Remote vehicle control, joystick, Arduino Wemos and communication ESP-NOW

Introduction:

Summary of work:
.- Several analog inputs through a single port.
.- WEMOS, electrical specifications.
.- Communication protocol ESP-NOW.
.- Circuit L298N. Specifications and pinout of it.
Mounting .- vehicle with two DC motors

In this article I explain how to take several analog values and place in a single port on a Wemos A0 plate. Values from a joystick, are transmitted quickly, securely and easily via Wifi using the ESP-NOW protocol. In the vehicle, another WEMOS receives the data and drives two DC motors for steering the vehicle.

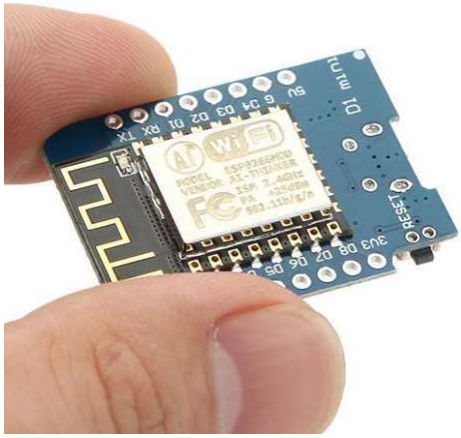
All part of the idea of moving a wheelchair for disabled staff remotely and to accompany them without pushing it. As a working example, I created this project. Later you can change the output circuits and motors with higher-power and coupled to the wheels of the chair a mechanical system that moves.

Perhaps someone can argue that things exposed these works can be achieved easily and cheaply in a web, but the fact of do it yourself and components low price is always a pleasure when you see it working. Other than that, I'll settle for a person like it or clarify a concept or doubt.

I will try to explain the concepts used for better understanding of the work. Perhaps some will find interesting any part of it.

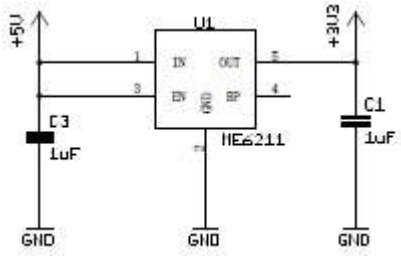
Arduino development board Wemos:

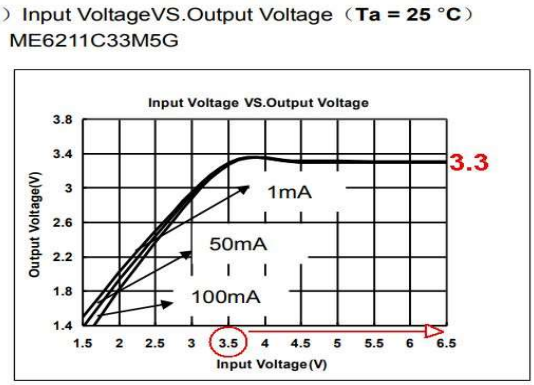
We are talking about a small development board with wide possibilities:



With it we can perform projector IoT, data analysis and delivery via networks and many other things, taking advantage of the wireless capabilities of the same. In another project I've done, I think own wifi network and I can open a remote lock, using a typed from our smartphone, I've also published key. The difference from the above is that instead of using HTLM protocol for communication, use very little published WiFi type communication ESP-NOW between two devices, being easy, quick, secure (encrypted) without feature pairings when acting (only when configuring the Arduino sketch). Later, when explaining the skit, I will discuss the details to consider.

The plate has a **entry**5v power to the pin (or USB) and input GND. Such feeding need not be 5v, and carrying a voltage regulator which makes 3.3v, which is really the working voltage. In the datasheet of Wemos we can see and also attached a picture of the regulator datasheet:



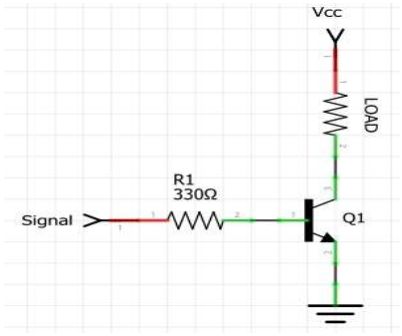


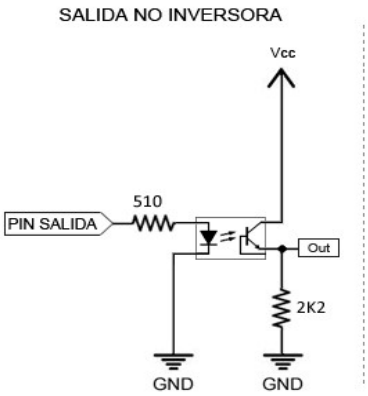
According to the link specification ESP8266, you could work even 3v, but should feed a voltage higher than 3.5V, so that the output of internal regulator have a minimum of 3v. In this link you can see other technical details that expand this information.

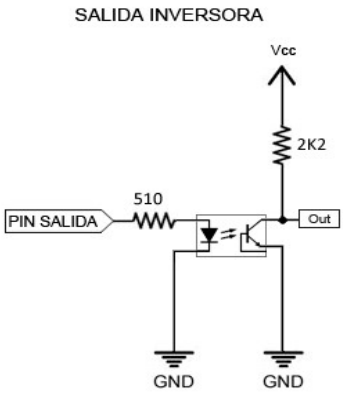
https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf

The plate also has 9 digital inputs / outputs (D0-D8). All have the ability to work with PWM outputs, I2C bus, etc.

Detail to take into account when connecting something to the digital output pins, to illuminate LEDs, activate relays, etc. The maximum current that can deliver a digital pin**is 12mA**. If you need to deliver more current, we must be inserted between the pin and a transistor device or an opto coupler higher power. For example:







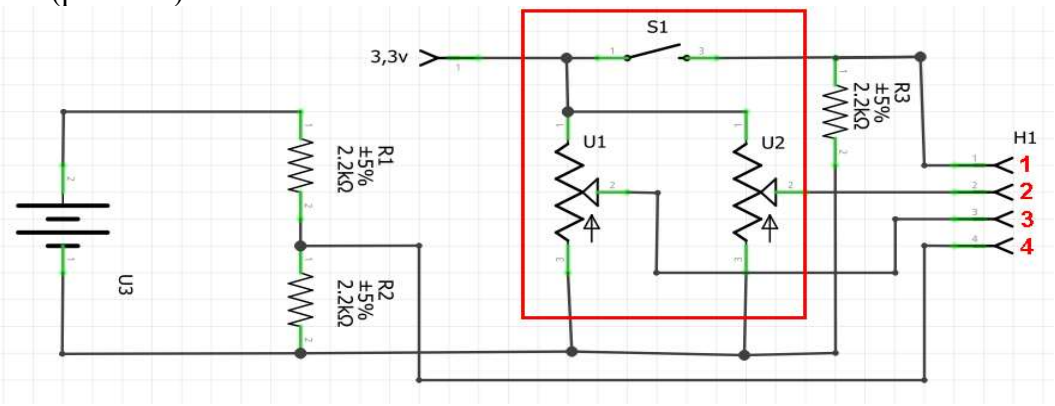
With a resistor in series with the output of 330 ohms, a current of 10mA is delivered, so if possible, increase the value of the resistors. There are many websites recommending a 330 ohm resistor in series with the LEDs I recommend using higher strengths. If the LED lights to our liking, we do not need to add any SAVINGS work mA power is always good.
NOTE: digital pins, we can give PWM values between 0 and 1023. Arduino Uno, between 0 and 254.
The WEMOS plate also has a digital input A0, to analog data analysis. Must take into account two things. The first is that NO can be applied more than 3.3V voltage directly, as it would deteriorate. If you want to measure higher voltage must be inserted external voltage divider. Said input values are 0 to 1,024.
Other features:
-Departure 3.3V to power external circuits. 12mA maximum current per pin.
-Connector micro USB for firmware load and 5v power
-Pulsador Reset.

There are many tutorials on how to setup the Arduino IDE to work with this type of plaque as well as the necessary libraries. I will not go into it too much not to extend this work.

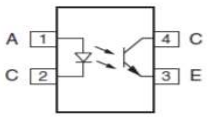
Joystick circuit (remote control):

I like the development board Wemos, as it has little size, it is cheap and has many possibilities. As only has an analog input A0, the problem of wanting to capture several analog values simultaneously arises. For my case in particular a Joysick consists of two potentiometers with analog individual outputs and a push. In addition, I want to analyze the current value of the battery used in the remote control, so now we need to take 3 different analog values.

In the following scheme, Created with Fritzing, we left a voltage divider. If the battery is more than 3.3V, the analog input is at risk of damaged therefore appropriate to reduce the voltage for analysis. I use 3.7v battery, so when loaded completely is approximately 4v and due to the voltage divider in the H1 pin 4 have 2v (variable depending on battery status). To the right is a basic joystick, consisting of two potentiometers and a button (R3 is external to the joystick). They feed with 3.3v providing WEMOS. In this general scheme, we have three analog values (pins 2, 3 and 4 H1) and a digital value (pin 1 H1).

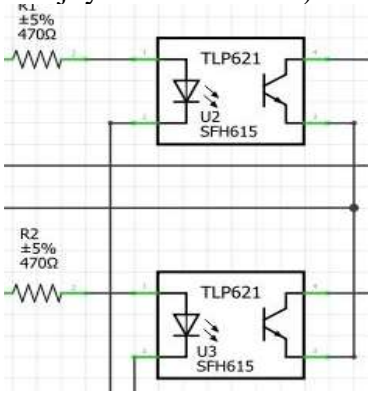


To analyze the plaque WEMOS 3 analog values, we use a small optocouplers, the chip SFH615A or TLP621. Operation is very basic for this work. In the pin 4 of chip I put one of the analog values to be analyzed. All pin 2 to GND. All pin 3 connected and A0 and each pin 1 to a digital output through a resistor, which will activating successively and depending which activate and reading the value A0, I assign each value variable (pot 1y pot 2 joystick and drums).



DESCRIPTION

The SFH615A feature a variety of transfer ratios, low coupling capacitance and high isolation voltage. These couplers have a GaAs infrared diode emitter, which is optically coupled to a silicon planar phototransistor detector, and is incorporated in a plastic DIP-4 package.



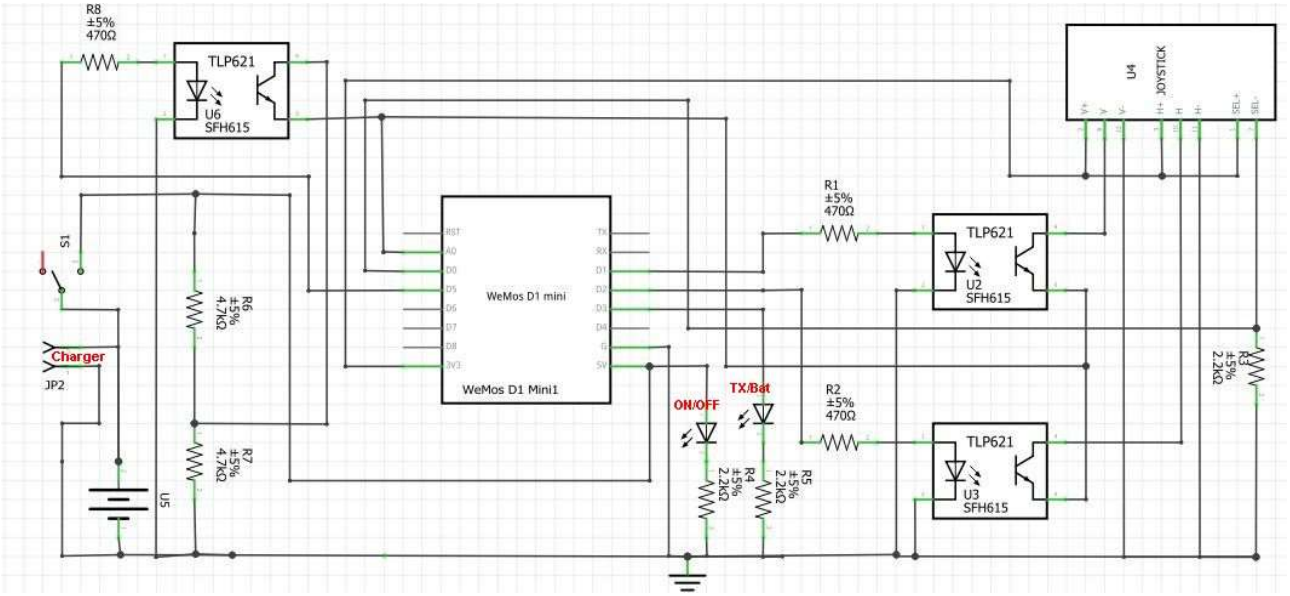
We must bear in mind that we can not connect the digital output of the Wemos directly to pin 1 of the TLP621, since this digital output will deteriorate. Each digital pin can supply about 12mA Wemos. Therefore we intersperse sufficient to activate the internal LED resistance. 470Ω It is sufficient to activate and requires only 7 mA.

Wanting to enter 3 analog values using this system, we use 3 digital outputs to activate them. If we want to introduce more analog values A0, we can use other digital outputs more or we can continue to use only 3 digital outputs, adding a demultiplexer circuit and giving binary values to the inputs,

we get up to 8 possible digital values.

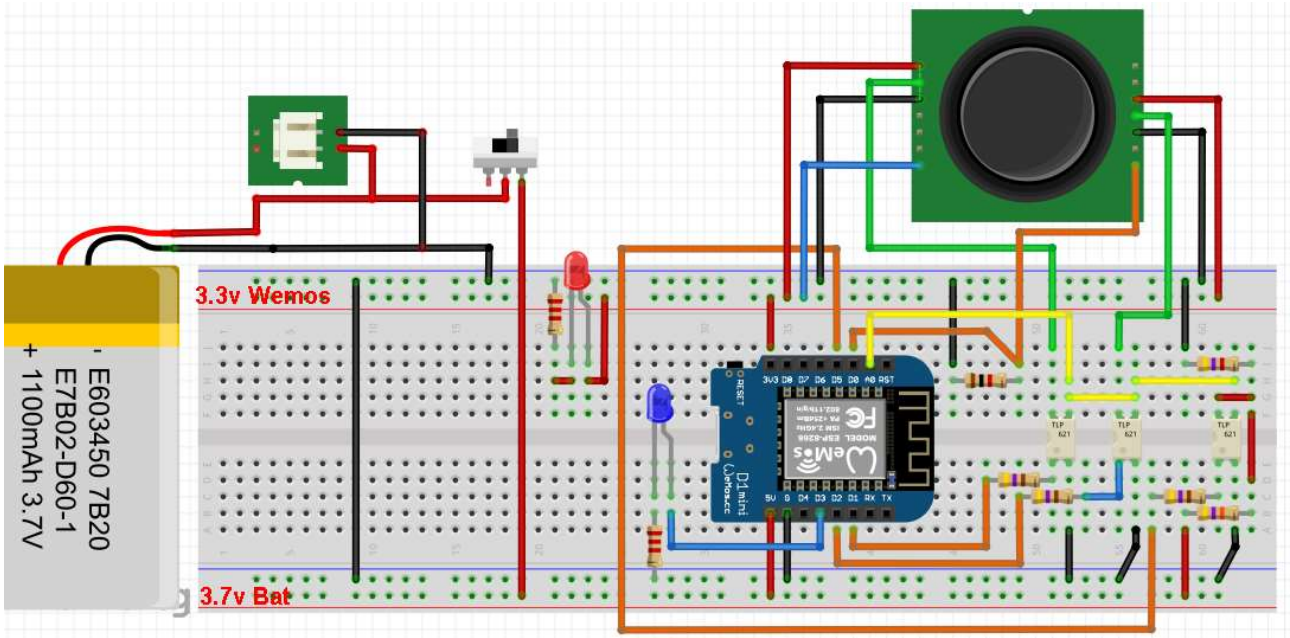
Add the remote control 2 LEDs, one to reflect "Power ON" and the other for battery status and "Transmission OK".

Add a switch circuit for battery and a connector to recharge it without having to remove it (warning: OFF TO RECHARGE to avoid damaging the regulator ME6211 of Wemos plate). With everything explained above, the complete circuit of the remote control joystick is as follows:



Explanation for further development in the Arduino IDE:
A0 I pick the values of the potentiometers and battery level.
In D0 goes HIGH when the joystick button ("STOP") is pressed
If active D1, read the state of the vertical potentiometer joystick in A0.
If active D2, read the status of the horizontal potentiometer joystick in A0.
If active D5, read the battery status in A0. NOTE: Initially I put it in D4, but I did not trouble flashing the program from the Arduino IDE, so I went to D5
The output D3 is used for the Activity LED (blue). Said LED lights when no movement of joystick and the transmission was successful. When at rest indicates the state of the battery (1 flicker between 3.6 and 3.5V, 2 flashings between 3.5 and 3 flashes 3.4vy below 3.4v).
The red LED indicates Power On / Power ON.
S1 is the ignition switch. I should have it off when the battery charge is done or make modifications to the software (via USB 5v).

The scheme mounted on a breadboard circuit is as follows:

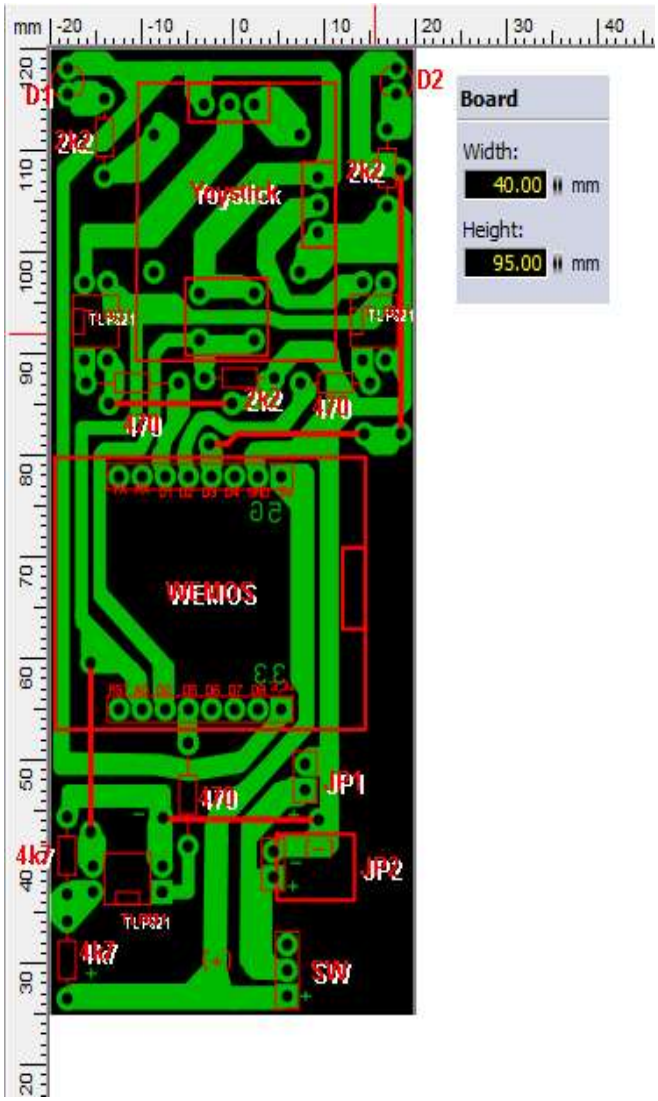


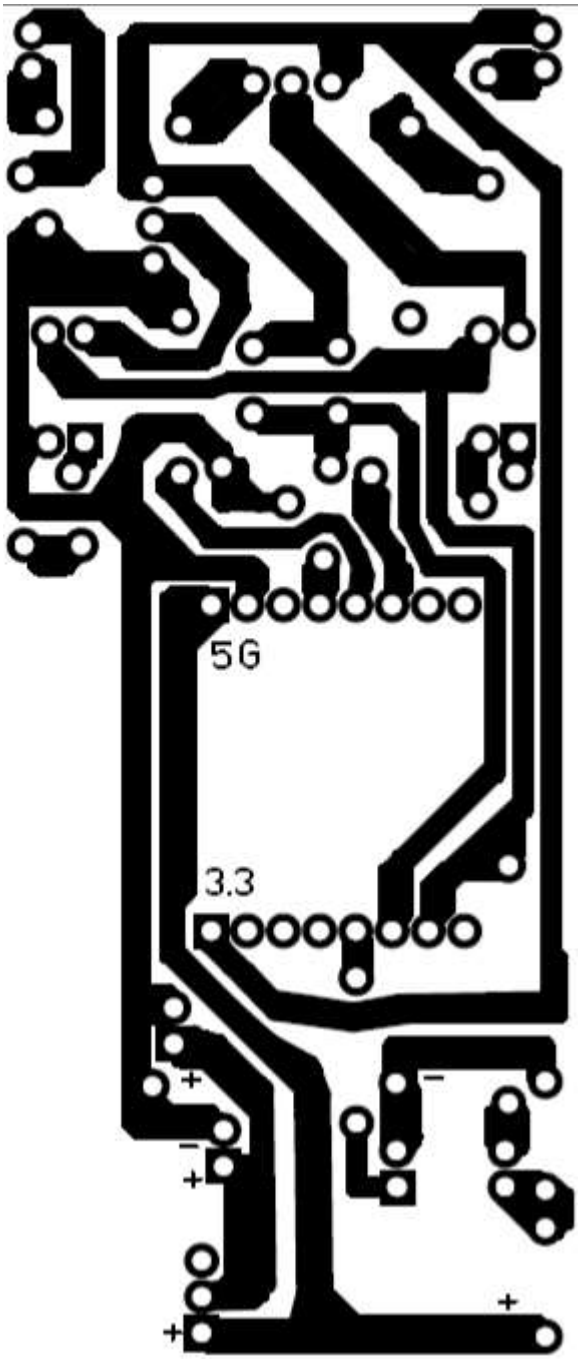
The bottom line is the positive battery voltage. The upper line is positive output of 3.3v WEMOS

I designed the following circuit board with Sprint-Layout 6.0 for connecting the joystick, opto couplers, Wemos and others. Indian measures in case anyone wants to do (40x95mm).

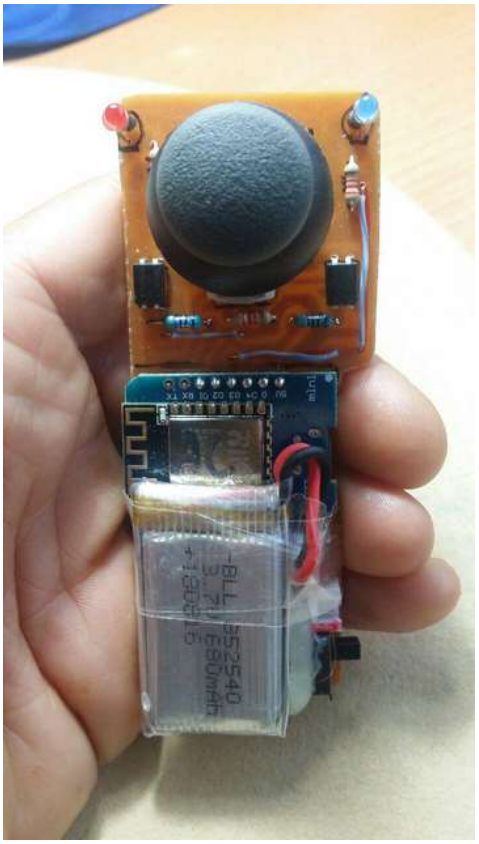
Must be careful with the pin 1 of the TLP621. They are welded to the square terminal and in the position shown seen from the side of the components.

The part of the next plate to connectors and WEMOS, the cut out later, it is so comfortably grip the knob, the ignition and external connections.





Photos of the remote control. At the edges, the USB connection, the charging connector on the battery and switch ON / OFF.
Easy to hold, albeit a little big. I need to make a case for himself as the 3D printer:

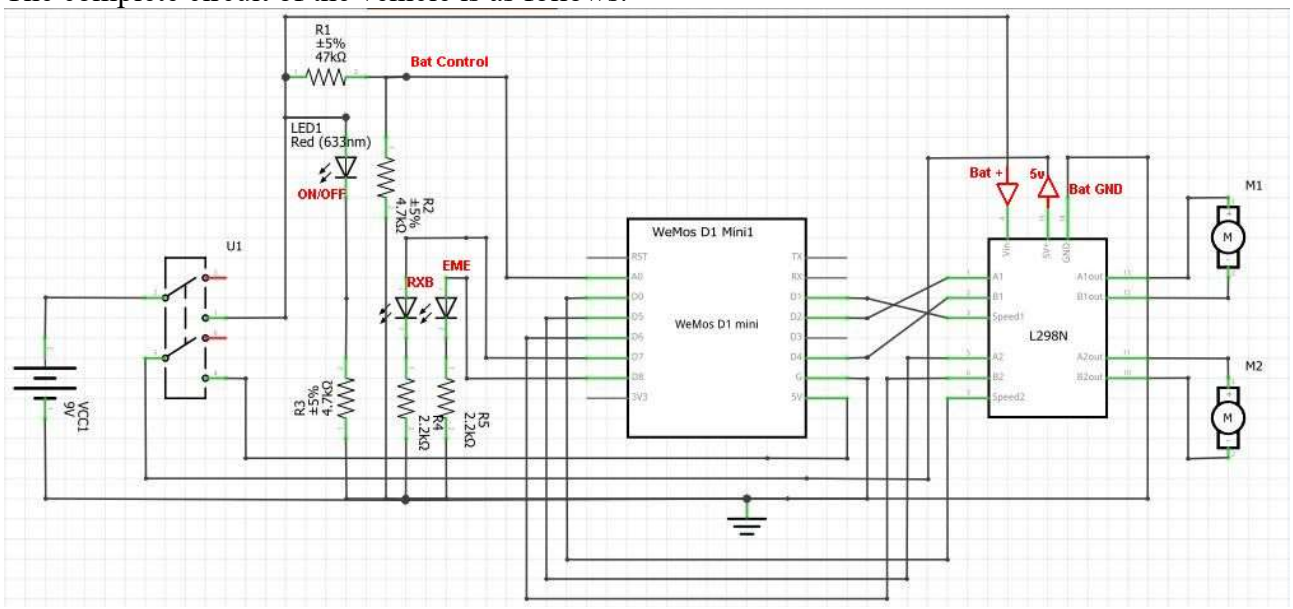


Receiver circuit (Motors):

It comprises another plate WEMOS where the data receipt joystick or remote control activates the necessary (dual H-bridge) signals to a control L298N and two motors, forward and backward, steering control. Complementing the circuit, 3 LEDs, one for power ON, the other for data transmission and a third as indicative of "emergency stop". I take the latter two (flashing) for indicating the status of the vehicle battery.

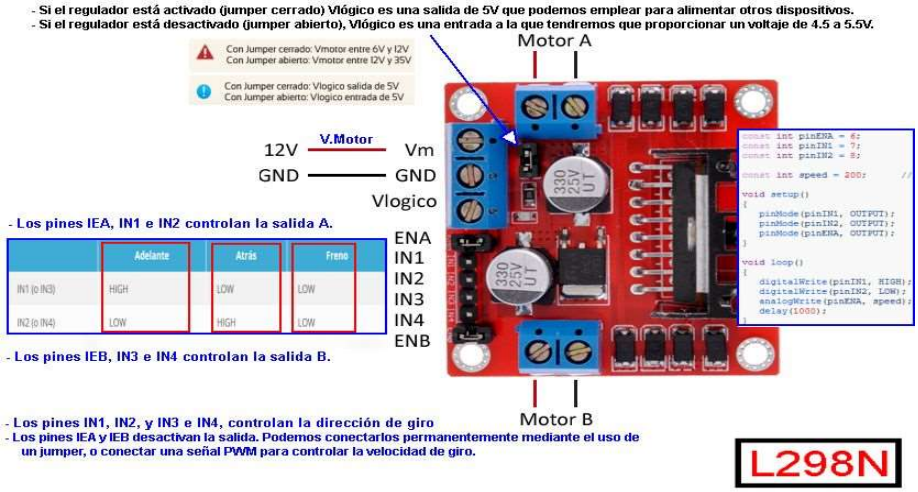
Control of battery status: The first thing to consider is that the battery I am using is 9v. Try to measure it in A0 directly, is damaging the port, since the maximum value that can be applied is 3.3V. To avoid this, we also another voltage divider, this time more unbalanced than in the remote control and reduce the value in A0. In this case, I use a 47k resistor in series with another 4K7. The focal point is where the reference volume to be measured. "Low Battery" between 7V and 5.5V,1 flashLED "Emergency". "Very Low Battery" (below 5.5V,3 flashings LED "Reception ok")

The complete circuit of the vehicle is as follows:



Because this circuit is mounted on a vehicle, I did not want to complicate a lot the Arduino sketch. Simply receives data via wireless joystick ESP-NOW and converts them into control signals for the motors. That makes it easier in future software changes or modifications path, be made only on the remote control (joystick) instead of both.
I have not made any special circuit board. Only a provisional for LEDs and resistors.

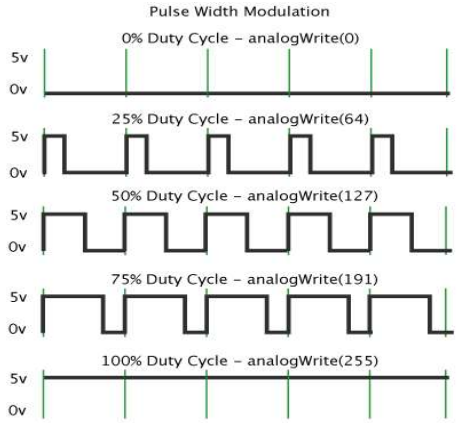
L298N (double H-bridge)



This is a short description of the circuit that controls the DC motor that drives the vehicle.
- Connectors A and B (blue 2 pin). They are the current outputs to the motors. If after testing, the motor rotates the opposite side we want, simply invert pins thereof
Power connector (blue 3-pin). It is the input current to the circuit. As it can be fed between 6 and 36 volts, one must take into account the jumper or bridge next to the connector.If fed with a voltage from 6 to 12V, the bridge is left in place and have a Vlogico **exit** 5v towards WEMOS (as in this study). If the circuit is powered by more than 12V voltage, you must remove the bridge so that the DC-DC converter leads is not damaged and if we want to run your logic circuitry, we take a cable 5v external to the circuit (5v input).In my case, as I use a 9v battery, I leave it since and I used to feed the Wemos board through the 5v pin. GND is negative battery and will also G Wemos and LEDs.

Control connector (6-pin). It has two parts. ENA, IN1, IN2 control the motor connected at A and ENB, IN3, IN4 controlling motor connected in Table B. In the figure above levels of the signals must have to set in motion the engines is indicated below , back or braking. ENA and ENB are some bridges. If we leave posts,L298N engines will input voltage Vm in the indicated direction, no speed control and voltage regulation. If we remove, we will use these pins for receiving a PWM signal from the WEMOS plate and thus control the speed of each motor.Arduino is achieved by a analogWrite () command. Wemos on the plate, all the D port have that capability.

In the figure L298N there is a box with a small sketch for **Arduino UNO**Which will rotate the motor A forward voltage near 75% of Vm.



The text before this graphic explains the relationship of analogWrite () with how output pins to Arduino UNO. In WEMOS, 100% is achieved with analogWrite (1023) and 50% would analogWrite (512).

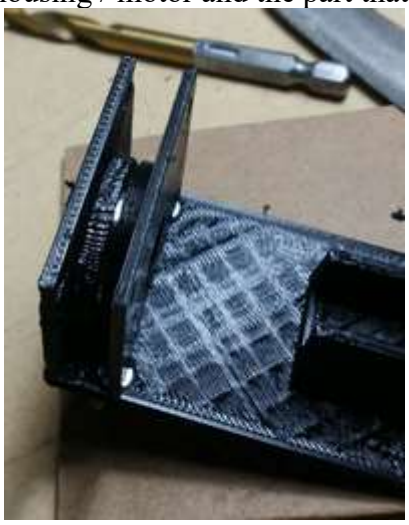
When this project, we must take into account possible ENA and ENB PWM values supplied by the analogWrite command because depend on the value of the battery voltage and voltage motors. In this case I use a 9V battery (Vm) and 6V engines. As you increase the PWM signal on them, the motor voltage rises, but does not begin to move until it reaches a certain value, so that in tests,***It must be set that minimum PWM to do the move***at low speed. On the other hand, if we put the PWM signal to the maximum, we give the motor voltage Vm battery (9V) and can damage the same, so in testing, we measure the voltage and***establish the maximum PWM so that it does not deteriorate***and as much provide the maximum 6V. Both, as I said earlier, in the Arduino sketch of the remote control.

Vehicle Mount:

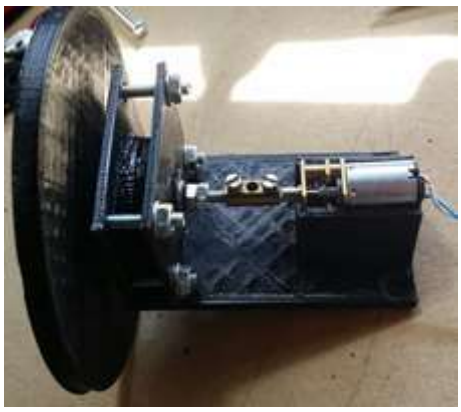
I have to admit that the assembly is a bit domesticated, but effective. Maybe 3D design and print a nicer model, but this model "home" has the advantage of better see the operation. There are a number of engines with gearbox and wheels included for coupling at low prices. I used what I have on hand.

For assembly, I have 3D printed pieces, wheels, bearing flange / motor and bushings and use 3mm diameter screws to join the pieces. For binding to the screw shaft motor, I used the contacts of a strip electrical connection by cutting the outer plastic. When mounting the wheels, the screw should stick to the wheel to prevent skidding when turning.

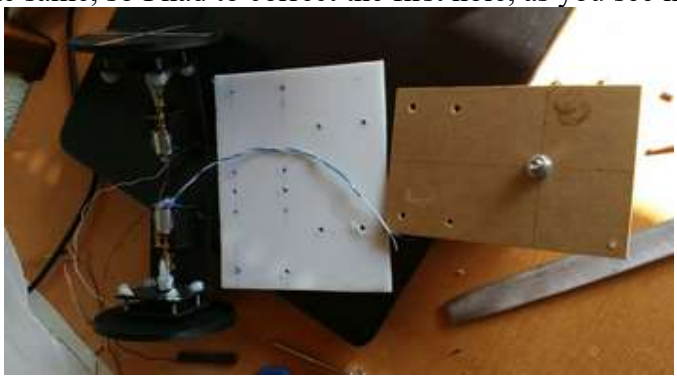
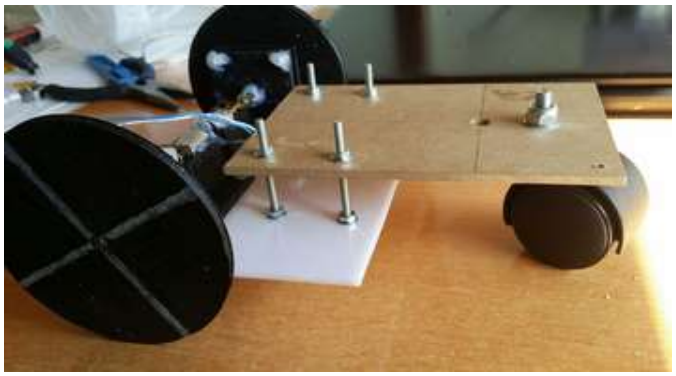
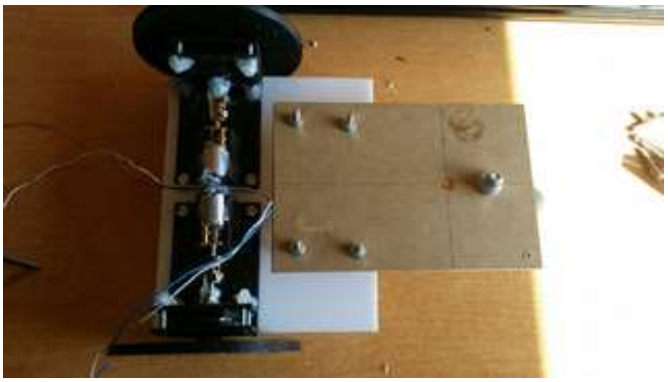
The following shows the bearing housing / motor and the part that holds 3D



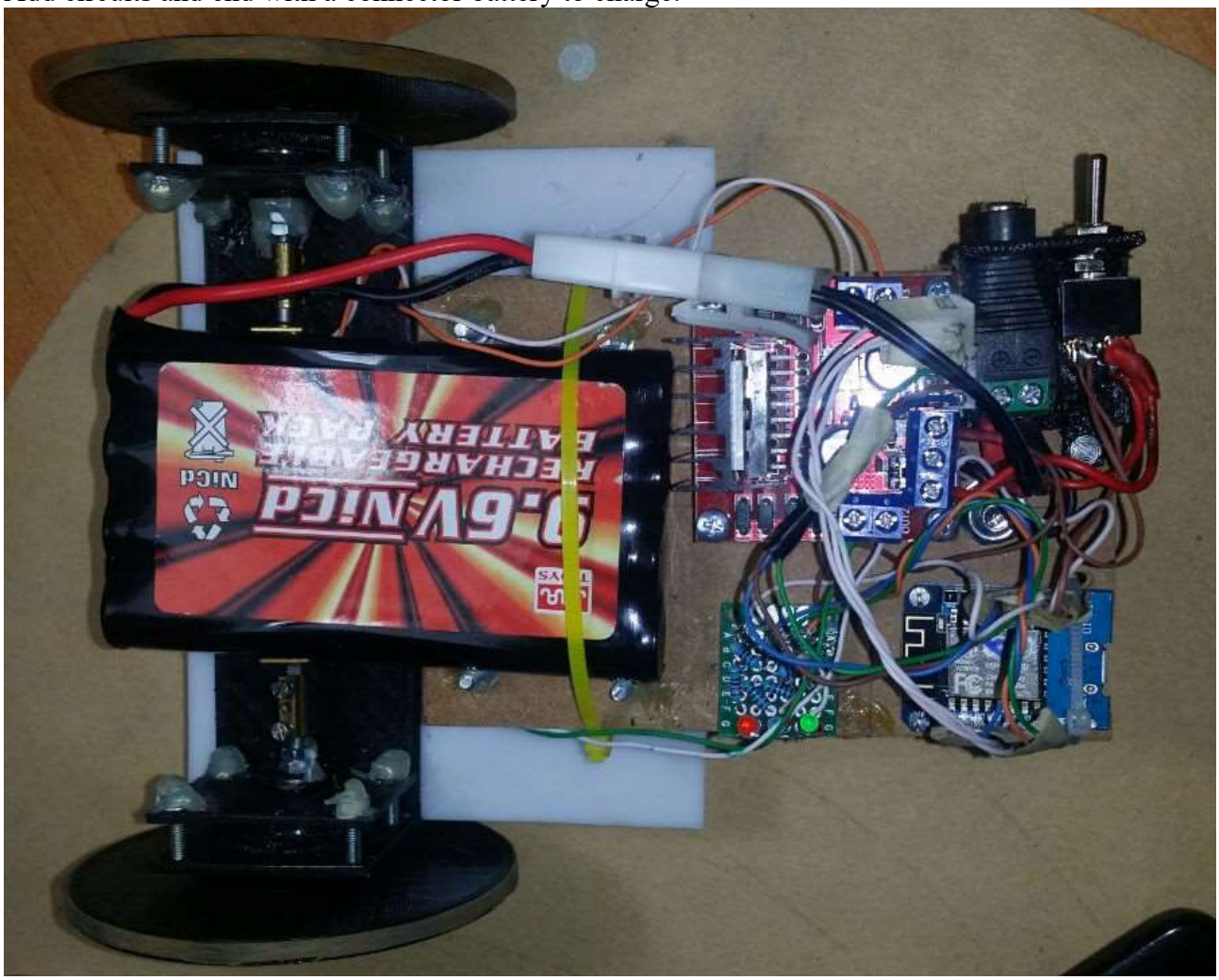
Amount wheel. Take measures, short screw and one is left:

Once completed the assembly of the two motor sets, attached to a platform (white). You one other platform to support circuit and the rear wheel. The height difference brand type wheel say, to maintain the horizontal vehicle. The distance between the rear wheel and the first platform we must ensure the rotation of the same, so I had to correct the first hole, as you see in the pictures.


Add circuits and end with a connector battery to charge.

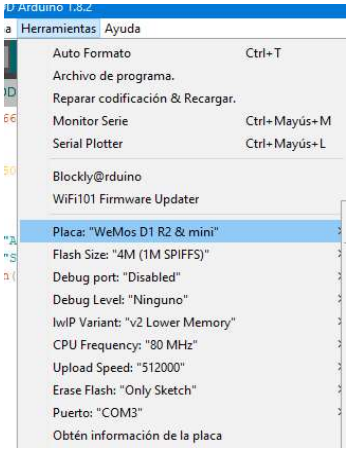


As you can see, it's not a great design. I intend to apply this system to a wheelchair as I said at the beginning of this work. But since I have it developed, possibly design a more elegant type of vehicle.

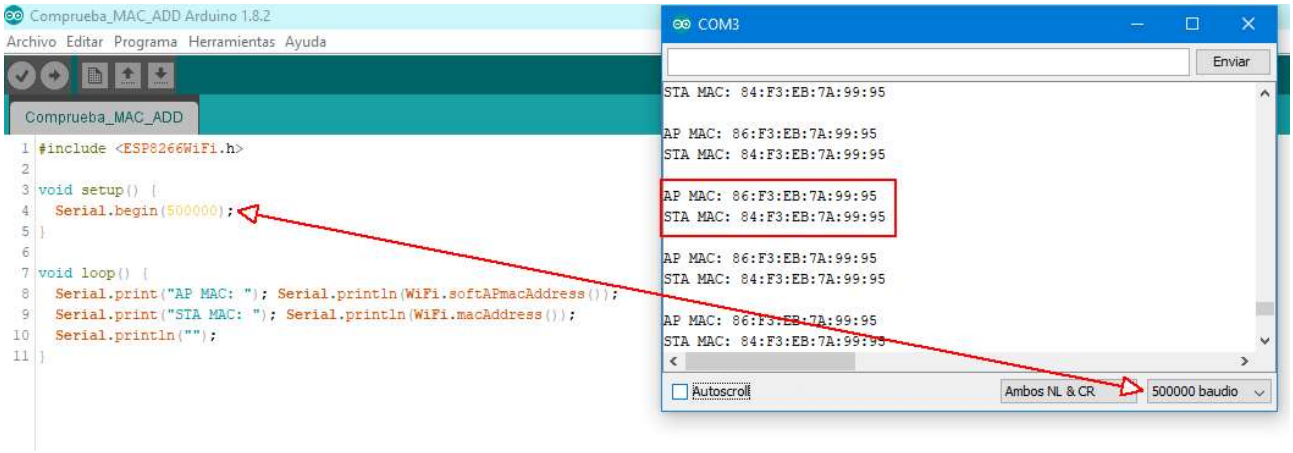
And now we come to the explanation of Arduino sketch I made.

Arduino

As I wrote at the beginning, I can not dwell much and I waive how to configure the Arduino IDE, libraries and how it should recognize the Wemos plate to work with them. Only a few details:
.- In Preferences Manager additional URLs:
http://arduino.esp8266.com/stable/package_esp8266com_index.json
.- In Tools (Tools), Manager cards:



As a preliminary and essential step before working with the ESP-NOW protocol, we charge this little skit in Wemos with which we will work to find out the MAC of AP ESP8266 leading integrated. Tools, Series Monitor can see the result of the sketch and note especially of each plate Wemos AP



I tend to buy on receiving that frame the bags and board with the data:

Once the AP MAC plates, start talking about the ESP-NOW protocol developed by Espressif:

"ESP-NOW allows a direct, low-power intelligent lights control without the need for a router. This method is energy efficient and convenient.
ESP-Now is another protocol developed by Espressif, which allows multiple devices to communicate with each other without using Wi-Fi. The protocol is similar to the low-power wireless connectivity 2.4GHz often deployed in wireless mice. Therefore, pairing between devices is required before disclosure. Once the match is made, the connection is secure and equal terms, without needed a handshake. "

More information in the link:
https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/network/esp_now.html
ESP-NOW is a spacious and with many possibilities protocol, but want to show you an easy way to communicate two devices and transfer data between them without using complex shapes.

The sketch I have prepared only one device transmits (joystick) and another receives data (vehicle). But both must have necessarily common things, which I describe.
.- Home Seller ESP-NOW

```
//Iniciamos la libreria de ESP-NOW
#include <ESP8266WiFi.h>
extern "C" {
#include <espnow.h>
}
```

.- The structure of data to transmit / receive. We can not define variables with variable length, but fixed length, because when all the data at once, which receives should know how to separate each byte received and know that variable value assign those bytes received are transmitted. It's like when a train is prepared with different station wagons and receiving them you must know how and that company should go. I want to transmit data at a time 5, If I press the joystick, and voltages (Left and Right motor) and direction (forward / backward) of each motor vehicle, which draw from the position thereof.

```
struct Data_ESP_NOW {
  uint8_t Pulsador;
  uint16_t VoltajeMI;
  uint16_t VoltajeMD;
  uint16_t SentidoMI;
  uint16_t SentidoMD;
};
```

.- I define the type of function performed every WEMOS. Perhaps because of the lack of experience in the ESP-NOW protocol, I had some problems when I define one as the master and the other as slave. It has always worked for me as putting the two-way (Role = 3)

```
  //0=sin función, 1=MAESTRO, 2=ESCLAVO y 3=MAESTRO+ESCLAVO
  esp_now_set_self_role(3);
```

.- Pairing the devices. In the skit joystck I put the AP MAC Wemos vehicle. In the sketch of the vehicle, I put the AP MAC joystick.

```
  uint8_t mac_addr[6] = {0x86, 0xF3, 0xEB, 0xB7, 0x32, 0x09};
  uint8_t role = 3;
  uint8_t channel = 3;
  //si la longitud es cero y la matriz vacia, no hay clave
  uint8_t key[12] = {0x3A, 0x2B, 0x78, 0x03, 0x77, 0x58, 0x86, 0xF3, 0xEB, 0xB7, 0x30, 0xFA};
  uint8_t key_len = sizeof(key);
  esp_now_add_peer(mac_addr, role, channel, key, key_len);
```

AP MAC

.- Sending data to the vehicle, following figure. First you have to prepare these train cars to be sent (data) with red box. Then you have to define who sent it (da)**which it is the AP MAC Wemos vehicle**and the total length of TREN. Once these above data, the data packet (green box) is sent.
Remember: I want to transmit data at a time 5, If I press the joystick, and voltages (Left and Right motor) and direction (forward / backward) of each motor vehicle.
After shipment, I verify that the vehicle has received the data correctly (blue box).

```
void transmision() {
  //***ENVIO DE LOS DATOS a uno ya emparejado previamente
  //Serial.print("Entro en transmision().  "); Serial.println(Comando);
  //***ENVIO DE LOS DATOS a uno ya emparejado previamente
  // Extructura del tren de datos a enviar
  Data_ESP_NOW TREN;
  TREN.Pulsador = Pulsador;
  TREN.VoltajeMI = VoltajeMI;
  TREN.VoltajeMD = VoltajeMD;
  TREN.SentidoMI = SentidoMI;
  TREN.SentidoMD = SentidoMD;
  uint8_t da[6] = {0x86, 0xF3, 0xEB, 0xB7, 0x32, 0x09};
  uint8_t data[sizeof(TREN)]; memcpy(data, &TREN, sizeof(TREN));
  uint8_t len = sizeof(data);
  esp_now_send(da, data, len); //envio de datos
  //***VERIFICACION DE LA RECEPCION CORRECTA DE LOS DATOS POR EL ESCLAVO***//
  esp_now_register_send_cb([](uint8_t* mac, uint8_t status) {
    //char MacTX[6];
    //sprintf(MacTX, "%02X:%02X:%02X:%02X:%02X:%02X", mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
    //Serial.print(". Enviado a ESP MAC: "); Serial.println(MacTX);
    //Serial.print(". Recepcion (0=OK - 1=ERROR):    "); Serial.println(status);
    TXgood = status;   // 0=OK
  });
}
```

.- Receiving data in the vehicle. This is the function I used in the Wemos the vehicle. As you can be seen the wear mode reception (reply, call back) and the received data assigned to variables (rail cars) with the same structure used in both:

```
void recepcion() {
  esp_now_register_recv_cb([](uint8_t *mac, uint8_t *data, uint8_t len ) {
    //char MACrecibida[6];
    //sprintf(MACrecibida, "%02X:%02X:%02X:%02X:%02X:%02X", mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
    //Serial.print("Recepcion desde ESP MAC: "); Serial.println(MACrecibida);
    Data_ESP_NOW TREN;
    memcpy(&TREN, data, sizeof(TREN));
    Pulsador = TREN.Pulsador;
    VoltajeMI = TREN.VoltajeMI;
    VoltajeMD = TREN.VoltajeMD;
    SentidoMI = TREN.SentidoMI;
    SentidoMD = TREN.SentidoMD;
  });  //Ojo con esos dos símbolos. Cierran la recepción
}
```

And just with this, I can transmit / receive data via wireless ESP-NOW easily.

In the next step will be a description of Arduino sketch remote control (joystick)

Arduino (Joystick)

.-After defining the library ESP-NOW, I define the pins that will use the Wemos:

```
// Pines de la placa
#define inA0 A0  //Entrada analogica del joystick (ambos potenciometros, Pot1 y Pot2)
#define inD0 16  // Entrada digital de la posicion del pulsador del joystick
#define outD1 5  // Salida digital para activar la lectura de Pot1
#define outD2 4  // Salida digital para activar la lectura de Pot2
#define outD5 14  // Salida digital para activar la lectura de Medida de bateria D4
#define outTXG 0 // Salida de led de TX Good D3
```

I define the variables .-'ll use later:

```
// Definicion de variables. Las 5 primeras son las que transmito.
byte Pulsador = 0;  // Variable del valor del pulsador del joystick
int VoltajeMI = 0; // PWM hacia el L298N motor izqierdo
int VoltajeMD = 0; // PWM hacia el L298N motor derecho
int SentidoMI = 0; // Hacia donde gira el motor izquierdo (0=parado, 1=adelante, 2=atrás) al L298N
int SentidoMD = 0; // Hacia donde gira el motor derecho (0=parado, 1=adelante, 2=atrás) al L298N
int Pot1 = 0; // variable analogica donde se almacena el valor de posicion horizontal en mi circuito
int Pot2 = 0;  // variable analogica donde se almacena el valor de posicion vertical en mi circuito
int Bat = 0;  // variable analogica donde se almacena el valor del voltaje de la bateria
// Los valores de los potenciometros los paso a valores entre 0 y 1023 (Wemos) y con su direccion.Indican el mov:
int IZQ = 0;  // Izquierda
int DER = 0;  // Derecha
int DEL = 0;  // Hacia adelante
int ATR = 0;  // Hacia atras
byte TXgood = 0;  //Si transmite correctamente tiene valor 0
byte ContadorReposo = 0;  // Acumula e indica cuantos bucles de espera de movimiento, para analizar el estado de
byte Movimiento = 0; // 1 indica que el joystick se mueve
int MinMotor = 140; // PWM minimo en la salida para que arranquen los motores
int MaxMotor = 450; // PWM maximo en la salida para maximo voltaje en los motores. Junto con MinMotor, dependen
```

.- Already in setup (), in the first part, I define as they go to work the pins of the Wemos and an initial value thereof. Also verify that the ESP-NOW protocol is initialized well. And after that, I define the working mode and previously commented pairings:

```
void setup() {
  //Serial.begin(500000);
  pinMode(inD0, INPUT);
  pinMode(outD1, OUTPUT); digitalWrite(outD1, LOW); // Apago las lecturas en los optoacopladores
  pinMode(outD2, OUTPUT); digitalWrite(outD2, LOW);
  pinMode(outD5, OUTPUT); digitalWrite(outD5, LOW);
  pinMode (outTXG, OUTPUT); digitalWrite(outTXG, LOW);  // Cuando transmite OK, se enciende

  //  Inicia protocolo ESP-NOW si no lo está
  if (esp_now_init() != 0) {
    //Serial.println("Protocolo ESP-NOW no inicializado...");
    ESP.restart();
    delay(10);
  }
  else {
    //Serial.println("Protocolo ESP-NOW INICIALIZADO --OK--...");
  }
```

.- Start the loop () with a delay that makes us the number of transmissions or joystick readings I want to do per second (see figure below). I put 60 msg, so I make about 15 readings per second or less. After I read the state of emergency button. If pressed, I put zero values engines, transmit and establish a delay which does not respond to anything until you pass the time (in my case 5 seconds delay (5000)).

```
void loop() {
  delay(60);  // Retardo entre tomas de datos de ambos potenciometros (sincronizar en el esclavo mas o menos)

  VoltajeMI = 0; VoltajeMD = 0; SentidoMI = 0; SentidoMD = 0; //Los valores  los pongo a 0

  Pulsador = digitalRead(inD0);  // Lee estado del Pulsador, por si hay emergencia
  if (Pulsador == 1) {  // Si se pulsa, envia el estado y el esclavo paraliza los motores durante 5 segundos (se puede varia
    transmision();  // Transmite que se ha pulsado "Emergencia"
    Pulsador = 0;
    if (TXgood == 0) { // Si se pulsa y ha transmitido ok, se mantiene encendido durante el retardo y bloquea el joystick
      digitalWrite(outTXG, HIGH);  //Se enciende el led si la transmision es correcta
    }
    else {
      digitalWrite(outTXG, LOW);
    }
    delay(5000);   // Fijo el retardo que quiero en estado de "Parada de emergencia" (en transmisor y receptor igual)
  }
```

.- The rest of the loop () are calls to functions that I use, which later explain.

```
  //      Si no está pulsador el switch, lee los potenciometros
  leePots();  // Realiza la lectura del potenciometro horizontal/vertical del joystick y del estado de la bateria (Pot1, Pot2 y Bat)

  ajustePots(); // Los valores de los Potenciometros horizontal/vertical, los transforma en valores hacia adelante/atras y derecha/izquierda (DEL, ATR, IZQ y DER)

  dirMot();  //Los valores DEL, ATR, IZQ y DER, los transformo en voltajes PWM de motores y el sentido de giro (VoltajeMI, VoltajeMD, SentidoMI y SentidoMD)

  // SI SE DESVIA EL VEHICULO, CORRIJO LA TRAYECTORIA AQUI antes de transmitir
  VoltajeMD = VoltajeMD + 50;  // en recto, se me desvia hacia la derecha, por lo que aumento el voltaje del motor derecho.
  transmision();

  if ((TXgood == 0) && (Movimiento == 1)) { //si la transmision ha sido correcta y hay movimiento
    digitalWrite(outTXG, HIGH);  //Se enciende el led
    ContadorReposo = 0;
    //Serial.print("Transmision correcta y Movimiento: TXgood=0...."); Serial.println(TXgood);
  }
  else {
    digitalWrite(outTXG, LOW);
    //Serial.print("ERROR al Transmitir o sin Movimiento: TXgood=1....."); Serial.println(TXgood);
  }

  ControlBat();
  // FIN de LOOP

}
```

.- Leo the status of potentiometers and battery. leePots (); . Delays (delay) that put the 5msg are for optocouplers readings are accurate. It should be borne in mind that since the LED is activated, it takes a few microseconds (about 10) to stabilize the output, so I put five msg to the readings are more accurate. This delay could be down perfectly.

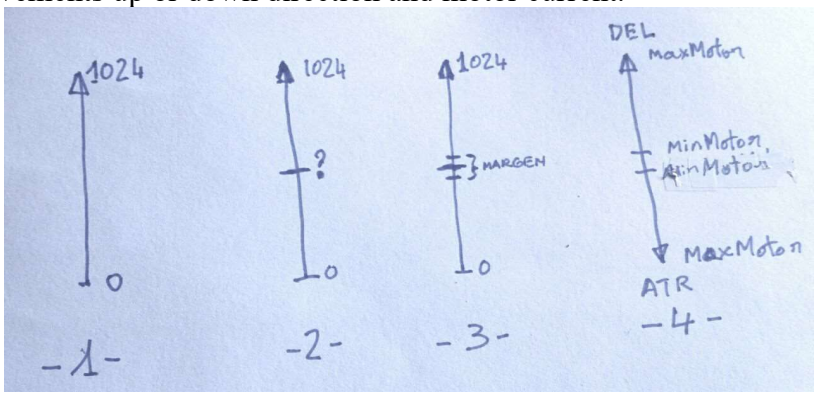
```
void leePots() {
  // Activa D1 y lee el valor de P1 del Joystick (Izquierda/derecha en mi circuito)
  digitalWrite(outD2, LOW);  //Asegura que solo lee Pot1
  digitalWrite(outD5, LOW);
  digitalWrite(outD1, HIGH);
  //Serial.println("Activado D1 ");
  delay(5);  // Retardo que necesita el optoacoplador para estabilizar la salida
  Pot1 = analogRead(inA0);
  digitalWrite(outD1, LOW);  // Apaga la lectura
  delay(5);

  // Activa D2 y lee el valor de P2 del Joystick (Delante/atras en mi circuito)
  digitalWrite(outD1, LOW);  //Asegura que solo lea Pot2
  digitalWrite(outD5, LOW);
  digitalWrite(outD2, HIGH);
  //Serial.println("Activado D2 ");
  delay(5);  // lectura estabilizada en el optoacoplador
  Pot2 = analogRead(inA0);
  digitalWrite(outD2, LOW);  // Apaga la lectura
  //delay(5);

  // Activa D4 y lee el valor del voltaje de la bateria
  digitalWrite(outD1, LOW);  //Asegura que solo lea el nivel de la bateria
  digitalWrite(outD2, LOW);
  digitalWrite(outD5, HIGH);
  //Serial.println("Activado D5 ");
  delay(5);  // lectura estabilizada en el optoacoplador
  Bat = analogRead(inA0);
  digitalWrite(outD5, LOW);  // Apaga la lectura
  //delay(5);
}
```

.- After reading the potentiometers and battery status, we must transform the movement of the joystick in the direction and flow to the engines. If we analyze the vertical potentiometer, for example, the steps are shown in the following figure.

1. The total value (minimum, rest maximum) movement is between 0 and 1024.
2. Find out which is the midpoint thereof (rest of the lever). See leePot ();
3. Establish a margin so that the vehicle does not move with slight movements or fluctuations do not affect electricity.
4. Convert the movements up or down direction and motor current.



Steps 2 through 4 were conducted at ajustePots () ;.

```
void ajustePots() {
  // MinMotor es el valor minimo en el L298N para que la señal PWM haga mover los motores. MaxMotor es el máximo voltaje de los motores y depende de ellos y de la ateria usada
  // Transforma el valor del potenciómetro Pot1 (izquierda/Derecha) en direcciones
  // Dar un margen de valores en el centro del mismo, para evitar fluctuaciones en reposo del joystick.
  // En las pruebas, en el centro, Pot1 vale 546. Centro entre 556 y 536 (margen). Para cada joystick, estos valores pueden variar. Ajustar tras las pruebas con el joystick en reposo.
  IZQ = 0; // Borra los anteriores valores
  DER = 0;
  int temp = 0;
  //Serial.print("Pot_1:   "); Serial.println(Pot1);
  if (Pot1 >= 556) {  // Asegura el centro del mismo, en reposo, mi joystick da 546. Fijo un margen de 10 izquierda y derecha
    IZQ = map(Pot1, 556, 1024, MinMotor, MaxMotor);  // Los dos ultimos dependen de los motores y bateria. Uno es el minimo para que puedan arrancar y el otro marca el voltaje maximo
    // Tras este mapeo, IZQ toma un valor entre MinMotor y MaxMotor
  }
  if (Pot1 <= 536) {
    temp = 1024 - Pot1;  //Invierto el valor 1024-536(menor)=488. Al dismnuir el valor de Pot1, aumenta el valor de hacia la derecha
    DER = map(temp, 488, 1024, MinMotor, MaxMotor);
  }

  // Transforma el valor del potenciómetro Pot2 en avance/retroceso de los motores
  // Dar un margen de valores en el centro del mismo, para evitar fluctuaciones en reposo del joystick
  // En las pruebas en reposo mi joystick, da Pot2 529. Centro entre 539 y 519 (margen)
  // Pot2 nos marca el sentido de movimiento
  DEL = 0; // Borra los anteriores valores
  ATR = 0;
  temp = 0;
  //Serial.print("Pot_2:   "); Serial.println(Pot2);
  if (Pot2 >= 539) {
    DEL = map(Pot2, 539, 1024, MinMotor, MaxMotor);  // Los dos ultimos dependen de los motores y bateria. Uno es el minimo para que puedan arrancar y el otro marca el voltaje maximo
  }
  if (Pot2 <= 519) {  // Invierto el valor del joystick. 1024-519(menor)=505
    temp = 1024 - Pot2;
    ATR = map(temp, 505, 1024, MinMotor, MaxMotor);   //
  }
}
```

.- We assume that a device with two motors, without steering shaft, need direction and values of voltage to them. The conversion of forward / backward and left / right direction / voltage as conducted in dirMot (), taking into account the three directions forward left / front / right, the same back and incorporate rotation on itself. When going forward and turning, I do is to reduce the voltage of the wheel to which rotation proportionally to movement of the joystick and avoiding negative values, therefore, the reduction value can never be less than the feed rate (as much for the motor). Hence the use of the variable rotation (VariableGiro).

```
void dirMot() {
  /*
     Los motores giran según los distintos casos siguientes.
     .-Parado
     .-Giro sobre si mismo (2 sentidos)
     .-Avance (sin giro, con giro derecha y con giro izquierda
     .-Retroceso (sin giro, con giro derecha y con giro izquierda
  */

  int VariableGiro = 0;

  if ((DEL == 0) && (ATR == 0) && (IZQ == 0) && (DER == 0)) { // Si el joystick está centrado (todos == 0)
    Movimiento = 0;
    VoltajeMI = 0; // PWM hacia el motor izqierdo
    VoltajeMD = 0; // PWM hacia el motor derecho
    SentidoMI = 0; // Hacia donde gira el motor izquierdo (0=parado, 1=adelante, 2=atrás)
    SentidoMD = 0;  // Hacia donde gira el motor derecho (0=parado, 1=adelante, 2=atrás)
    //Serial.println("Parado");
    return;
  }
  // ***** GIRO SOBRE SI MISMO *****
  // Solo cuando el joystick está en pocicion centrado verticalmente y DER/IZQ tienen un valor positivo uno de ellos
  if ((DEL == 0) && (ATR == 0)) {  //solo giros de izquierda o derecha
    //Serial.println("Giro sobre si mismo");
    if (DER > 0) { // Giro sobre si mismo a la derecha
      Movimiento = 3;
      SentidoMI = 1;  //motor izquierdo hacia adelante
      SentidoMD = 2;  // motor derecho hacia atrás
      VoltajeMI = 160;  // ambos motores a una velocidad determinada, pero en sentido contrario
      VoltajeMD = 160;  // Estos valores, siempre mayores a la velocidad minima de comienzo de movimiento
      return;
      //Serial.println("Giro sobre si mismo a la derecha");
    }
    if (IZQ > 0) { // Giro sobre si mismo a la izquierda
      Movimiento = 3;
      SentidoMI = 2;  // motor izquierdo hacia atrás
      SentidoMD = 1;  // motor derecho hacia adelante
      VoltajeMI = 160;  // ambos motores a una velocidad determinada, pero en sentido contrario
      VoltajeMD = 160;  // Estos valores, siempre mayores a la velocidad minima de comienzo de movimiento
      return;
      //Serial.println("Giro sobre si mismo a la izquierda");
    }
  }

  // Para los avances con giro , al motor que marca el giro le resto un valor proporcional al movimiento del joystick, pero siempre menor que el avance/retroceso
  // Si el giro es muy brusco, restar un numero a las variables ultimas del MAP, ejemplo (DEL-10). Asi el motor que reduce nunca va a cero auque el joystick lo tengamos al tope de direcci
  // Ejemplo: Si DEL=300 y giro el joystick a la izquierda, el motor derecho va a 300 y el izquierdo va DE 300 A 0 (avanza menos o para), dependiendo de su posicion.
  if ((DEL > 0) && (IZQ > 0)) {
    VariableGiro = map(IZQ, MinMotor, MaxMotor, 0, DEL);
  }
  if ((DEL > 0) && (DER > 0)) {
    VariableGiro = map(DER, MinMotor, MaxMotor, 0, DEL);
  }
  if ((ATR > 0) && (IZQ > 0)) {
    VariableGiro = map(IZQ, MinMotor, MaxMotor, 0, ATR);
  }
  if ((ATR > 0) && (DER > 0)) {
    VariableGiro = map(DER, MinMotor, MaxMotor, 0, ATR);
  }

  // ***** AVANZA*****
  if (DEL > 0) {  // avance hacia adelante.Tres casos. Hacia la derecha, centro o izquierda
    //Serial.println("Marcha Adelante");
    if ((DER == 0) && (IZQ == 0)) { // Avance sin giro
      SentidoMI = 1;  // motor izquierdo hacia adelante
      SentidoMD = 1;  // motor derecho hacia adelante
      VoltajeMI = DEL; // ambos motores a la misma velocidad o modificar la trayectoria recta restando algo a uno de los valores en loop() antes de transmitir
      VoltajeMD = DEL;
      Movimiento = 3;
      return;
      //Serial.println("Adelante sin giro");
    }
    if (DER > 0) {  // avance con giro a la derecha
      SentidoMI = 1;  // motor izquierdo hacia adelante
      SentidoMD = 1;  // motor derecho hacia adelante
      VoltajeMI = DEL;                // El motor izquierdo avanza a la velocidad de hacia adelante
      VoltajeMD = DEL - VariableGiro;   //El motor derecho gira menos, dependiendo del joystick hacia la derecha.
      Movimiento = 3;
      return;
      //Serial.println("Adelante-Derecha");
    }
    if (IZQ > 0) {  // avance con giro a la izquierda
      SentidoMI = 1;  // motor izquierdo hacia adelante
      SentidoMD = 1;  // motor derecho hacia adelante
      VoltajeMI = DEL - VariableGiro;   //El motor izqquierdo gira menos, dependiendo del joystick hacia la izquierda. Disminuir el ultimo numero si queremos que gire menos
      VoltajeMD = DEL;                  //El motor derecho avanza a la velocidad de hacia adelante
      Movimiento = 3;
      return;
      //Serial.println("Adelante-Izquierda");
    }
  }

  // ***** RETROCEDE *****
  if (ATR > 0) {  // avance hacia atras. Tres casos. Hacia la derecha, centro o izquierda
    //Serial.println("Marcha Atras");
    if ((DER == 0) && (IZQ == 0)) { // Avance sin giro
      SentidoMI = 2;  // motor izquierdo hacia atrás
      SentidoMD = 2;  // motor derecho hacia atras
      VoltajeMI = ATR; // ambos motores a la misma velocidad o modificar la trayectoria recta restando algo a uno de los valores en loop() antes de transmitir
      VoltajeMD = ATR;
      Movimiento = 4;
      return;
      //Serial.println("Atras sin giro");
    }
    if (DER > 0) {  // retrocede con giro a la derecha
      SentidoMI = 2;  // motor izquierdo hacia atras
      SentidoMD = 2;  // motor derecho hacia atras
      VoltajeMI = ATR;                  // El motor izquierdo avanza lo normal
      VoltajeMD = ATR - VariableGiro ;   //El motor derecho gira menos, dependiendo del joystick hacia la derecha. Disminuir el ultimo numero si queremos que gire menos
      Movimiento = 4;
      return;
      //Serial.println("Atras-Derecha");
    }
    if (IZQ > 0) {  // avance con giro a la izquierda
      SentidoMI = 2;  // motor izquierdo hacia atras
      SentidoMD = 2;  // motor derecho hacia atras
      VoltajeMI = ATR - VariableGiro;   //El motor izqquierdo gira menos, dependiendo del joystick hacia la izquierda. Disminuir el ultimo numero si queremos que gire menos
      VoltajeMD = ATR;                  // El motor derecho avanza lo normal
      Movimiento = 4;
      return;
      //Serial.println("Atras-Izquierda");
    }
  }
}
```

.- Finally, control of battery status. When the joystick is at rest, or was unable to transmit, increase onecounter. If reaches a desired (50 times) value, analyze the state of the battery and I flashing LED (1 flash = low, 2 = very low flicker)

```
void ControlBat() {
  if (Movimiento = 0) {
    ContadorReposo = ContadorReposo + 1;
    digitalWrite(outTXG, LOW);
    //Serial.print("Contador ");  Serial.println(ContadorReposo);
  }
  else {
    //digitalWrite(outTXG, HIGH);
    ContadorReposo = 0;
  }
  if (ContadorReposo >= 50) {
    // 1024 equivale a 3.3v.
    // Con bateria de 3,7v (Normal cargada a tope=4v) y el divisor de voltaje, el valor máximo es 2v
    // Con la bateria cargada (4v) el valor leido tras el divisor es de 620 (2v)
    // Con la bateria baja (3.5v) el valor leido tras el divisor es de 543 (1.75v)
    // Con la bateria MUY baja (3.4v) el valor leido tras el divisor es de 527 (1.7v)
    //Serial.print("Valor de bateria ");  Serial.println(Bat);
    //Serial.println("");
    if (Bat <= 543 && Bat > 527) { //Bateria baja. 1 parapadeo
      digitalWrite(outTXG, HIGH);  // enciende el led de TXGood
      delay(100);
      digitalWrite(outTXG, LOW);  // apaga el led de TXGood
      delay(100);
    }
    if (Bat < 527) { //Bateria MUY baja. 3 parpadeo
      digitalWrite(outTXG, HIGH);  // enciende el led de TXGood
      delay(100);
      digitalWrite(outTXG, LOW);  // apaga el led de TXGood
      delay(100);
      digitalWrite(outTXG, HIGH);  // enciende el led de TXGood
      delay(100);
      digitalWrite(outTXG, LOW);  // apaga el led de TXGood
      delay(100);
    }
    ContadorReposo = 0;
  }
}
```

Once commented the joystick for Arduino sketch, we see the sketch of the vehicle.

Arduino (Vehicle)

On the corresponding communications (ESP-NOW) with the joystick part, and they were discussed above, so I discuss the rest. Must take into account that I have simplified enough, so whether to make changes, you work better by changing the remote control have to put the vehicle back on the table and connect it to your computer. Therefore, I limit myself to collect motion data and transfer them to L298N to move engines. Prioritized receiving emergency button and motionless time, I analyze the state of the battery.

Pines .- input and output variables used WEMOS plate:

```
// Pines de Salida hacia el L298N Control de Motores
#define pinENA 5   //D1   PWM Pasos de motor izquierdo
#define pinIN1 4   //D2   Motor Izq hacia a delante
#define pinIN2 2   //D4   Motor Izq hacia atras  //D3 da problemas y se mantiene siempre HIGH
#define pinENB 16  //D0  PWM Pasos de motor derecho
#define pinIN3 14  //D5  Motor Der hacia a delante
#define pinIN4 12  //D6  Motor Der hacia atras
#define pinEME 13  //D7  Emergencia
#define pinRXB 15  //D8  RX-Bad
#define pinBat 0   //A0  Nivel de bateria

// Valores iniciales de Variables
int Pulsador = LOW;  // Variable del valor del pulsador del joystick
int RXgood = 0;  //Si transmite correctamente
int Bateria = 0;   // Memoriza el estado de la bateria. Muestra de entrada de 12v con un divisor 47k/4k7
int VoltajeMI = 0; // PWM hacia el motor izqierdo
int VoltajeMD = 0; // PWM hacia el motor derecho
int SentidoMI = 0; // Hacia donde gira el motor izquierdo (0=parado, 1=adelante, 2=atrás)
int SentidoMD = 0; // Hacia donde gira el motor derecho (0=parado, 1=adelante, 2=atrás)
int ContadorInactivo = 0; // Controla cuando pasa un tiempo sin recibir datos, para medir el nivel de bateria y hacer parpadear el led
```

.- already in the setup () start the pins and its initial state. The remaining setup is about ESP-NOW:

```
void setup() {
  Serial.begin(500000);

  pinMode(pinENA, OUTPUT);  pinMode(pinIN1, OUTPUT);  pinMode(pinIN2, OUTPUT);  // motor izquierdo en L298N
  digitalWrite(pinIN1, LOW);  digitalWrite(pinIN2, LOW);  analogWrite(pinENA, 0);  //Paro motor izquierdo

  pinMode(pinENB, OUTPUT);  pinMode(pinIN3, OUTPUT);  pinMode(pinIN4, OUTPUT);  // motor derecho en L298N
  digitalWrite(pinIN3, LOW);  digitalWrite(pinIN4, LOW);  analogWrite(pinENB, 0);  //Paro motor derecho

  pinMode(pinEME, OUTPUT);  pinMode(pinRXB, OUTPUT);  pinMode(pinBat, INPUT);  //Leds y entrada analogica de la bateria.
  digitalWrite(pinRXB, LOW);  // Se enciende si recibe datos ok.
  digitalWrite(pinEME, LOW);  // Se enciende al pulsar Emergencia o parpadea si está baja la bateria

  //***INICIALIZACION DEL PROTOCOLO ESP-NOW si no lo esta***  (ESP-NOW)***//
  if (esp_now_init() != 0) {
```

.- In loop (), Aside from looking at the battery status, run two control functions, an annotated and speaking of the ESP-NOW, reception () and the other performs management L298N with the received data. Of course, the first thing is to analyze a possible emergency and stop the vehicle. First I set a small delay in communications to synchronize the receiver more or less with the transmitter. Run the receive function () and analyze if pressed "Emergency" to proceed with immobilisation. If I do not receive data or movement of any of the engines, unemployment also by sending data to the writeL298N () function. If no data, increment a counter for battery check. If there is data received, turn on the LED communications and of course the command to writeL298N () function to move the engine according to the data.

```
void loop() {

  delay(80);  //Si se baja, no controla bien el status de comunicaciones

  recepcion();  //Recoge los datos de la red

  if (Pulsador == 1) {
    VoltajeMI = 0; // PWM hacia el motor izqierdo
    VoltajeMD = 0; // PWM hacia el motor derecho
    SentidoMI = 0; // Hacia donde gira el motor izquierdo (0=parado, 1=adelante, 2=atrás)
    SentidoMD = 0;  // Hacia donde gira el motor derecho (0=parado, 1=adelante, 2=atrás)
    writeL298N();  // Con los valores anteriores, para los motores
    digitalWrite(pinEME, HIGH);
    //Serial.println("Parada de Emergencia. Esperar 5 segundos");
    delay(5000);   // RETARDO EN CASO DE PULSAR "EMERGENCIA"
    Pulsador = 0;
    digitalWrite(pinEME, LOW);
  }

  if ((SentidoMI == 0) && (SentidoMD == 0)) { // Cuando el movimiento recibido es cero o no recibe datos
    //Serial.println("***FALLO EN RECEPCION o sin movimiento***");
    ContadorInactivo = ContadorInactivo + 1;  // Si no recibo movimiento, incremento un contador
    digitalWrite(pinRXB, LOW);  //Apago el LED y paro motores
    VoltajeMI = 0; // PWM hacia el motor izqierdo
    VoltajeMD = 0; // PWM hacia el motor derecho
    SentidoMI = 0; // Hacia donde gira el motor izquierdo (0=parado, 1=adelante, 2=atrás)
    SentidoMD = 0; // Hacia donde gira el motor derecho (0=parado, 1=adelante, 2=atrás)
```

```
    writeL298N();  // Con los datos anteriores, paro los motores

  }
  else {
    ContadorInactivo = 0;
    digitalWrite(pinRXB, HIGH);  //Si hay recepcion de datos, enciende el led
  }

  if (ContadorInactivo >= 20) {  // Si está en reposo, controlo el estado de la bateria. Aumentar el numero si queremos que haga menos controles de la misma.
    // en A0, 1024 equivale a un voltaje de 3.3v.
    //Tras el divisor de voltaje de 11v, estando ok, medimos 1v, equivalente a 310 en A0
    Bateria = analogRead(pinBat);
    if (Bateria <= 153) { // "Bateria MUY baja" (5,5v) Si no hay transmisión, aprovecho para controlar el estado de la bateria
      //Serial.println("Bateria MUY baja");   //  3 parpadeos cortos del led de Emergencia
      digitalWrite(pinEME, HIGH); delay(50); digitalWrite(pinEME, LOW); delay(50);
      digitalWrite(pinEME, HIGH); delay(50); digitalWrite(pinEME, LOW); delay(50);
      digitalWrite(pinEME, HIGH); delay(50); digitalWrite(pinEME, LOW); delay(50);
      ContadorInactivo = 0;
    }
    if (Bateria > 153 && Bateria <= 155) {  // "Bateria baja", entre 7v y 5.5v
      //Serial.println("Bateria baja");  // 1 parpadeo del led de Emergencia
      digitalWrite(pinEME, HIGH); delay(100); digitalWrite(pinEME, LOW); delay(100);
      ContadorInactivo = 0;
    }
    else {
      //Serial.println("Bateria OK");
    }

  }

  writeL298N();

  //FIN de loop()
}
```

.- writeL298N function ()

If you remember L298N table, simply write these values with the data received


```
void writeL298N() {
  //Serial.println("Movimiento de Motores");
  // Motor Izquierdo   =   ENA, IN1, IN2
  //Serial.println("Muevo Motores "); Serial.println("");
  switch (SentidoMI) {  //Izquierda
    case 0:  // Parado
      digitalWrite(pinIN1, LOW);
      digitalWrite(pinIN2, LOW);
      analogWrite(pinENA, 0);
      break;
    case 1:  // Adelante
      digitalWrite(pinIN1, HIGH);
      digitalWrite(pinIN2, LOW);
      analogWrite(pinENA, VoltajeMI);
      //Serial.print("Adelante Motor Izquierdo "); Serial.println(VoltajeMI);
      break;
    case 2:  // Atrás
      digitalWrite(pinIN1, LOW);
      digitalWrite(pinIN2, HIGH);
      analogWrite(pinENA, VoltajeMI);
      //Serial.print("Atras Motor Izquierdo "); Serial.println(VoltajeMI);
      break;
  }
  // Motor Derecho  =   ENB, IN3, IN4
  switch (SentidoMD) {  //Izquierda
    case 0:  // Parado
      digitalWrite(pinIN3, LOW);
      digitalWrite(pinIN4, LOW);
      analogWrite(pinENB, 0);
      break;
    case 1:  // Adelante
      digitalWrite(pinIN3, HIGH);
      digitalWrite(pinIN4, LOW);
      analogWrite(pinENB, VoltajeMD);
      //Serial.print("Adelante Motor Derecho "); Serial.println(VoltajeMD);
      break;
    case 2:  // Atrás
      digitalWrite(pinIN3, LOW);
      digitalWrite(pinIN4, HIGH);
      analogWrite(pinENB, VoltajeMD);
      //Serial.print("Atras Motor Derecho "); Serial.println(VoltajeMD);
      break;
  }
  //Serial.print("Motor Izquierdo "); Serial.println(VoltajeMI);
  //Serial.print("Motor Derecho "); Serial.println(VoltajeMD);
}
```

This is all. It is difficult for a project goes prize in a contest where there are not many friends to vote for you. I do not intend to win competitions, but to clarify concepts. If a person appreciates this work serves to acquire knowledge and then develop some own idea, I'm content.

A greeting:
Miguel A.