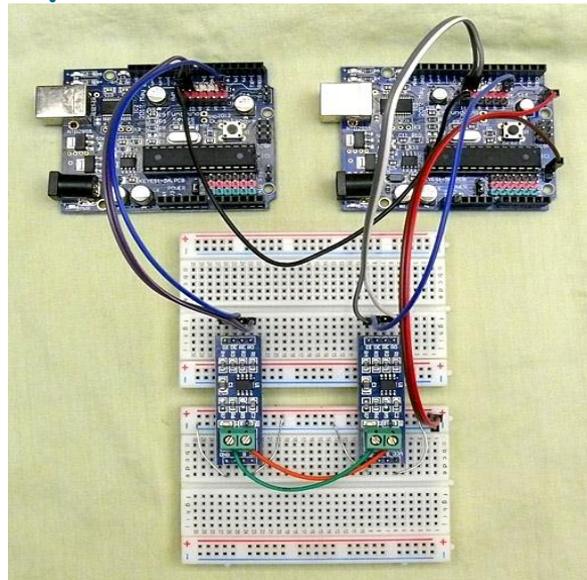


SoftwareSerialRS485Example

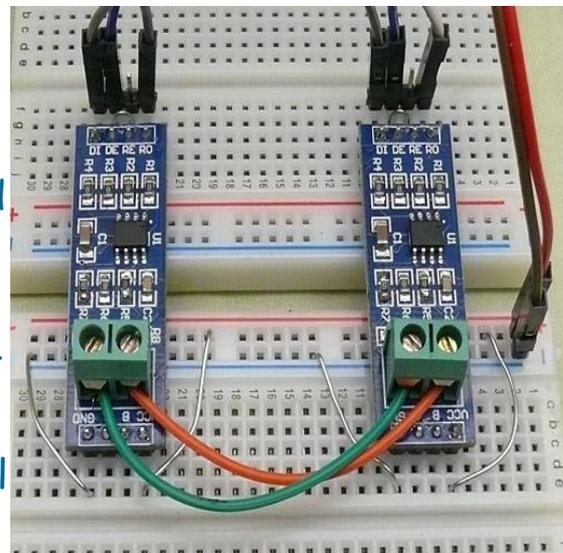
RS485 is a physical connection standard for Serial data transfer. It can be used between 2 or more Arduinos and to various I/O devices that support it.

The example on the right shows two Arduinos connected with low-cost RS485 "Electronic Bricks" or "breakout boards" that use a MAX485 chip. [See it HERE:](#)



More details [HERE](#).

RS485 uses a pair of wires in a differential configuration that allows long distance communication (up to 1.2 Km) and very high speeds. It also allows multiple senders/receivers along the same cable: "**Multi-Drop**". It also needs Ground - but that usually can be the local electrical ground, so only the two wires (usually a twisted pair in some type of cable) are needed.



In the example these are the red and green wires connected to **A** and **B** at the bottom. Also note the connection details on the right photo and refer to the pin connections shown in

the software examples below:

- DI (data in) to pin 11
- RO (receive out) to pin 10
- DE (data enable) and RE (receive enable) jumpered together and to pin 3
- Vcc and Gnd connected
- A and B : the RS485 pair
- Gnd between distant Arduinos can be in cable or local electrical ground.

RS485 in this example we show is **Half-Duplex**. This means it can send or receive on the same wires, but in only one direction at a time. See more about these modules and RS485 networks [HERE](#):

The example Software Sketches below use the Software Serial library that is provided with the Arduino IDE. This allows use of almost any Arduino pins for Serial Data connections, and allows simultaneous use of the standard Arduino USB Serial communications. The example here uses the Arduino serial monitor to allow you to type in data to send to the Master Arduino. Another serial port is created with the Software Serial library, using pins 11 and 12. This port sends the data to a second remote Arduino over RS485. That Arduino sends the same data back to the Master Arduino where it is sent back to the Serial Monitor for you, the user to view.

NOTE: A working system with your application running over multiple Arduinos should have some organized data

movement and error-checking. Nick Gammon has a nice library that does this

here:<http://www.gammon.com.au/forum/?id=11428>

The second Arduino runs the code in the second example below. You can copy and paste these into the Arduino IDE. (NOTE: You can run two copies of the Arduino software at once if you wish. Keep track of which Arduino is on which Serial Port!).

You use this by bringing up both connected Arduinos and running the Serial Monitor on the "Master". Type in the top window of the Serial Monitor and it should be echoed back as it is sent to the second Arduino and looped back. You should see the Pin13 LED on the Remote Arduino blink as the data is received and sent back.

Master Arduino Sketch:

```
/* YourDuino SoftwareSerialExample1
 - Connect to another Arduino running
"YD_SoftwareSerialExampleRS485_1Remote"
 - Connect this unit Pins 10, 11, Gnd
 - Pin 3 used for RS485 direction control
 - To other unit Pins 11,10, Gnd (Cross over)
 - Open Serial Monitor, type in top window.
 - Should see same characters echoed back from remote Arduino

Questions: terry@yourduino.com
*/

/*-----( Import needed libraries )-----*/
#include <SoftwareSerial.h>
/*-----( Declare Constants and Pin Numbers )-----*/
#define SSerialRX      10 //Serial Receive pin
#define SSerialTX      11 //Serial Transmit pin

#define SSerialTxControl 3 //RS485 Direction control

#define RS485Transmit   HIGH
#define RS485Receive    LOW

#define Pin13LED        13
```

```

/*-----( Declare objects )-----*/
SoftwareSerial RS485Serial(SSerialRX, SSerialTX); // RX, TX

/*-----( Declare Variables )-----*/
int byteReceived;
int byteSend;

void setup()   /***** SETUP: RUNS ONCE *****/
{
  // Start the built-in serial port, probably to Serial Monitor
  Serial.begin(9600);
  Serial.println("YourDuino.com SoftwareSerial remote loop example");
  Serial.println("Use Serial Monitor, type in upper window, ENTER");

  pinMode(Pin13LED, OUTPUT);
  pinMode(SSerialTxControl, OUTPUT);

  digitalWrite(SSerialTxControl, RS485Receive); // Init Transceiver

  // Start the software serial port, to another device
  RS485Serial.begin(4800); // set the data rate

} //--(end setup )---

void loop()   /***** LOOP: RUNS CONSTANTLY *****/
{
  digitalWrite(Pin13LED, HIGH); // Show activity
  if (Serial.available())
  {
    byteReceived = Serial.read();

    digitalWrite(SSerialTxControl, RS485Transmit); // Enable RS485
    RS485Serial.write(byteReceived); // Send byte to Remote
    Arduino

    digitalWrite(Pin13LED, LOW); // Show activity
    delay(10);
    digitalWrite(SSerialTxControl, RS485Receive); // Disable RS485
    Transmit
  }

  if (RS485Serial.available()) //Look for data from other Arduino
  {
    digitalWrite(Pin13LED, HIGH); // Show activity
    byteReceived = RS485Serial.read(); // Read received byte
    Serial.write(byteReceived); // Show on Serial Monitor
    delay(10);
    digitalWrite(Pin13LED, LOW); // Show activity
  }

} //--(end main loop )---

/*-----( Declare User-written Functions )-----*/

//NONE
//***** ( THE END )*****

```

Remote Arduino Sketch:

```
/* YourDuino SoftwareSerialExample1Remote
 - Used with YD SoftwareSerialExampleRS485 1 on another Arduino
 - Remote: Receive data, loop it back...
 - Connect this unit Pins 10, 11, Gnd
 - To other unit Pins 11,10, Gnd (Cross over)
 - Pin 3 used for RS485 direction control
 - Pin 13 LED blinks when data is received

  Questions: terry@yourduino.com
*/

/*-----( Import needed libraries )-----*/
#include <SoftwareSerial.h>
/*-----( Declare Constants and Pin Numbers )-----*/
#define SSerialRX      10 //Serial Receive pin
#define SSerialTX      11 //Serial Transmit pin

#define SSerialTxControl 3 //RS485 Direction control
#define RS485Transmit   HIGH
#define RS485Receive    LOW

#define Pin13LED        13

/*-----( Declare objects )-----*/
SoftwareSerial RS485Serial(SSerialRX, SSerialTX); // RX, TX

/*-----( Declare Variables )-----*/
int byteReceived;
int byteSend;

void setup() //***** SETUP: RUNS ONCE *****/
{
  // Start the built-in serial port, probably to Serial Monitor
  Serial.begin(9600);
  Serial.println("SerialRemote"); // Can be ignored

  pinMode(Pin13LED, OUTPUT);
  pinMode(SSerialTxControl, OUTPUT);

  digitalWrite(SSerialTxControl, RS485Receive); // Init Transceiver

  // Start the software serial port, to another device
  RS485Serial.begin(4800); // set the data rate
}

void loop() //***** LOOP: RUNS CONSTANTLY *****/
{
  //Copy input data to output
  if (RS485Serial.available())
  {
```

```
byteSend = RS485Serial.read(); // Read the byte

digitalWrite(Pin13LED, HIGH); // Show activity
delay(10);
digitalWrite(Pin13LED, LOW);

digitalWrite(SSerialTxControl, RS485Transmit); // Enable RS485
Transmit
RS485Serial.write(byteSend); // Send the byte back
delay(10);
digitalWrite(SSerialTxControl, RS485Receive); // Disable RS485
Transmit
// delay(100);
} // End If RS485SerialAvailable

} // --(end main loop )---

/*-----( Declare User-written Functions )-----*/
//NONE

//***** ( THE END )*****
```