MASTER – COM SPI

```
#include <SPI.h>


void setup (void)

{


//INICIALIZACIÓN MASTER

 Serial.begin(9600);

 digitalWrite(SS, HIGH); //Deshabilitamos el Slave

 SPI.begin ();

 SPI.setClockDivider(SPI_CLOCK_DIV16); //Dividimos la velocidad del reloj para que el bit sea más estable (1MHz)

}


void loop (void)

{

 char c;


 digitalWrite(SS, LOW);    //Habilitamos el Slave en el pin 53


 for (const char * string = "HOLA\n" ; c = *string; string++) {// Recorremos todo el string

   SPI.transfer (c); //Enviamos caracter a caracter

   Serial.print(c); //Escribimos en pantalla todos los caracteres enviados

 }


 digitalWrite(SS, HIGH);//Deshabilitamos el Slave

 delay (1000);

}
```

SLAVE – COM SPI

```
#include <SPI.h>


char buf [150];

char a;

volatile byte pos;

volatile boolean recibido;


void setup (void)

{

//INICIALIZACIÓN SLAVE

 Serial.begin (9600);

 pinMode(MISO, OUTPUT);//Configuramos el Miso (MASTER INPUT SLAVE OUTPUT) como salida ya que el Master recibira por aqui.

 SPCR |= _BV(SPE);//Activamos el modo Slave: registro maestro, es decir, contiene los bits para inicializar SPI y controlarlo.

 pos = 0;   // Posicion 0 de los bytes recibidos

 recibido = false; //Inicializamos que no hemos recibido.

 SPI.attachInterrupt();//Activamos la interrupcion del SPI

}

// Interrupcion SPI

ISR (SPI_STC_vect)

{

 byte c = SPDR;  //Obtenemos el byte: Este es el registro de estado. Este registro se utiliza para leer el estado de las líneas de bus.


   buf [pos++] = c;

   if (c == '\n')

   recibido = true;

}
```

```
void loop (void)

{

  if (recibido)

  {

    buf [pos] = 0;

    Serial.print ("RECIBIDO: ");

    Serial.println (buf);

    pos = 0;


    recibido = false;


  }

}
```