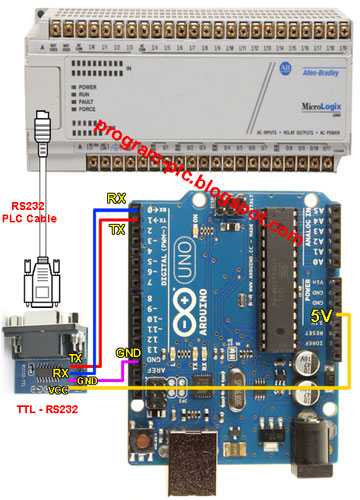
**[simple\_lcd\_serial.ino](https://gist.github.com/ravenlp/5794749" \l "file-simple_lcd_serial-ino)**

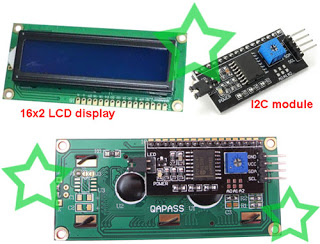
|  |  |
| --- | --- |
|  | /\* |
|  | LiquidCrystal Library - Serial Input |
|  | \*/ |
|  |  |
|  | /\* |
|  | This example code is in the public domain. |
|  |  |
|  | http://arduino.cc/en/Tutorial/LiquidCrystalSerial |
|  | \*/ |
|  |  |
|  | // include the library code: |
|  | #include <LiquidCrystal.h> |
|  |  |
|  |  |
|  | char c[33]; |
|  |  |
|  | // initialize the library with the numbers of the interface pins |
|  | LiquidCrystal lcd(8, 9, 4, 5, 6, 7); |
|  |  |
|  | void setup(){ |
|  | // set up the LCD's number of columns and rows: |
|  | lcd.begin(16, 2); |
|  | // initialize the serial communications: |
|  | Serial.begin(9600); |
|  | } |
|  |  |
|  | void loop() |
|  | { |
|  | // when characters arrive over the serial port... |
|  | if (Serial.available()) { |
|  | // wait a bit for the entire message to arrive |
|  | delay(100); |
|  | // clear the screen |
|  | lcd.clear(); |
|  | // read all the available characters |
|  | int i = 0; |
|  | while (Serial.available() > 0) { |
|  | c[i++] = Serial.read(); |
|  | //lcd.write(); |
|  | //data += Serial.read(); |
|  |  |
|  | } |
|  | // Clear the rest |
|  | for(; i < 33; i++){ |
|  | c[i] = ' '; |
|  | } |
|  |  |
|  | //Print the first row |
|  | for( i = 0; i < 16; i++){ |
|  | lcd.write(c[i]); |
|  | } |
|  |  |
|  | //Print the second row |
|  | lcd.setCursor(0,1); |
|  | for( i = 16; i < 33; i++){ |
|  | lcd.write(byte(c[i])); |
|  | } |
|  | } |
|  | delay(100); |
|  | } |

<https://program-plc.blogspot.com/2016/01/lcd-display-arduino-for-allen-bradley.html>



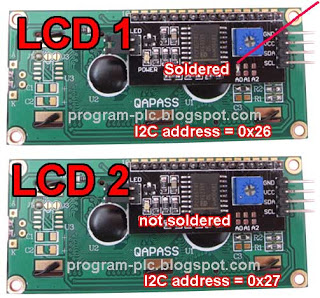
**Hardware for LCD display application for Allen Bradley PLC**

1. 2 pieces 16x2 LCD display module hd44780
2. 2 pieces  I2C module for LCD display
3. Arduino UNO
4. Allen-Bradley PLC Micrologix
5. RS232 PLC Cable Programming for Allen-Bradley PLC Micrologix
6. TTL to RS232 Module
7. Adaptor for Arduino
8. 4 pieces push button switch for testing

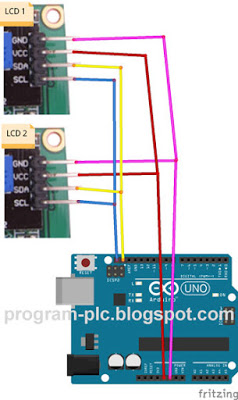
[](https://3.bp.blogspot.com/-WMDhAfaU5Vo/VpXUXGUYImI/AAAAAAAAE1o/fSYwtfAY41o/s1600/I2C-module-and-16x2-LCD-display.jpg)

**Hardware Connections for LCD display application for Allen Bradley PLC**

**1. I2C LCD Display**

[](https://3.bp.blogspot.com/--bauajc5mS4/VpXUkxvAxjI/AAAAAAAAE1w/C5HgC_cOHTw/s1600/I2C-LCD-Display.jpg)

**2. I2C LCD Display and Arduino**

[](https://1.bp.blogspot.com/-hU2cEJOgfA0/VpXUtV75mNI/AAAAAAAAE14/xY_e8YnZ80A/s1600/I2C-LCD-Display-and-Arduino.jpg)

**3. TTL to RS232 Module, Arduino and PLC**

[http://2.bp.blogspot.com/-zKEswKAUS0U/Vmjcph0y7yI/AAAAAAAAEo0/m3613S487-Y/s1600/Hardware-Connection-Between-Allen-Bradley-PLC-and-Arduino.jpg](https://2.bp.blogspot.com/-zKEswKAUS0U/Vmjcph0y7yI/AAAAAAAAEo0/m3613S487-Y/s1600/Hardware-Connection-Between-Allen-Bradley-PLC-and-Arduino.jpg)

**4. TTL-RS232 Module to RS232 PLC Cable**

Connect DB9 of TTL RS232 Module to DB9 RS232 PLC Cable  
If TTL to RS232 use DB9 female then use DB9 Male crossover cable : [http://2.bp.blogspot.com/-BhR43WPLxIk/VAUNuok0FfI/AAAAAAAACxo/JMyctiWw3s0/s1600/DB9-Male-crossover-cable.jpg](https://2.bp.blogspot.com/-BhR43WPLxIk/VAUNuok0FfI/AAAAAAAACxo/JMyctiWw3s0/s1600/DB9-Male-crossover-cable.jpg)

**5. Push Button to PLC Input**

1. Connect push button 1 to PLC Input I:0/0
2. Connect push button 2 to PLC Input I:0/1
3. Connect push button 3 to PLC Input I:0/2
4. Connect push button 4 to PLC Input I:0/3

**Software for LCD display application for Allen Bradley PLC**

1. Arduino Software (IDE) Version 1.6.6
2. Arduino Library : [click here](http://program-plc.blogspot.com/p/project-file-lcd-display-arduino-for.html)

Unzip and copy paste DF1 and LiquidCrystal\_I2C to ..\arduino-nightly\libraries

**Download File for LCD  Display Arduino Project**

1. LCD display arduino project file : [click here](http://program-plc.blogspot.com/p/project-file-lcd-display-arduino-for.html)
2. Allen Bradley PLC MicroLogix project file: [click here](http://program-plc.blogspot.com/p/project-file-lcd-display-arduino-for.html)
3. #include <Wire.h>
4. #include <**LiquidCrystal\_I2C**.h>
5. #include <DF1.h>
6. //Allen Bradley PLC
7. #define DF1destination 1
8. #define DF1baud 9600
9. #define DF1format SERIAL\_8N1
10. #define DF1timeout 1000
11. enum
12. {
13. DF1PACKET1,
14. DF1TOTAL\_NO\_OF\_PACKETS
15. };
16. **DF1Packet** DF1packets[DF1TOTAL\_NO\_OF\_PACKETS];
17. **DF1packetPointer** DF1packet1 = &DF1packets[DF1PACKET1];
18. unsigned int DF1readRegsLCD[32];
19. //lcd display arduino 16 x 2
20. **LiquidCrystal\_I2C** lcd1(0x26,16,2);
21. **LiquidCrystal\_I2C** lcd2(0x27,16,2);
22. String LCD\_STR[4];
23. String LCD\_STR\_PREV[4];
25. void setup()
26. {
28. DF1\_construct(DF1packet1, DF1destination, DF1\_READ\_N7, 0, 32, DF1readRegsLCD);
30. DF1\_configure(&**Serial**, DF1baud, DF1format, DF1timeout, DF1packets, DF1TOTAL\_NO\_OF\_PACKETS);
31. lcd1.init();
32. lcd2.init();
33. //lcd display screen backlight
34. lcd1.backlight();
35. lcd2.backlight();
36. //clear lcd display
37. lcd1.clear();
38. lcd1.home();
39. lcd2.clear();
40. lcd2.home();
41. }
43. void loop()
44. {
45. DF1\_update();
47. //lcd display memory from N7:0 to N7:31 of Allen Bradley PLC
48. for (int i = 0; i < 4; i++)LCD\_STR[i] ="";
49. int counter=0;
50. int index=0;
51. for (int i = 0; i < 32; i++){
52. if(DF1readRegsLCD[i]==0){
53. LCD\_STR[index]+="  "; // 2 space
54. }else{
55. LCD\_STR[index]+=char((DF1readRegsLCD[i] >> 8) & 0x00FF);
56. LCD\_STR[index]+=char(DF1readRegsLCD[i] & 0x00FF);
57. }
58. counter++;
59. if(counter==8){
60. index++;
61. counter=0;
62. }
63. }
64. //LCD1 Display Line 1
65. if(LCD\_STR\_PREV[0] != LCD\_STR[0]){
66. lcd1.setCursor(0, 0);
67. lcd1.print(LCD\_STR[0]);
68. LCD\_STR\_PREV[0] = LCD\_STR[0];
69. }
70. //LCD1 Display Line 2
71. if(LCD\_STR\_PREV[1] != LCD\_STR[1]){
72. lcd1.setCursor(0, 1);
73. lcd1.print(LCD\_STR[1]);
74. LCD\_STR\_PREV[1] = LCD\_STR[1];
75. }
76. //LCD2 Display Line 1
77. if(LCD\_STR\_PREV[2] != LCD\_STR[2]){
78. lcd2.setCursor(0, 0);
79. lcd2.print(LCD\_STR[2]);
80. LCD\_STR\_PREV[2] = LCD\_STR[2];
81. }
82. //LCD2 Display Line 2
83. if(LCD\_STR\_PREV[3] != LCD\_STR[3]){
84. lcd2.setCursor(0, 1);
85. lcd2.print(LCD\_STR[3]);
86. LCD\_STR\_PREV[3] = LCD\_STR[3];
87. }
88. }

<https://forum.arduino.cc/index.php?topic=511826.0>

the LCD works fabulous, but still can't get the Arduino to receive something from the transducer

// include the library code:  
#include <LiquidCrystal.h>    
 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  
   
String sensorValue;  
  
void setup() {  
Serial.begin(9600);  
 //Contrast  
 analogWrite(6,100);  
  // set up the LCD's number of columns and rows:  
 lcd.begin(16, 2);  
}  
  
void Pressure() {  
lcd.setCursor(0, 0);  
lcd.print("Pressure:"); //texto que me dice que estoy midiendo  
  
Serial.println ("@253PR3?;FF");  
sensorValue = Serial.readString();  
sensorValue.remove(0, 7); //Quitar los primeros 7 caracteres de @253ACK7.10E+2;FF para tener 7.10E+2;FF  
sensorValue.remove(7); //Quitar caracteres desde el 7mo, oara obtener 7.39E+2  
  
lcd.setCursor(0,1);  
lcd.print(sensorValue);  
lcd.print(" Torr");   
}  
  
void loop() {  
 //  
Pressure(); //loop de sensorValue

<https://www.teachmemicro.com/microcontroller-serial-communication/>

Arduino Software Serial

Running out of pins to use is common in microcontroller programming. This is especially true for the Arduino UNO which has only one serial port. The GSM module in fact uses software serial to communicate witht the arduino board.  Arduino's software serial library is demonstrated by the included built-in example File -> Examples -> Software Serial -> SoftwareSerialExample:

Arduino code

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | #include <SoftwareSerial.h>    SoftwareSerial mySerial(10, 11); // RX, TX    void setup()  {    // Open serial communications and wait for port to open:    Serial.begin(57600);    while (!Serial) {      ; // wait for serial port to connect. Needed for Leonardo only    }      Serial.println("Goodnight moon!");      // set the data rate for the SoftwareSerial port    mySerial.begin(4800);    mySerial.println("Hello, world?");  }    void loop() // run over and over  {    if (mySerial.available())      Serial.write(mySerial.read());    if (Serial.available())      mySerial.write(Serial.read());  } |

You can assign whichever pins are available for use as Tx and Rx pins! (The above code assigns the serial pins to digital pins 10 and 11). Once the software serial object is defined, you can now use the functions of the library which are the basically the same functions as with hardware serial.

<http://image.dfrobot.com/image/data/DFR0077/MAX202%20datasheet.pdf>



Modbus y Arduino

<https://techsparx.com/energy-system/modbus/arduino-modbus-rtu-temperature-sensor.html>

nuevos Arduinios para IOT

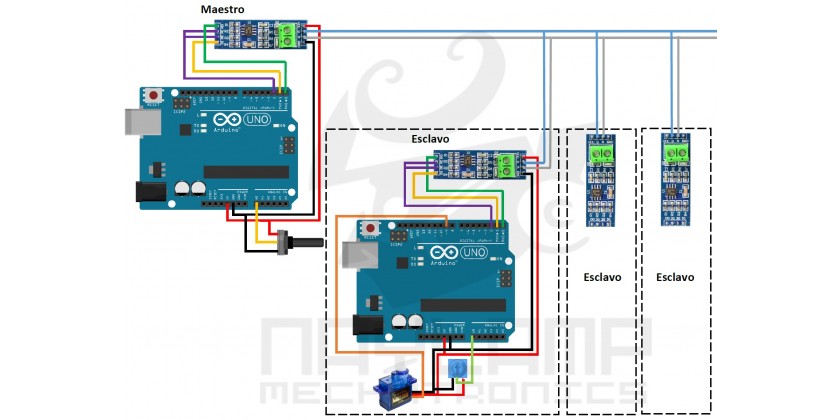
<https://programarfacil.com/podcast/novedades-arduino-iot-industrial/>

Shields para Arduino y Redes Industriales

<https://www.industriaembebidahoy.com/shields-arduino-para-redes-de-comunicaciones-industriales/>

# Comunicación RS485 con Arduino

<https://naylampmechatronics.com/blog/37_Comunicaci%C3%B3n-RS485-con-Arduino.html>

[](https://naylampmechatronics.com/modules/smartblog/images/37-single-default.jpg)

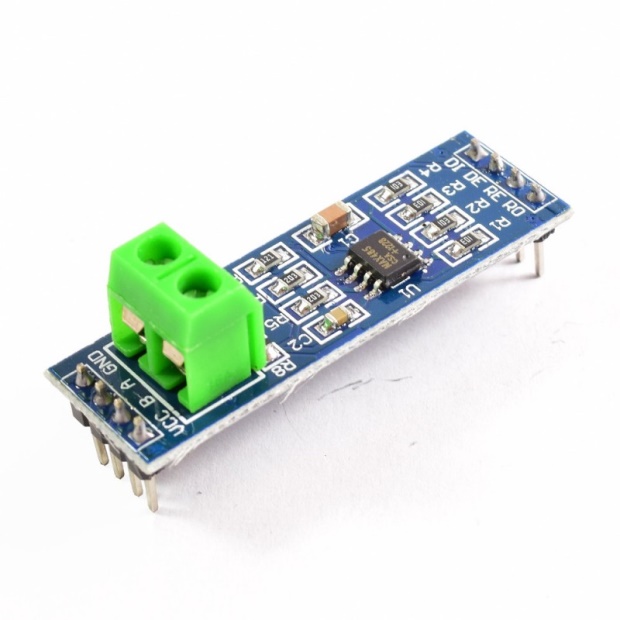
Explicaremos como utilizar los Módulos RS485 e implementaremos varios ejemplos para comunicar 2 Arduinos a través de RS485, desarrollaremos la comunicación simplex, full duplex y half duplex.

El estándar industrial RS485 o también conocido como **EIA-485** es un estándar de capa física según el modelo OSI. Es decir, no pone normas ni restricciones sobre el contenido, forma, o codificación de los mensajes enviados. Utilizando RS485 como capa física es posible implementar un bus de campo industrial tipo MODBUS para comunicación entre equipos industriales o DMX para el control de luces entre otras aplicaciones. El estándar RS485 es ampliamente utilizado por su robustez, fácil implementación y buenas prestaciones.

En este tutorial trabajaremos el tema de Comunicación Serial a sobre RS485 con Arduino.

RS485 está definido como un sistema de comunicación en bus diferencial multipunto, es ideal para transmitir a velocidades medianas sobre largas distancias (35 Mbit/s hasta 10 metros y 100 kbit/s en 1200 metros) y a través de canales ruidosos, ya que al emplear voltajes diferenciales reduce los ruidos eléctricos que aparecen en la línea de transmisión. El medio físico de transmisión (cableado) es el cable par trenzado que admite hasta 32 estaciones en 1 solo bus, con una longitud máxima de 1200 metros operando entre 300 y 19200 bit/s bajo comunicación half-duplex (semiduplex).

Para la implementación de los ejemplos del tutorial utilizaremos los [**Módulos RS485 a Serial TTL**](http://www.naylampmechatronics.com/alambrico/62-modulo-rs485.html)



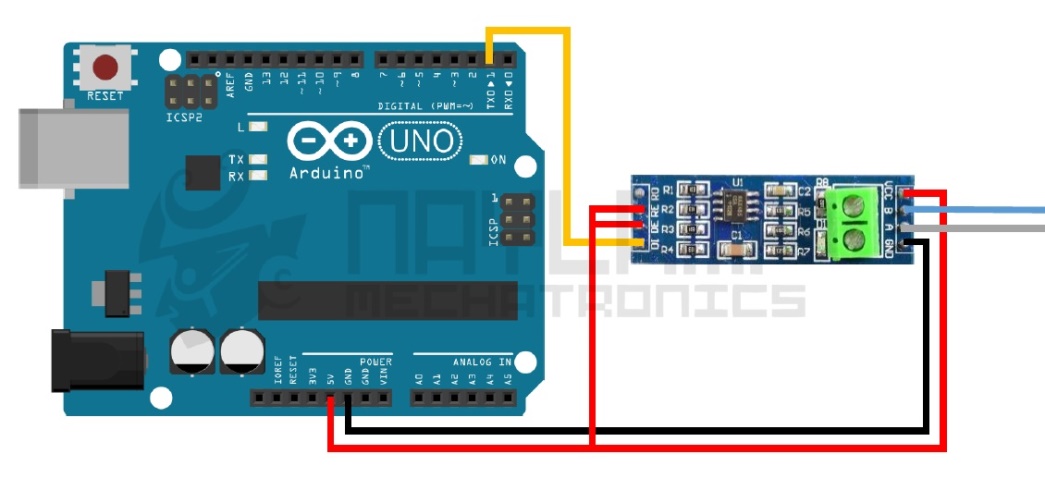
El módulo tiene acceso a todos los pines del chip MAX485 y tiene todos los componentes adicionales para su correcto funcionamiento. Los pines "A" y "B" se pueden conectar también desde la bornera.

Las pines A y B son por donde se envían y reciben los datos, B es la negación en voltaje de A, se envían los mismos datos pero en la línea B están negados, de esta forma se pueden restar ambas señales y eliminar el ruido y quedarnos solo con la señal de datos. El encargado de esta transmisión diferencial es el chip MAX485.

Desde el punto de vista del microcontrolador o Arduino, la comunicación se trabaja como una comunicación serial, trabajamos con los pines RX y TX; y dependiendo del tipo de conexión se puede usar un pin más para especificar si estamos enviando o recibiendo datos.

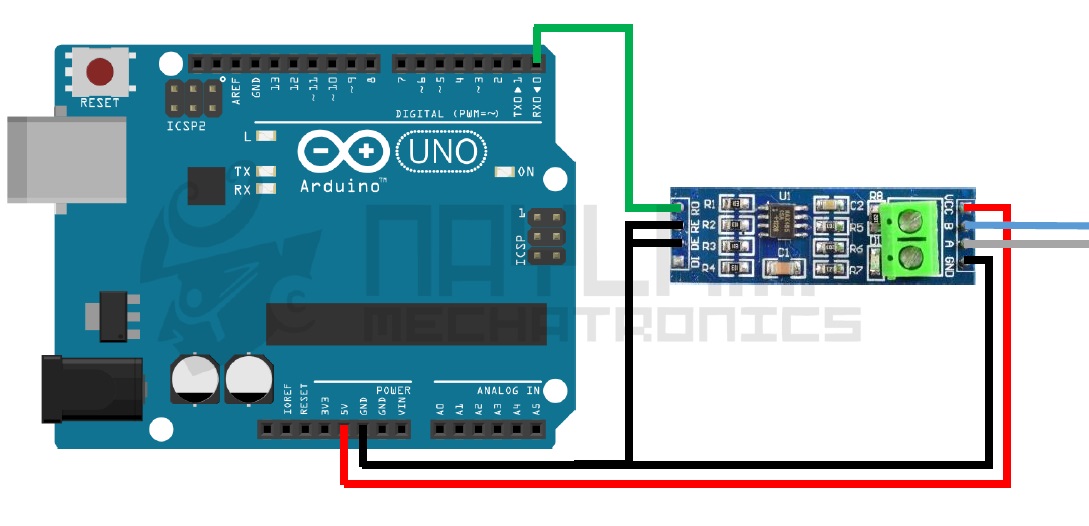
## ****Usar le módulo RS485 como transmisor****

En esta configuración el modulo solo trabaja como transmisor, para que el modulo sepa que  las salida A B se van a comportar como salida de datos, se tiene que conectar a 5V los pines RE y DE.  Desde  el Arduino se envían los datos hacia el pin DI (Data Input) del módulo y este  transmitirá los datos por los pines AB



## ****Usar el módulo RS485 como receptor****

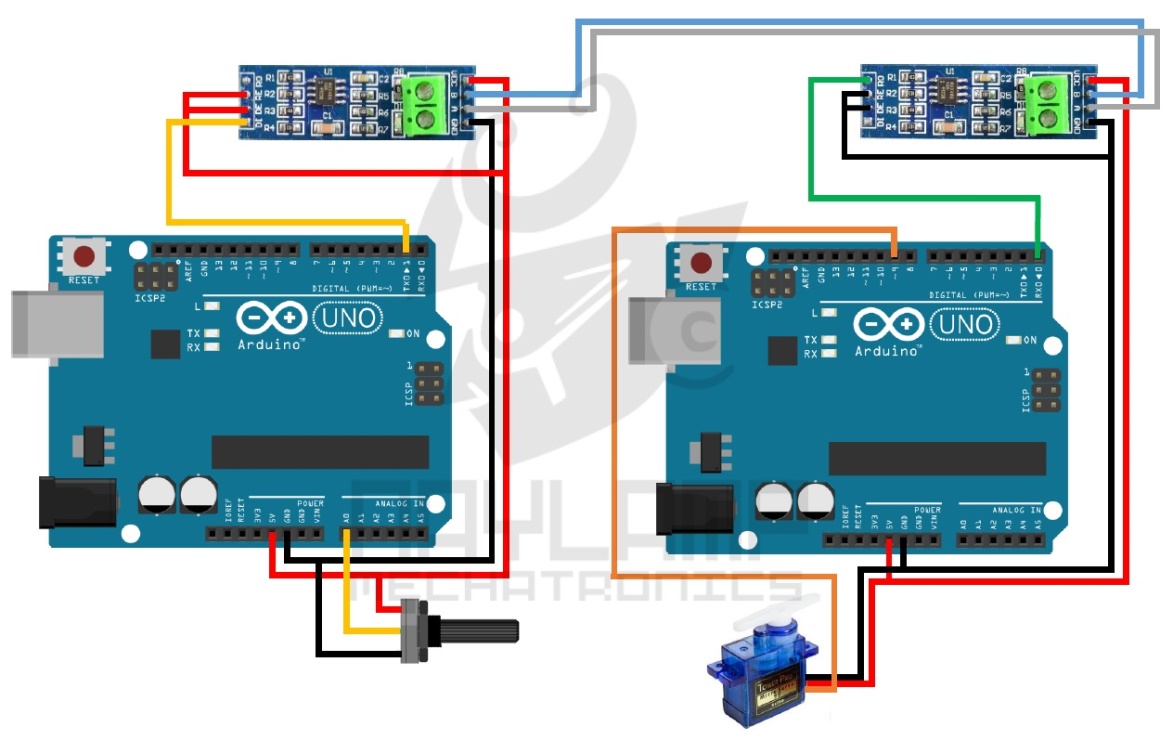
Al conectar los pines RE y DE el modulo se comporta como Receptor, y los datos recibidos por AB estarán presentes en el pin RO(Receiver Output), conectando el pin RO del módulo al RX de nuestro Arduino podemos leer los datos recibidos.



## ****Ej 1. Comunicación Simplex entre dos Arduinos por RS485****

Una comunicación simplex es una comunicación unidireccional, en este caso un Arduino se comporta solo como transmisor y el otro solo como receptor, a nivel de programación es como si estuviéramos trabajando con una comunicación serial, pero en un solo sentido. Uno envía y el otro solo recibe datos.

Veamos un ejemplo:  
  
Desde un Arduino a través de un potenciómetro moveremos un servomotor que estará conectado  en otro Arduino, solo dos cables (salidas A y B del RS485) unirán a los Arduinos, si la distancia es larga se recomienda usar cable trenzado.



El código del transmisor es el siguiente:

void **setup**()

{

**Serial**.begin(9600);

}

void **loop**()

{

  int lectura = analogRead(0);//leemos el valor del potenciómetro (de 0 a 1023)

  byte angulo= map(lectura, 0, 1023, 0, 180); // escalamos la lectura a un valor de ángulo (entre 0 y 180)

**Serial**.write(angulo); //enviamos el ángulo correspondiente

  delay(50);

}

El código del Arduino receptor es el siguiente:

#include <Servo.h>

Servo myservo; // creamos el objeto servo

void **setup**()

{

**Serial**.begin(9600);

  myservo.attach(9); // asignamos el pin 9 para el servo.

}

void **loop**()

{

  if (**Serial**.available()) {

    int angulo = **Serial**.read(); //Leemos el dato recibido

    if(angulo<=180) //verificamos que sea un valor en el rango del servo

    {

      myservo.write(angulo); //movemos el servomotor al ángulo correspondiente.

    }

  }

}

Como se observa es una simple comunicación serial, en el Arduino transmisor se hace la lectura del potenciómetro se lo escala y se lo envía serialmente, en el receptor recibimos el dato y movemos el servomotor.

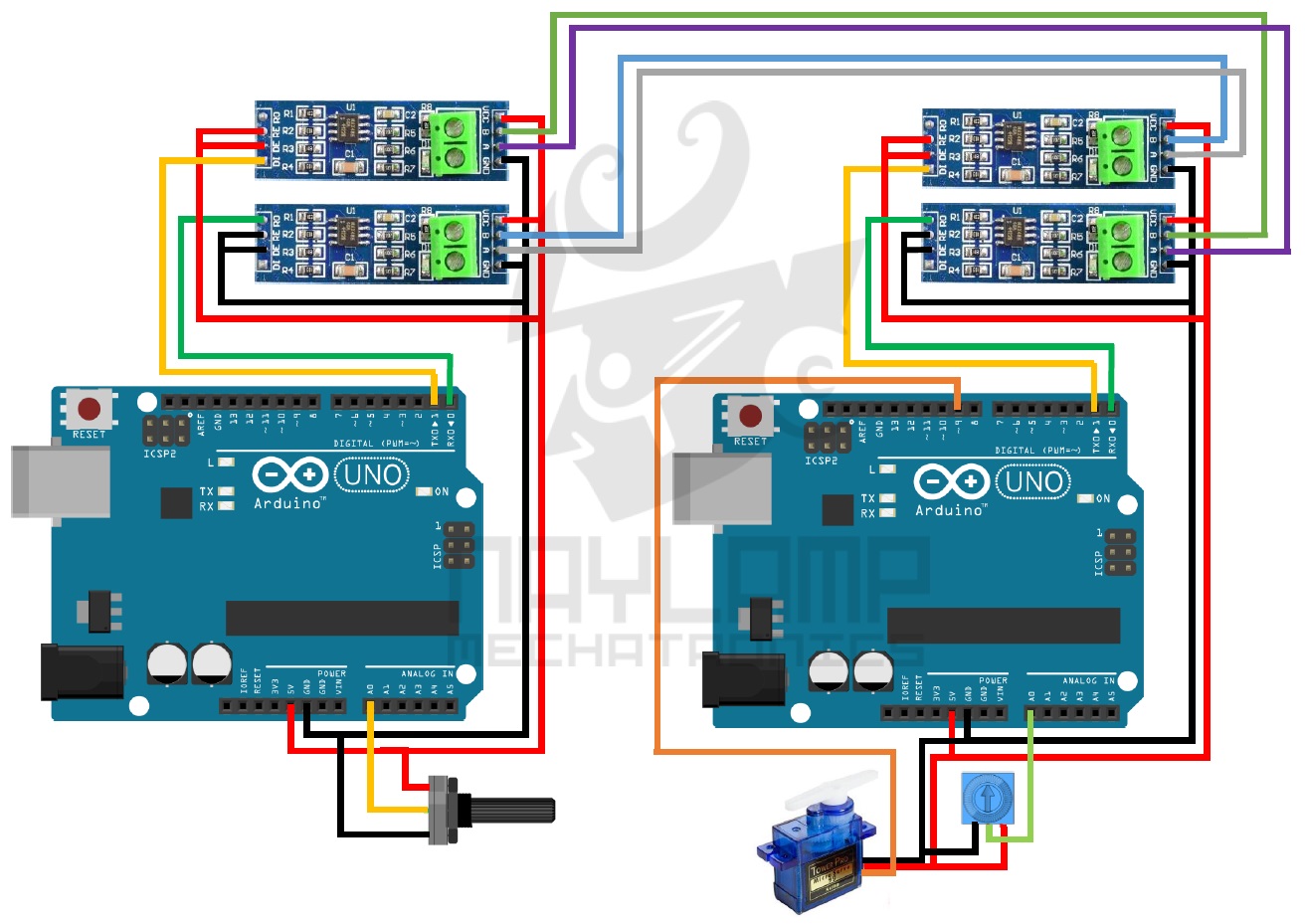
De esta forma podemos realizar una comunicación entre dos Arduino pero en una sola dirección, para hacerlo en dos direcciones tenemos dos formas, half-duplex y full-duplex

## ****Ej 2. Comunicación full dúplex entre dos Arduinos****

En este caso necesitamos agregar otro par de líneas más, en total unirían a los Arduino 4 líneas, un par son para  transmitir (TX) y otro par para recibir (RX).

En el siguiente ejemplo desde un Arduino no solo enviaremos datos para mover un servomotor sino también recibiremos datos de un sensor, usaremos un potenciómetro para simular el sensor.

Las conexiones serían las siguientes:



Al Arduino de la izquierda, lo llamaremos Maestro, pues es el Arduino principal y quien administrara todas las órdenes, mientras que el segundo Arduino  lo denominaremos Esclavo; esta no es específicamente la definición de Maestro/Esclavo en una comunicación RS485 pero usaremos estas denominaciones para saber a qué Arduino nos estamos refiriendo.

El código del Arduino Maestro.

const int ledPin = 13; // Numero del pin para el Led

void **setup**()

{

**Serial**.begin(9600);

  pinMode(ledPin, OUTPUT);//inicializamos el pin del Led como salida

}

void **loop**()

{

  if(**Serial**.available())

  {

    if(**Serial**.read()=='i') //si recibimos el inicio de trama

    {

       int dato=**Serial**.parseInt(); //recibimos valor numérico

       if(**Serial**.read()=='f') //Si el fin de trama es el correcto

       {

         funcion(dato);  //Realizamos la acción correspondiente

       }

    }

  }

  int lectura = analogRead(0);//leemos el valor del potenciómetro (de 0 a 1023)

  int angulo= map(lectura, 0, 1023, 0, 180); // escalamos la lectura a un valor de ángulo (entre 0 y 180)

  //---enviamos el ángulo para mover el servo------

**Serial**.print("I"); //inicio de trama

**Serial**.print("S"); //S para indicarle que vamos a mover el servo

**Serial**.print(angulo); //ángulo o dato

**Serial**.print("F"); //fin de trama

  //----------------------------

  delay(50);

  //---solicitamos una lectura del sensor----------

**Serial**.print("I"); //inicio de trama

**Serial**.print("L"); //L para indicarle que vamos a Leer el sensor

**Serial**.print("F"); //fin de trama

  //------------------------------------------------

  delay(50);

}

//esta función  puede variar de acuerdo a su necesidad

void funcion(int dato)

{

  if(dato>500)

  digitalWrite(ledPin, HIGH);

  else

  digitalWrite(ledPin, LOW);

}

Código del Arduino Esclavo:

#include <Servo.h>

Servo myservo; // creamos el objeto servo

void **setup**()

{

**Serial**.begin(9600);

  myservo.attach(9); // asignamos el pin 9 para el servo.

}

void **loop**()

{

  if(**Serial**.available()>0)

  {

    if(**Serial**.read()=='I') //Si recibimos el inicio de trama

    {

      char funcion=**Serial**.read(); //leemos el carácter de función

      //---Si el carácter de función es una S entonces la trama es para mover el motor-----------

      if(funcion=='S')

       {

           int angulo=**Serial**.parseInt(); //recibimos el ángulo

           if(**Serial**.read()=='F') //Si el fin de trama es el correcto

           {

             if(angulo<=180) //verificamos que sea un valor en el rango del servo

              {

                myservo.write(angulo); //movemos el servomotor al ángulo correspondiente.

              }

           }

       }

       //---Si el carácter de función es L entonces el maestro está solicitando una lectura del sensor

       else if(funcion=='L')

       {

          if(**Serial**.read()=='F') //Si el fin de trama es el correcto

           {

             int lectura = analogRead(0); //realizamos la lectura del sensor

             //----enviamos la respuesta-----

**Serial**.print("i"); //inicio de trama

**Serial**.print(lectura); //valor del sensor

**Serial**.print("f"); //fin de trama

             //-----------------

           }

       }

    }

  }

  delay(10);

}

Como se observa se ha establecido una trama para la comunicación:

***[Inicio de trama][Función][Valor][Fin de trama]***

En nuestro caso el inicio de trama es el carácter ‘A’ , la función es el carácter S o L para indicar que vamos a mover el servo o solicitar una lectura del sensor, el [valor] solo estará presente cuando la función necesite enviar una dato, y el fin de trama que usamos es el carácter F. Unos ejemplos de esta trama serian: “IS90F”,”IS120F”,”ILF”, etc.

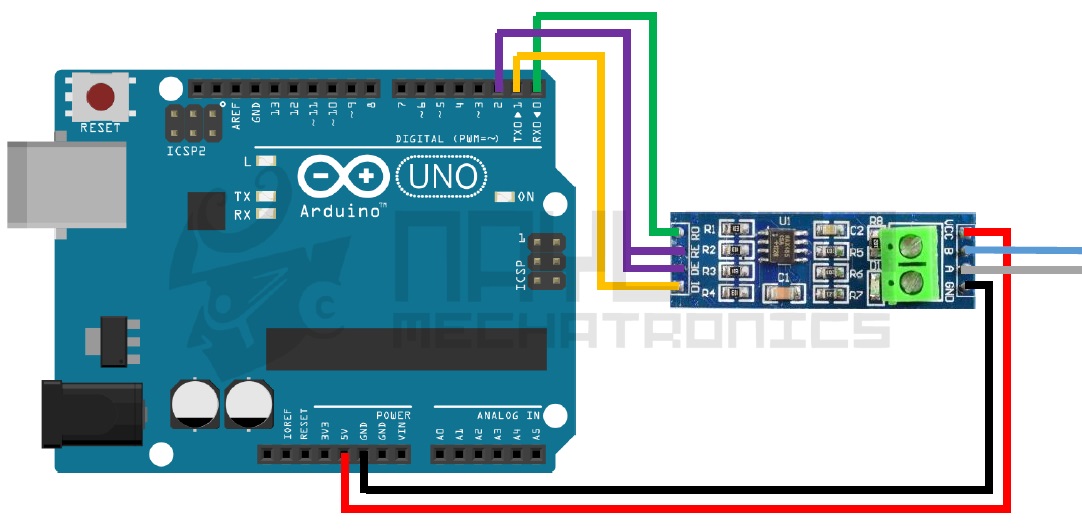
 El esclavo interpreta esta trama y realiza la función correspondiente, si es una función que necesite responder, la trama de respuesta es:

***[Inicio de trama][Valor][Fin de trama]*** , como por ejemplo “i865f”, “i64f”

## ****Ej 3. Comunicación half dúplex entre dos Arduinos****

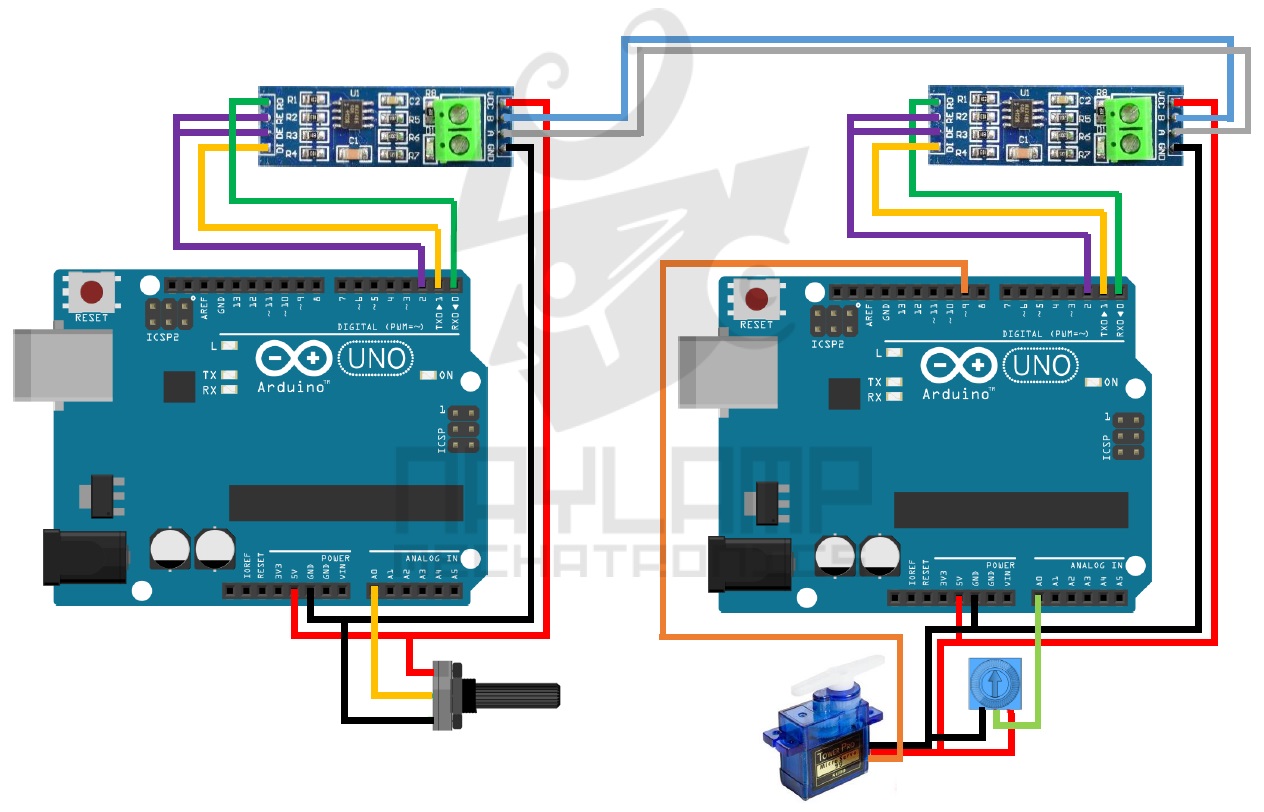
En una comunicación half dúplex utiliza un solo canal para comunicarse, en un momento por el canal se transmiten datos y en otro momento se reciben datos, pero nunca podremos transmitir y recibir a la vez.

Para realizar esta comunicación los pines DE y RE del módulo RS485 deben ir conectados al Arduino, con esto desde el programa podemos establecer al módulo como transmisor o receptor



El siguiente ejemplo hace lo mismo que el ejemplo anterior solo que esta vez se usa un módulo rs485 por Arduino y un par de cables para comunicarse.

Las conexiones serían las siguientes:



Al igual que en el caso anterior el Arduino de la izquierda será el maestro y el de la derecha será el esclavo.

Código del Maestro:

const int ledPin = 13; // Numero del pin para el Led

const int EnTxPin = 2; // HIGH:TX y LOW:RX

void **setup**()

{

**Serial**.begin(9600);

**Serial**.setTimeout(100);//establecemos un tiempo de espera de 100ms

  //inicializamos los pines

  pinMode(ledPin, OUTPUT);

  pinMode(EnTxPin, OUTPUT);

  digitalWrite(ledPin, LOW);

  digitalWrite(EnTxPin, HIGH);

}

void **loop**()

{

  int lectura = analogRead(0);//leemos el valor del potenciómetro (de 0 a 1023)

  int angulo= map(lectura, 0, 1023, 0, 180);// escalamos la lectura a un valor de ángulo (entre 0 y 180)

  //---enviamos el ángulo para mover el servo------

**Serial**.print("I"); //inicio de trama

**Serial**.print("S"); //S para indicarle que vamos a mover el servo

**Serial**.print(angulo); //ángulo o dato

**Serial**.print("F"); //fin de trama

  //----------------------------

  delay(50);

  //---solicitamos una lectura del sensor----------

**Serial**.print("I"); //inicio de trama

**Serial**.print("L"); //L para indicarle que vamos a Leer el sensor

**Serial**.print("F"); //fin de trama

**Serial**.flush(); //Esperamos hasta que se envíen los datos

  //----Leemos la respuesta del Esclavo-----

  digitalWrite(EnTxPin, LOW); //RS485 como receptor

  if(**Serial**.find("i"))//esperamos el inicio de trama

  {

      int dato=**Serial**.parseInt(); //recibimos valor numérico

      if(**Serial**.read()=='f') //Si el fin de trama es el correcto

       {

         funcion(dato);  //Realizamos la acción correspondiente

      }

  }

  digitalWrite(EnTxPin, HIGH); //RS485 como Transmisor

  //----------fin de la respuesta-----------

}

void funcion(int dato)

{

  if(dato>500)

  digitalWrite(ledPin, HIGH);

  else

  digitalWrite(ledPin, LOW);

}

Código de Esclavo:

#include <Servo.h>

Servo myservo; // creamos el objeto servo

const int EnTxPin = 2;

void **setup**()

{

**Serial**.begin(9600);

  myservo.attach(9); // asignamos el pin 9 para el servo.

  pinMode(EnTxPin, OUTPUT);

  digitalWrite(EnTxPin, LOW); //RS485 como receptor

}

void **loop**()

{

  if(**Serial**.available())

  {

    if(**Serial**.read()=='I') //Si recibimos el inicio de trama

    {

      char funcion=**Serial**.read();//leemos el carácter de función

      //---Si el carácter de función es una S entonces la trama es para mover el motor-----------

      if(funcion=='S')

       {

           int angulo=**Serial**.parseInt(); //recibimos el ángulo

           if(**Serial**.read()=='F') //Si el fin de trama es el correcto

           {

             if(angulo<=180) //verificamos que sea un valor en el rango del servo

              {

                myservo.write(angulo); //movemos el servomotor al ángulo correspondiente.

              }

           }

       }

       //---Si el carácter de función es L entonces el maestro está solicitando una lectura del sensor---

       else if(funcion=='L')

       {

          if(**Serial**.read()=='F') //Si el fin de trama es el correcto

           {

             int lectura = analogRead(0); //realizamos la lectura del sensor

             digitalWrite(EnTxPin, HIGH); //rs485 como transmisor

**Serial**.print("i"); //inicio de trama

**Serial**.print(lectura); //valor del sensor

**Serial**.print("f"); //fin de trama

**Serial**.flush(); //Esperamos hasta que se envíen los datos

             digitalWrite(EnTxPin, LOW); //RS485 como receptor

           }

       }

    }

  }

  delay(10);

}

Como se observa el código cada vez que vamos a escribir o leer datos activamos o desactivamos respectivamente el pin que va conectado a DE y RE del módulo RS485.

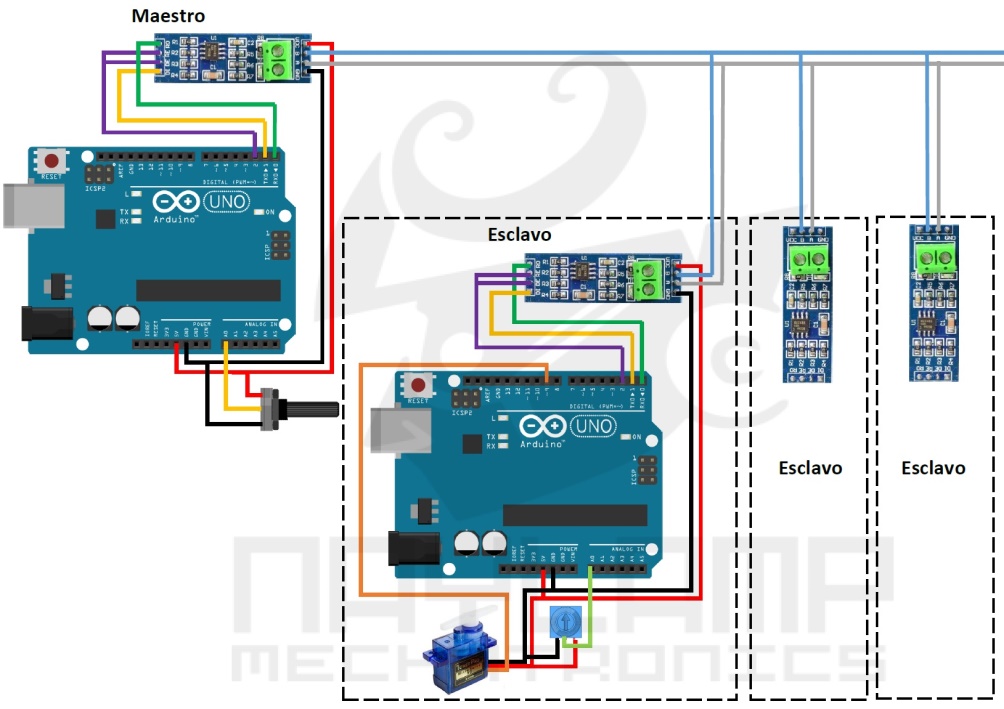
EL maestro siempre tiene activo la línea como transmisión pudiendo escribir en cualquier momento, mientras que el esclavo siempre está en modo recepción, escuchando los datos que le lleguen. Cuando el maestro necesite una respuesta por parte del esclavo, después de enviar la consulta, debe cambiar a modo de receptor para que el esclavo puede usar el canal para transmitir los datos, finalizado la respuesta, el maestro nuevamente debe tener libre el canal para transmitir.

## ****Ej 4. Comunicación half dúplex entre varios arduinos.****

Esta es la configuración más común que se usa, todos los Arduinos están conectados al mismo bus RS485, Un Arduino es maestro y todos los demás son Esclavos. Cada esclavo tiene una dirección el cual le identifica, el maestro para que pueda comunicarse con un esclavo usa esta dirección. El maestro envía la información por el bus y solo el esclavo con la dirección correcta es quien interpreta o procesa los datos.

A continuación mostramos el mismo ejemplo que se está trabajando anteriormente, pero enfocado une una conexión multipunto.

Las conexiones serían las siguientes



El ejemplo solo se muestra para un esclavo pero para los otros esclavos las conexiones son las mismas y en el código solo hay que tener en cuenta las direcciones de los esclavos.

La diferencia con los casos anteriores es en la trama para la comunicación, ahora es necesario enviar la dirección del esclavo, quedando la trama de la siguiente forma:

***[Inicio de trama][Dirección][Función][Valor][Fin de trama]***

La dirección es un número entero e indica con cual esclavo nos queremos comunicar. Ejemplos de esta trama serian: “I101S90F”,”I25S120F”,”I223LF”, etc.

 El esclavo interpreta esta trama y si coincide su dirección entonces realiza la función correspondiente, si es una función que necesite responder, en la trama de respuesta también agrega su dirección, esto para que el maestro sepa que ha respondido el esclavo correspondiente. La trama de respuesta es de la forma:

***[Inicio de trama][Dirección] [, ] [Valor][Fin de trama]***

En este caso entre la dirección y el valor enviamos una coma como separador puesto que ambos son valores numéricos, unos ejemplo de trama serian: “i101,865f”, “i26,64f”

Código del Maestro:

const int ledPin = 13; // Numero del pin para el Led

const int EnTxPin = 2; // HIGH:TX y LOW:RX

void **setup**()

{

**Serial**.begin(9600);

**Serial**.setTimeout(100); //establecemos un tiempo de espera de 100ms

  // inicializamos los pines

  pinMode(ledPin, OUTPUT);

  pinMode(EnTxPin, OUTPUT);

  digitalWrite(ledPin, LOW);

  digitalWrite(EnTxPin, HIGH); //RS485 como Transmisor

}

void **loop**()

{

  int lectura = analogRead(0);//leemos el valor del potenciómetro (de 0 a 1023)

  int angulo= map(lectura, 0, 1023, 0, 180);// escalamos la lectura a un valor de ángulo (entre 0 y 180)

  //---enviamos el ángulo para mover el servo------

**Serial**.print("I"); //inicio de trama

**Serial**.print("101");//dirección del esclavo

**Serial**.print("S"); //función S para indicarle que vamos a mover el servo

**Serial**.print(angulo); //ángulo o dato

**Serial**.print("F"); //fin de trama

  //----------------------------

  delay(50);

  //---solicitamos una lectura del sensor----------

**Serial**.print("I"); //inicio de trama

**Serial**.print("101");//direccion del esclavo

**Serial**.print("L"); //L para indicarle que vamos a Leer el sensor

**Serial**.print("F"); //fin de trama

**Serial**.flush(); //Esperamos hasta que se envíen los datos

  //----Leemos la respuesta del Esclavo-----

  digitalWrite(EnTxPin, LOW); //RS485 como receptor

  if(**Serial**.find("i")) //esperamos el inicio de trama

  {

      int esclavo=**Serial**.parseInt(); //recibimos la direccion del esclavo

      int dato=**Serial**.parseInt(); //recibimos el dato

      if(**Serial**.read()=='f'&&esclavo==101) //si fin de trama y direccion son los correctos

      {

         funcion(dato);   //realizamos la acción con el dato recibido

      }

  }

      digitalWrite(EnTxPin, HIGH); //RS485 como Transmisor

  //----------fin de la respuesta----------

}

void funcion(int dato)

{

  if(dato>500)

  digitalWrite(ledPin, HIGH);

  else

  digitalWrite(ledPin, LOW);

}

Código del Esclavo:

#include <Servo.h>

Servo myservo; // creamos el objeto servo

const int EnTxPin = 2; // HIGH:TX y LOW:RX

const int mydireccion =101; //Direccion del esclavo

void **setup**()

{

**Serial**.begin(9600);

**Serial**.setTimeout(100); //establecemos un tiempo de espera de 100ms

  myservo.attach(9); // asignamos el pin 9 para el servo.

  pinMode(EnTxPin, OUTPUT);

  digitalWrite(EnTxPin, LOW); //RS485 como receptor

}

void **loop**()

{

  if(**Serial**.available())

  {

    if(**Serial**.read()=='I') //Si recibimos el inicio de trama

    {

        int direccion=**Serial**.parseInt(); //recibimos la direccion

        if(direccion==mydireccion) //Si direccion es la nuestra

        {

            char funcion=**Serial**.read(); //leemos el carácter de función

            //---Si el carácter de función es una S entonces la trama es para mover el motor-----------

            if(funcion=='S')

             {

                 int angulo=**Serial**.parseInt(); //recibimos el ángulo

                 if(**Serial**.read()=='F') //Si el fin de trama es el correcto

                 {

                   if(angulo<=180) //verificamos que sea un valor en el rango del servo

                    {

                      myservo.write(angulo); //movemos el servomotor al ángulo correspondiente.

                    }

                 }

             }

             //---Si el carácter de función es L entonces el maestro está solicitando una lectura del sensor---

             else if(funcion=='L')

             {

                if(**Serial**.read()=='F') //Si el fin de trama es el correcto

                 {

                   int lectura = analogRead(0); //realizamos la lectura del sensor

                   digitalWrite(EnTxPin, HIGH); //rs485 como transmisor

**Serial**.print("i"); //inicio de trama

**Serial**.print(mydireccion); //direccion

**Serial**.print(",");

**Serial**.print(lectura); //valor del sensor

**Serial**.print("f"); //fin de trama

**Serial**.flush(); //Esperamos hasta que se envíen los datos

                   digitalWrite(EnTxPin, LOW); //RS485 como receptor

                 }

             }

        }

    }

  }

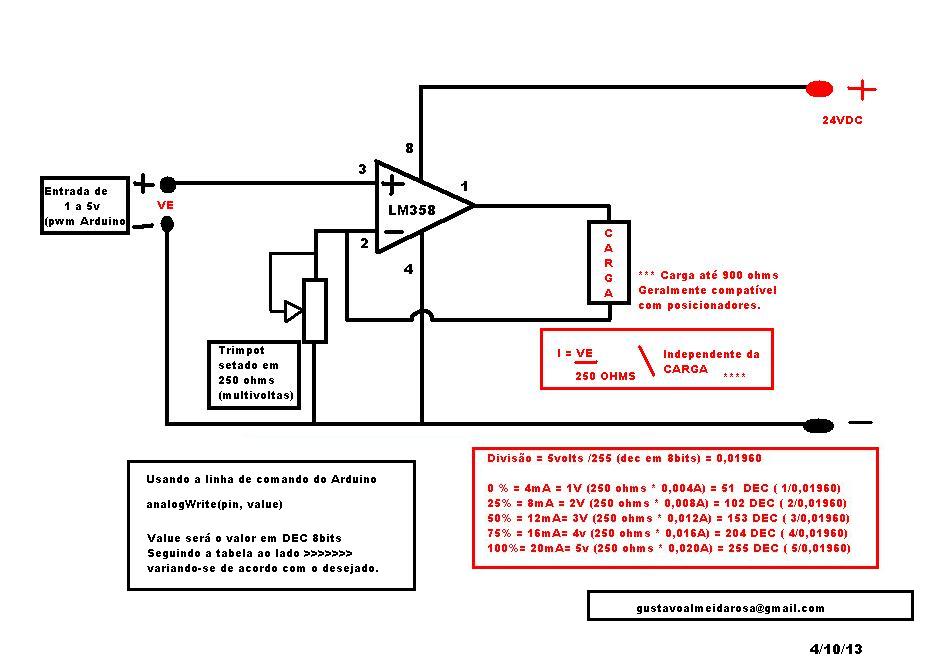
  delay(10);

}

El código para un segundo esclavo si se desea que tenga la misma función, en el código solo debe cambiar su dirección.

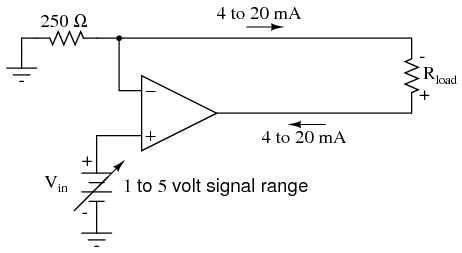
En un próximo tutorial implementaremos el protocolo Modbus en Arduino.

Circuito para Arduino y señal de 4-20 ma.



[**Convertidor Tensión-Corriente 0-5v 0-20mA**](https://www.arde.cc/foro/viewtopic.php?t=4110&start=10#p41112)

* [**Quote**](https://www.arde.cc/foro/posting.php?mode=quote&f=3&p=41112&sid=8fa0360a065422354b09a162f9466479)

si metes 0-5V salen 0-20mA, si metes 1-5V salen 4-20mA:  
  
  
  
Si se queda corto de corriente el operacional puedes poner un mosfet: [http://www.daycounter.com/Circuits/Curr ... ervo.phtml](http://www.daycounter.com/Circuits/Current-Servo/Current-Servo.phtml)" onclick="window.open(this.href);return false;

# Conversor voltaje a corriente



Modelo **MOD-V2C**

Transmisor de voltaje a corriente, usado para enviar señales analógicas y evitar la caída de voltaje en largas distancias.

Final del formulario

### DESCRIPCIÓN

### INFO

En la industria, cuando se quiere representar cantidades físicas de manera analógica, las señales en corriente DC son frecuentemente usadas sobre las señales en voltaje DC. La corriente en un circuito en serie es absolutamente igual en cualquier punto sin importar la resistencia del cableado, mientras que el voltaje en un circuito paralelo puede variar de extremo a extremo debido a la caída de voltaje en el cableado, esto hace que las señales en corriente sean mas precisas desde el instrumento emisor hasta el receptor.

Las señales en corriente son mas difíciles de producir que las de voltaje, este módulo te facilita el trabajo con un circuito que incluye un opamp que convierte el voltaje de entrada en una corriente de salida, además se puede regular la corriente mínima y corriente máxima. Usualmente la corriente mínima se regula en 4mA y la máxima en 20mA, esto es para que el receptor pueda detectar 0mA como una falla del conversor.  
Para usar el módulo simplemente aliméntalo con los pines de Vcc y GND, el pin OUT generará la señal en corriente correspondiente al voltaje en los pines VIN+ y GND, los potenciometros regulan la corriente mínima y máxima.

### ESPECIFICACIONES TÉCNICAS

* Voltaje de alimentación: 7 - 30 VDC
* Voltaje de entrada de señal de voltaje: 0-5VDC
* Rango de corriente de salida: 0-20 mA
* Temperatura de funcionamiento: -10 - 60 ºC

### LINKS

* [Datasheet del opamp usado](http://www.st.com/content/ccc/resource/technical/document/datasheet/bd/fc/46/43/26/8f/40/7f/CD00001046.pdf/files/CD00001046.pdf/jcr:content/translations/en.CD00001046.pdf)
* [Fundamentos de transmision en corriente](https://www.predig.com/indicatorpage/back-basics-fundamentals-4-20-ma-current-loops)

# Módulo Convertidor De Señal Voltaje A Corriente 5v 4 - 20 Ma

Principio del formulario

$150



1

1

I have a Mechanical Engineering Master Thesis about positioning control of a pneumatic system with Model Predictive Control. I am primarily an electrical engineer and i do not know much about electronics. I need to design an interface card in order to connect a valve (5/3 way), two pressure sensors, a linear scale to my NI-PCI-6010 data acquisition card.

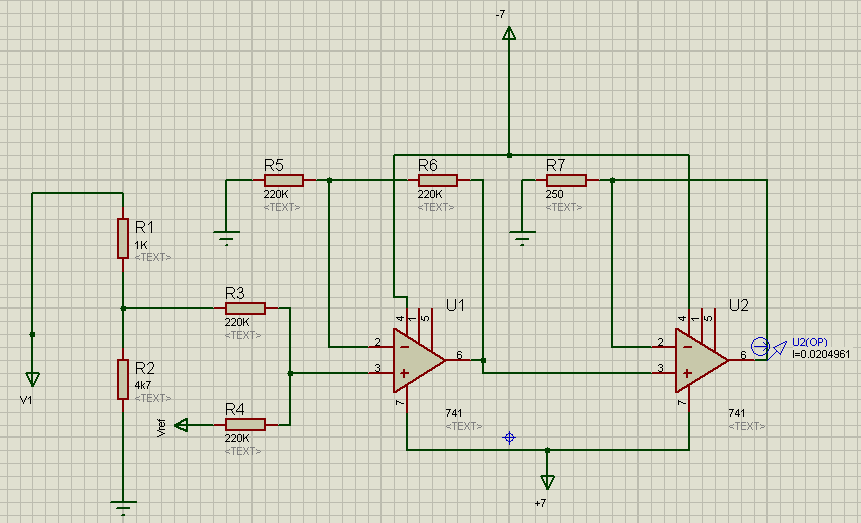
I have a bunch of questions about them. I tried to find solutions about them but none of them is enough for me. I wish you can help me.

However proportional valve (<http://xdki.festo.com/xdki/data/doc_ENGB/PDF/EN/MPYE_EN.PDF>) needs to be driven with 4-20 mA, my PCI-6010 card (<http://www.ni.com/pdf/manuals/371498a.pdf>) doesn't support it as i understand. Its analog output have a range of -5 to +5 V, i need to convert it into 4-20 mA.

First of all, do I need to convert -+5V or can I use 0-5V instead? I have seen lots of voltage-to-current converters in other websites, but all of them is about converting 1-5 V to 4-20 mA, I think there is an offset problem for 0-5V. I have found a non-inverting summing amplifier that can do what I want, but when input voltage reaches 5V, output current reaches 20,496 mA and this is surely a problem, also i need linearity.

I have found an IC that can convert 0-5 V to 4-20 mA (XTR110, I think you all have a word to say about it), but I cannot wait to purchase it, because i couldn't find any distributor in my country. So that I want to design a PCB in order to get over this problem.

Also I don't want to use a passive voltage-to-current converter. Here is the circuit I designed.



V1 = 5 V, Vref = 1 V (for offset problem)

Also; when V1 = 0V, output current is 4,037 mA. But it is not a big problem. Other values are;

V1=1 V, Iout=7,329 mA

V1=2 V, Iout=10,62 mA

**V1=2.5 V, Iout=12,267 mA**

V1=3 V, Iout=13,912 mA

V1=4 V, Iout=17,204 mA

As you see, I cannot achieve a setpoint value for my valve too (It needs 12 mA for middle position).

Of course I don't want you to do my thesis for me, but I stuck on early stage of my thesis and if I cannot found a solution about it I absolutely have a real bad time.

Programa Analizador de Protocolos <https://docklight.de/downloads/>

Redes Inalámbricas

<https://www.arduineando.com/xbee-y-arduino/>

<https://www.electronicosonline.com/arduino-se-empodera-y-lanza-su-plataforma-de-desarrollo-iot-cloud/>

<https://www.xataka.com/internet-of-things/las-3-tecnologias-clave-para-el-internet-de-las-cosas>

<https://polaridad.es/programar-un-modulo-wifi-esp8266-desde-el-ide-de-arduino/>

comunicación serial I2c

memoria

<https://www.youtube.com/watch?v=J3nuIL2dBak>

2 Arduinos

<https://www.youtube.com/watch?v=Jndb2vpAWwU>

Armando una red i2c

<https://www.youtube.com/watch?v=6IAkYpmA1DQ>

HMI

<https://electroniclinicall.blogspot.com/2018/08/arduino-mega-hmi-touch-screen-7-inch.html>

<https://slideplayer.es/slide/12702478/release/woothee>

Recopilación de Armando Mtz. R. para alumnos del ITNL. Febrero 2019