrite(pin2,0);

analogWrite(pin3,0);

analogWrite(pin4,0);

analogWrite(pin5,0);

analogWrite(pin6,0);

}

if (request.indexOf("/STANGA") != -1)

{

value = 2;

analogWrite(pin,0);

analogWrite(pin2,0);

analogWrite(pin3,200);

analogWrite(pin4,0);

analogWrite(pin5,0);

analogWrite(pin6,0);

}

if (request.indexOf("/DREAPTA") != -1)

{

value = 3;

analogWrite(pin,0);

analogWrite(pin2,0);

analogWrite(pin3,0);

analogWrite(pin4,200);

analogWrite(pin5,0);

analogWrite(pin6,0);

}

if (request.indexOf("/SPATE") != -1)

{

value = 4;

analogWrite(pin,0);

analogWrite(pin2,200);

analogWrite(pin3,0);

analogWrite(pin4,0);

analogWrite(pin5,0);

analogWrite(pin6,0);

}

if (request.indexOf("/STOP") != -1)

{

value =5;

analogWrite(pin,0);

analogWrite(pin2,0);

analogWrite(pin3,0);

analogWrite(pin4,0);

analogWrite(pin5,200);

analogWrite(pin6,0);

}

if (request.indexOf("/CANTA") != -1)

{

value =6;

analogWrite(pin,0);

analogWrite(pin2,0);

analogWrite(pin3,0);

analogWrite(pin4,0);

analogWrite(pin5,0);

analogWrite(pin6,200);

}

//Html code for all the pages that construct the different states of the robot

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println(""); // do not forget this one

client.println("<!DOCTYPE HTML>");

client.println("<html>");

client.println("<style>"

"body { background-color: #363636;"

" background-image: linear-gradient(45deg, hsla(0,0%,0%,.25) 25%, transparent 25%, transparent 75%, hsla(0,0%,0%,.25) 75%, hsla(0,0%,0%,.25)), "

" linear-gradient(45deg, hsla(0,0%,0%,.25) 25%, transparent 25%, transparent 75%, hsla(0,0%,0%,.25) 75%, hsla(0,0%,0%,.25));"

" background-position:0 0, 2px 2px;"

" background-size:4px 4px;"

"}"

"button {"

"margin-left: 2cm;"

"}"

"button:hover,"

"button:active {"

" outline: 0;"

"}"

/\* 3D Button \*/

"button.depth {"

" background: #444;"

" border: none;"

" border-radius: 80px;"

"box-shadow: inset 0 0 2px 2px hsla(0,0%,0%,.2),"

" inset 0 0 2px 4px hsla(0,0%,0%,.2),"

" inset 0 0 2px 6px hsla(0,0%,0%,.2),"

" inset 0 0 1px 8px hsla(0,0%,0%,.5),"

" inset 0 1px 1px 8px hsla(0,0%,100%,.25),"

" inset 0 -30px 30px hsla(0,0%,0%,.2),"

" 0 -4px 8px 4px hsla(0,0%,0%,.5),"

" 0 10px 10px hsla(0,0%,0%,.25),"

" 0 0 2px 2px hsla(0,0%,0%,.2),"

" 0 0 2px 4px hsla(0,0%,0%,.2),"

" 0 0 2px 6px hsla(0,0%,0%,.2),"

" 0 0 2px 8px hsla(0,0%,0%,.5),"

" 0 1px 2px 8px hsla(0,0%,100%,.25),"

" 0 -1px 2px 8px hsla(0,0%,0%,.5);"

" color: #303030;"

" cursor: pointer;"

" font: bold 85px/85px sans-serif;"

" height: 300px;"

"width:400px;"

" padding:50px 70px;"

" text-shadow: 0 1px 1px hsla(0,0%,100%,.25),"

" 0 -1px 1px hsla(0,0%,0%,.75);"

"}"

"button.depth:focus {"

" color: #0ab;"

" text-shadow: 0 0 20px hsla(240,75%,75%,.5),"

" 0 1px 1px hsla(0,0%,100%,.25),"

" 0 -1px 1px hsla(0,0%,0%,.75);"

"}"

"button.depth:active {"

" box-shadow: inset 0 0 2px 2px hsla(0,0%,0%,.2),"

" inset 0 0 2px 4px hsla(0,0%,0%,.2),"

" inset 0 0 2px 6px hsla(0,0%,0%,.2),"

" inset 0 0 1px 7px hsla(0,0%,0%,.5),"

" inset 0 5px 15px 7px hsla(0,0%,0%,.15),"

" inset 0 -4px 2px 3px hsla(0,0%,0%,.5),"

" inset 0 1px 1px 7px hsla(0,0%,100%,.25),"

" inset 0 -30px 30px hsla(0,0%,0%,.1),"

" inset 0 30px 30px hsla(0,0%,0%,.2),"

" 0 -4px 8px 4px hsla(0,0%,0%,.5),"

" 0 5px 10px hsla(0,0%,0%,.25),"

" 0 0 2px 2px hsla(0,0%,0%,.2),"

" 0 0 2px 4px hsla(0,0%,0%,.2),"

" 0 0 2px 6px hsla(0,0%,0%,.2),"

" 0 1px 2px 8px hsla(0,0%,100%,.25),"

" 0 -1px 2px 8px hsla(0,0%,0%,.5);"

"line-height: 86px;"

"}"

"p{margin-left=4cm;}"

"</style>");

client.println("<br><br>");

client.println("<a href=\"/FATA\"\"><button class='depth'>&#8593;</button></a>");

client.println("<br><br>");

client.println("<a href=\"/STANGA\"\"><button class='depth'>&#8592;</button></a>");

client.println("</html>");

client.println("<a href=\"/DREAPTA\"\"><button class='depth'>&#8594;</button></a>");

client.println("</html>");

client.println("<br><br>");

client.println("<a href=\"/SPATE\"\"><button class='depth'>&#8595;</button></a>");

client.println("<br><br>");

client.println("<br><br>");

client.println("<br><br>");

client.println("<a href=\"/STOP\"\"><button class='depth'>STOP</button> </a>");

client.println("<a href=\"/CANTA\"\"><button class='depth'>CANTA</button> </a>");

delay(1);

Serial.println("Client disonnected");

Serial.println("");

}

**Music.h code :**

//Music defines

#define NOTE\_B0 31

#define NOTE\_C1 33

#define NOTE\_CS1 35

#define NOTE\_D1 37

#define NOTE\_DS1 39

#define NOTE\_E1 41

#define NOTE\_F1 44

#define NOTE\_FS1 46

#define NOTE\_G1 49

#define NOTE\_GS1 52

#define NOTE\_A1 55

#define NOTE\_AS1 58

#define NOTE\_B1 62

#define NOTE\_C2 65

#define NOTE\_CS2 69

#define NOTE\_D2 73

#define NOTE\_DS2 78

#define NOTE\_E2 82

#define NOTE\_F2 87

#define NOTE\_FS2 93

#define NOTE\_G2 98

#define NOTE\_GS2 104

#define NOTE\_A2 110

#define NOTE\_AS2 117

#define NOTE\_B2 123

#define NOTE\_C3 131

#define NOTE\_CS3 139

#define NOTE\_D3 147

#define NOTE\_DS3 156

#define NOTE\_E3 165

#define NOTE\_F3 175

#define NOTE\_FS3 185

#define NOTE\_G3 196

#define NOTE\_GS3 208

#define NOTE\_A3 220

#define NOTE\_AS3 233

#define NOTE\_B3 247

#define NOTE\_C4 262

#define NOTE\_CS4 277

#define NOTE\_D4 294

#define NOTE\_DS4 311

#define NOTE\_E4 330

#define NOTE\_F4 349

#define NOTE\_FS4 370

#define NOTE\_G4 392

#define NOTE\_GS4 415

#define NOTE\_A4 440

#define NOTE\_AS4 466

#define NOTE\_B4 494

#define NOTE\_C5 523

#define NOTE\_CS5 554

#define NOTE\_D5 587

#define NOTE\_DS5 622

#define NOTE\_E5 659

#define NOTE\_F5 698

#define NOTE\_FS5 740

#define NOTE\_G5 784

#define NOTE\_GS5 831

#define NOTE\_A5 880

#define NOTE\_AS5 932

#define NOTE\_B5 988

#define NOTE\_C6 1047

#define NOTE\_CS6 1109

#define NOTE\_D6 1175

#define NOTE\_DS6 1245

#define NOTE\_E6 1319

#define NOTE\_F6 1397

#define NOTE\_FS6 1480

#define NOTE\_G6 1568

#define NOTE\_GS6 1661

#define NOTE\_A6 1760

#define NOTE\_AS6 1865

#define NOTE\_B6 1976

#define NOTE\_C7 2093

#define NOTE\_CS7 2217

#define NOTE\_D7 2349

#define NOTE\_DS7 2489

#define NOTE\_E7 2637

#define NOTE\_F7 2794

#define NOTE\_FS7 2960

#define NOTE\_G7 3136

#define NOTE\_GS7 3322

#define NOTE\_A7 3520

#define NOTE\_AS7 3729

#define NOTE\_B7 3951

#define NOTE\_C8 4186

#define NOTE\_CS8 4435

#define NOTE\_D8 4699

#define NOTE\_DS8 4978

//Defines related to the singing part of the project

int melody[] = {

NOTE\_C4, NOTE\_G3,NOTE\_G3, NOTE\_A3, NOTE\_G3,0, NOTE\_B3, NOTE\_C4};

int noteDurations[] = {

4, 8, 8, 4,4,4,4,4 };

int sang\_flag = 1;

short int notePointer ;

short int GOT\_Number = 67;

short unsigned int GOT\_NOTES[] = {NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_E4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_E4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_E4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_E4,NOTE\_F4,

NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,NOTE\_D4,

NOTE\_G3,NOTE\_AS3,NOTE\_C4,NOTE\_D4,

NOTE\_G3,NOTE\_AS3,NOTE\_C4,NOTE\_D4,

NOTE\_G3,NOTE\_AS3,NOTE\_C4,NOTE\_D4,

NOTE\_G3,NOTE\_AS3,NOTE\_C4,NOTE\_D4,NOTE\_F4,NOTE\_AS3,NOTE\_DS4,NOTE\_D4,NOTE\_F4,NOTE\_AS3,NOTE\_DS4,NOTE\_D4,NOTE\_C4,

NOTE\_GS3,NOTE\_AS3,NOTE\_C4,NOTE\_F3,

NOTE\_GS3,NOTE\_AS3,NOTE\_C4,NOTE\_F3,

NOTE\_GS3,NOTE\_AS3,NOTE\_C4,NOTE\_F3,

NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,NOTE\_G4,NOTE\_C4,NOTE\_DS4,NOTE\_F4,NOTE\_D4};

short unsigned int GOT\_LENGHTS[] = {500,500,250,250,

500,500,250,250,

500,500,250,250,

500,500,250,250,

500,500,250,250,

500,500,250,250,

500,500,250,250,

500,500,250,250,

500,500,250,250,500,

500,250,250,500,

500,250,250,500,

500,250,250,500,

500,250,250,1000,1000,1000,250,250,1000,1000,250,250,500,

250,250,500,500,

250,250,500,500,

250,250,500,500,

1000,1000,250,250,1000,1000,250,250,500};

# 8.Bibliography

<http://www.geeetech.com/wiki/index.php/Arduino_Mega_2560>

https://www.toppr.com/bytes/principles-of-led/

<https://www.arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf>

www.fritzing.org

<https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>

<https://learn.adafruit.com/adafruit-motor-shield>

<http://wiki.sunfounder.cc/index.php?title=L293D_Motor_Driver_Shield>

https://en.wikipedia.org/wiki/ESP8266