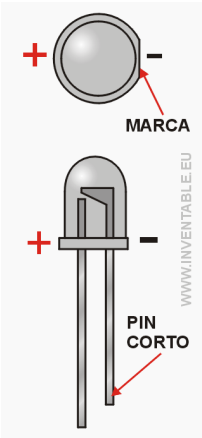
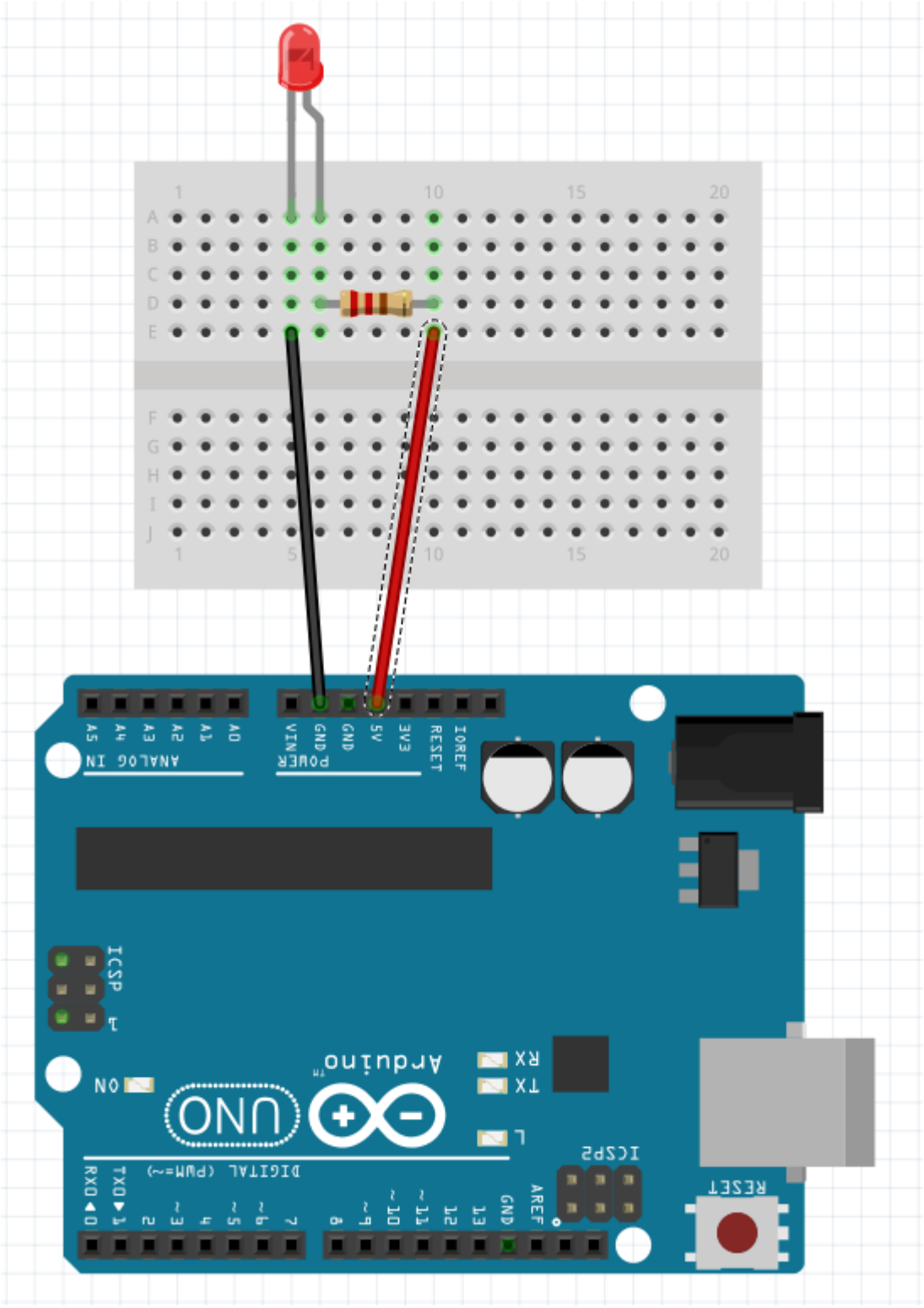
Marca Led Negativo

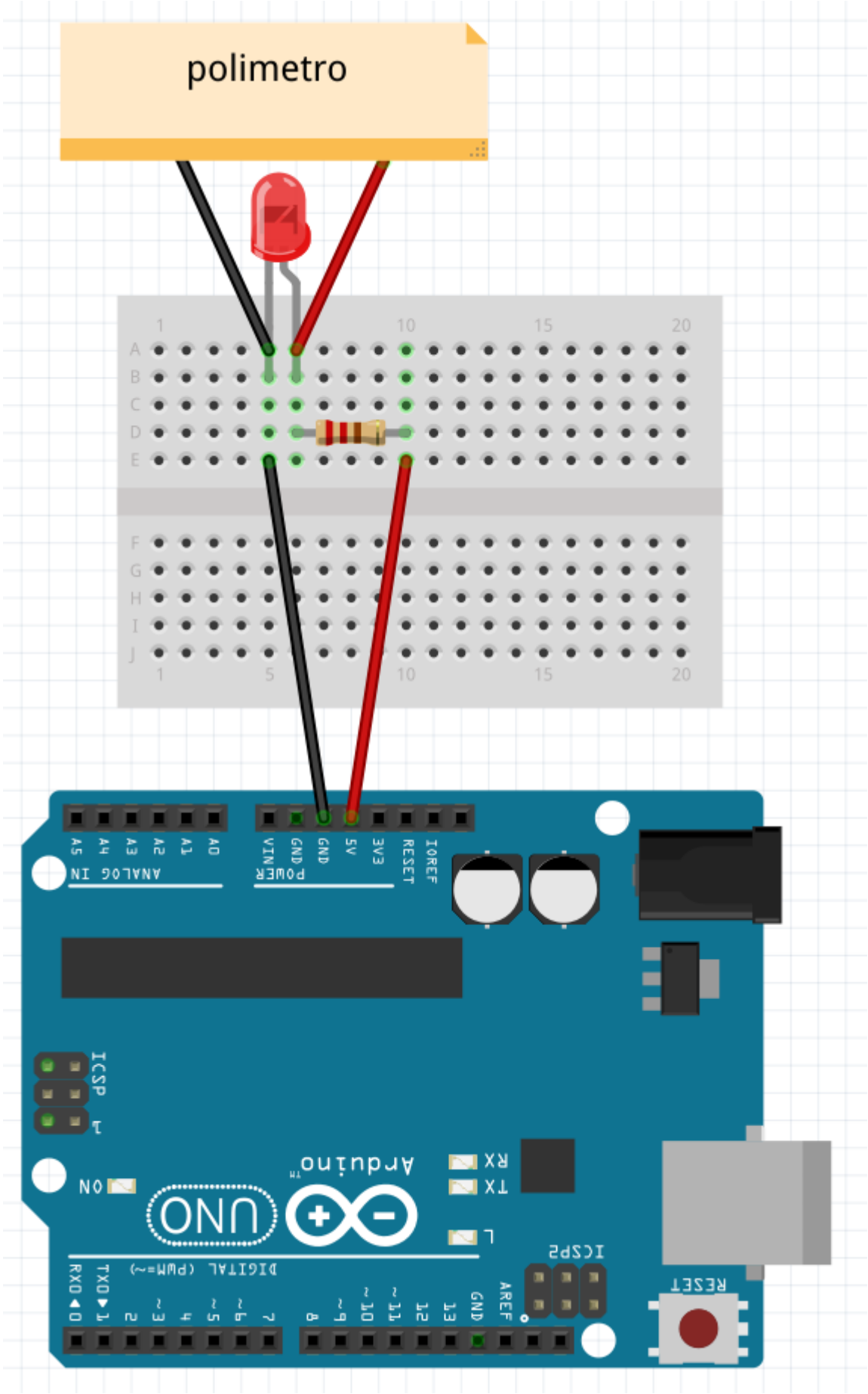


Hay que poner una resistencia de unos 330mA si lo conectamos a 5V

Así se enciende directamente:



Para calcular la resistencia exacta, primero calculamos el voltaje del led poniendo un polimetro en paralelo:



Si en vez de 5V ponemos una pila de 9V por ejemplo, ponemos una reistencia de 900Ω por si acaso.

En mi caso me da un led rojo 1,9v y uno azul 3v

Paneles led

http://www.minitronica.com/producto/matriz-led-de-8x8-con-max7219/

http://tienda.bricogeek.com/componentes/124-matriz-de-led-r-g-mediana-con-interfaz-serie.html

https://www.sparkfun.com/products/11861

http://www.dx.com/es/p/1-8-8x8-seamless-cascadable-red-led-matrix-display-module-with-spi-interface-for-arduino-408734

MAX7219 Matrix 8x32

Este funciona: https://microcontrollerslab.com/led-dot-matrix-display-esp32-max7219/

Aquí hay ejemplos sin probar: https://www.circuitgeeks.com/arduino-max7219-led-matrix-display/

MAX7219 Matrix 8x8

Minitutorial:
http://www.prometec.net/scroll-max7219/

Nos descargamos la biblioteca, que no librería, de aquí:
https://code.google.com/archive/p/arudino-maxmatrix-library/downloads

Desde el IDE de arduino, en sketch, import library la añadimos.

CS - 10
CLK - 11
DIN -12

maxInUse es el número de paneles conectados.

```
#include <MaxMatrix.h>
//#include <avr/pgmspace.h>

PROGMEM const byte CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
```

```
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
```

```
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0101000, B0000000, // c
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};

int data = 12;
int load = 10;
int clock = 11;
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll
use
MaxMatrix m(data, load, clock, maxInUse);
byte buffer[10];
char msg[] = "19";

void setup()
{
  m.init();
  m.setIntensity(5);
```

```
  printString(msg);
}


void loop()
{
}

void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

Texto con Scroll

http://www.minitronica.com/blog/matrices-led-8x8-arduino-max7219/

```
// INCLUIMOS LA LIBRERÍA MAXMATRIX.H Y PGMSPACE.H NECESARIO
// PARA PODER USAR EL MODIFICADOR DE VARIABLE PROGMEM
#include <MaxMatrix.h>
#include <avr/pgmspace.h>


PROGMEM const byte CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
```

```
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
```

```
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0101000, B0000000, // c
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};


// DEFINIMOS LOS PINES DEL ARDUINO
int data = 12;    // Pin DIN del módulo MAX7219
int load = 10;    // Pin CS del módulo MAX7219
int clock = 11;  // Pin CLK del módulo MAX7219

// CUANTOS MÓDULOS TENEMOS CONECTADOS EN SERIE?
int maxInUse = 2;

// DEFINIMOS LA FUNCIÓN DE CADA PIN
MaxMatrix m(data, load, clock, maxInUse);
byte buffer[10];

// ESTA ES LA VARIABLE QUE CONTIENE EL TEXTO QUE APARECERÁ EN LOS DISPLAYS
//char string1[] = "Esto es una prueba de texto en scroll          ";
char string1[] = "ESTO ES UNA PRUEBA    ";


void setup(){
```

```
  m.init();          // INICIAMOS EL MODULO
  m.setIntensity(5); // DEFINIMOS LA INTENSIDAD DE LOS LED (0-15)
}

void loop(){
  byte c;
  delay(100);  // PAUSA ENTRE MOVIMIENTOS
  m.shiftLeft(false, true);
  printStringWithShift(string1, 100);  // ENVIAMOS EL TEXTO A LOS MODULOS
}

// FUNCIONES PARA MOSTRAR LOS CARACTERES EN EL DISPLAY DE LOS MODULOS
void printCharWithShift(char c, int shift_speed){
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
  m.writeSprite(maxInUse*8, 0, buffer);
  m.setColumn(maxInUse*8 + buffer[0], 0);
  for (int i=0; i<buffer[0]+1; i++)
  {
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}

void printStringWithShift(char* s, int shift_speed){
  while (*s != 0){
    printCharWithShift(*s, shift_speed);
    s++;
  }
}
```

Contador de números del 1 al 10

Como convertir número a cadena: sprintf:

https://programarfacil.com/blog/arduino-blog/conversion-de-numeros-a-cadenas-en-arduino/

```
#include <MaxMatrix.h>
//#include <avr/pgmspace.h>

PROGMEM const byte CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
```

```
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
```

```
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0101000, B0000000, // c
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};

int data = 12;
int load = 10;
int clock = 11;
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll
use
MaxMatrix m(data, load, clock, maxInUse);
byte buffer[10];
void setup()
{
  Serial.begin(115200);
  m.init();
```

```
  m.setIntensity(5);
}


void loop()
{
   for (int i=0; i <= 10; i++)
   {
     char msg[2];
     sprintf (msg,"%i ", i);
     printString(msg);
     Serial.print(i);
     delay(1000);
   }
   printString("x");
   delay(1000);
}

void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

Lee variable de serial

Con un 1 suma el primer dígito y con un 2 suma el segundo

```
#include <MaxMatrix.h>
//#include <avr/pgmspace.h>

PROGMEM const byte CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
```

```
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
```

```
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0101000, B0000000, // c
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};

int data = 12;
int load = 10;
int clock = 11;
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll
use
MaxMatrix m(data, load, clock, maxInUse);
byte buffer[10];
void setup()
{
  Serial.begin(115200);
  m.init();
  m.setIntensity(5);
```

```
}
int estado=0;
int i;

void loop()
{
     if(Serial.available()>0)
     {
       estado = Serial.read();
     }
     if (estado == '1')
     {
       i=i+10;
     }
     if (estado == '2')
     {
       i=i+1;
     }
     char msg[1];
     sprintf (msg,"%i ", i);
     printString(msg);
     delay(1000);
     estado=0;
}

void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

Marcador con dos displays

```
#include <MaxMatrix.h>
#include <SoftwareSerial.h>
SoftwareSerial BT1(4,2);      // RX, TX recorder que se cruzan
//#include <avr/pgmspace.h>

PROGMEM const byte CH[] = {
```

```
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
```

```
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0101000, B0000000, // c
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};

int data = 12;
int load = 10;
int clock = 11;
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll
use
```

```
MaxMatrix m(data, load, clock, maxInUse);

int vdata = 9;
int vload = 7;
int vclock = 8;
int vmaxInUse = 1;    //change this variable to set how many MAX7219's
you'll use
MaxMatrix v(vdata, vload, vclock, vmaxInUse);

byte buffer[10];
void setup()
{
  Serial.begin(9600);
  BT1.begin(9600);
  m.init();
  m.setIntensity(5);
  v.init();
  v.setIntensity(5);
}
int estado=0;
int i;
int j;

void loop()
{
     if(BT1.available())
     {
       estado = BT1.read();
     }
     if (estado == '1')
     {
       i=i+1;
       char msg[1];
       sprintf (msg,"%i ", i);
       printString(msg);
       delay(1000);
       estado=0;
     }
     if (estado == '2')
     {
       j=j+1;
       char msg[1];
       sprintf (msg,"%i ", j);
       vprintString(msg);
       delay(1000);
       estado=0;
     }
     if (estado == '0')
     {
       printString("0");
       vprintString("0");
```

```
       i=0;
       j=0;
     }

}

void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}

void vprintString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    v.writeSprite(col, 0, buffer);
    v.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

3 y 4 borra y bug 0 arreglado

```
#include <MaxMatrix.h>
#include <SoftwareSerial.h>
SoftwareSerial BT1(4,2);      // RX, TX recorder que se cruzan
//#include <avr/pgmspace.h>

PROGMEM const byte CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
```

```
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
```

```
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0101000, B0000000, // c
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};

int data = 12;
int load = 10;
int clock = 11;
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll
use
MaxMatrix m(data, load, clock, maxInUse);

int vdata = 9;
int vload = 7;
int vclock = 8;
int vmaxInUse = 1;    //change this variable to set how many MAX7219's
you'll use
```

```
MaxMatrix v(vdata, vload, vclock, vmaxInUse);

byte buffer[10];
void setup()
{
  Serial.begin(9600);
  BT1.begin(9600);
  m.init();
  m.setIntensity(5);
  v.init();
  v.setIntensity(5);
}
int estado=0;
int i;
int j;

void loop()
{
     if(BT1.available())
     {
       estado = BT1.read();
     }
     if (estado == '2')
     {
       i=i+1;
       char msg[1];
       sprintf (msg,"%i ", i);
       printString(msg);
       delay(1000);
       estado=0;
     }
     if (estado == '1')
     {
       j=j+1;
       char msg[1];
       sprintf (msg,"%i ", j);
       vprintString(msg);
       delay(1000);
       estado=0;
     }
     if (estado == '4')
     {
       i=i-1;
       char msg[1];
       sprintf (msg,"%i ", i);
       printString(msg);
       delay(1000);
       estado=0;
     }
     if (estado == '3')
     {
```

```
       j=j-1;
       char msg[1];
       sprintf (msg,"%i ", j);
       vprintString(msg);
       delay(1000);
       estado=0;
     }
     if (estado == '0')
     {
       printString("0 ");
       vprintString("0 ");
       i=0;
       j=0;
     }

}

void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}

void vprintString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    v.writeSprite(col, 0, buffer);
    v.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```