#**include** <LedControl.h> //inport the library, make sure to install it first

**String** message =""; //create an empty string to store the future received data

**const** **int** numDevices = 4; // number of MAX7219s used in this case 4

**const** **long** scrollDelay = 60; // adjust scrolling speed (lower dela -> higher scroll speed)

**unsigned** **long** bufferLong [14] = {0};

**int** initial = 0;

LedControl lc=LedControl(11,13,10,numDevices);//D11=DATA, D13=CLK, D10=LOAD //select the used pins

**const** **unsigned** **char** initialText[] PROGMEM ={"Open App and send text "}; //This will be the initial displayed text

/\*After you send a text using the smartphone (bluetooth) you have to wait till the

last text finish scrooling. Once it finish, your new text will be displayed. \*/

/\*Here we sabe on to the FLASH memory all the characters, icons and symbols.

If we use the SRAM memory we will have problems. \*/

**const** **unsigned** **char** scrollText0[] PROGMEM ={" "};

**const** **unsigned** **char** scrollText1[] PROGMEM ={"!"};

**const** **unsigned** **char** scrollText2[] PROGMEM ={"''"};

**const** **unsigned** **char** scrollText3[] PROGMEM ={"#"};

**const** **unsigned** **char** scrollText4[] PROGMEM ={"$"};

**const** **unsigned** **char** scrollText5[] PROGMEM ={"%"};

**const** **unsigned** **char** scrollText6[] PROGMEM ={"&"};

**const** **unsigned** **char** scrollText7[] PROGMEM ={"'"};

**const** **unsigned** **char** scrollText8[] PROGMEM ={"("};

**const** **unsigned** **char** scrollText9[] PROGMEM ={")"};

**const** **unsigned** **char** scrollText10[] PROGMEM ={"\*"};

**const** **unsigned** **char** scrollText11[] PROGMEM ={"+"};

**const** **unsigned** **char** scrollText12[] PROGMEM ={","};

**const** **unsigned** **char** scrollText13[] PROGMEM ={"-"};

**const** **unsigned** **char** scrollText14[] PROGMEM ={"."};

**const** **unsigned** **char** scrollText15[] PROGMEM ={"/"};

**const** **unsigned** **char** scrollText16[] PROGMEM ={"0"};

**const** **unsigned** **char** scrollText17[] PROGMEM ={"1"};

**const** **unsigned** **char** scrollText18[] PROGMEM ={"2"};

**const** **unsigned** **char** scrollText19[] PROGMEM ={"3"};

**const** **unsigned** **char** scrollText20[] PROGMEM ={"4"};

**const** **unsigned** **char** scrollText21[] PROGMEM ={"5"};

**const** **unsigned** **char** scrollText22[] PROGMEM ={"6"};

**const** **unsigned** **char** scrollText23[] PROGMEM ={"7"};

**const** **unsigned** **char** scrollText24[] PROGMEM ={"8"};

**const** **unsigned** **char** scrollText25[] PROGMEM ={"9"};

**const** **unsigned** **char** scrollText26[] PROGMEM ={":"};

**const** **unsigned** **char** scrollText27[] PROGMEM ={"<"};

**const** **unsigned** **char** scrollText28[] PROGMEM ={"="};

**const** **unsigned** **char** scrollText29[] PROGMEM ={">"};

**const** **unsigned** **char** scrollText30[] PROGMEM ={"?"};

**const** **unsigned** **char** scrollText31[] PROGMEM ={"@"};

**const** **unsigned** **char** scrollText32[] PROGMEM ={"A"};

**const** **unsigned** **char** scrollText33[] PROGMEM ={"B"};

**const** **unsigned** **char** scrollText34[] PROGMEM ={"C"};

**const** **unsigned** **char** scrollText35[] PROGMEM ={"D"};

**const** **unsigned** **char** scrollText36[] PROGMEM ={"E"};

**const** **unsigned** **char** scrollText37[] PROGMEM ={"F"};

**const** **unsigned** **char** scrollText38[] PROGMEM ={"G"};

**const** **unsigned** **char** scrollText39[] PROGMEM ={"H"};

**const** **unsigned** **char** scrollText40[] PROGMEM ={"I"};

**const** **unsigned** **char** scrollText41[] PROGMEM ={"J"};

**const** **unsigned** **char** scrollText42[] PROGMEM ={"K"};

**const** **unsigned** **char** scrollText43[] PROGMEM ={"L"};

**const** **unsigned** **char** scrollText44[] PROGMEM ={"M"};

**const** **unsigned** **char** scrollText45[] PROGMEM ={"N"};

**const** **unsigned** **char** scrollText46[] PROGMEM ={"O"};

**const** **unsigned** **char** scrollText47[] PROGMEM ={"P"};

**const** **unsigned** **char** scrollText48[] PROGMEM ={"Q"};

**const** **unsigned** **char** scrollText49[] PROGMEM ={"R"};

**const** **unsigned** **char** scrollText50[] PROGMEM ={"S"};

**const** **unsigned** **char** scrollText51[] PROGMEM ={"T"};

**const** **unsigned** **char** scrollText52[] PROGMEM ={"U"};

**const** **unsigned** **char** scrollText53[] PROGMEM ={"V"};

**const** **unsigned** **char** scrollText54[] PROGMEM ={"W"};

**const** **unsigned** **char** scrollText55[] PROGMEM ={"X"};

**const** **unsigned** **char** scrollText56[] PROGMEM ={"Y"};

**const** **unsigned** **char** scrollText57[] PROGMEM ={"Z"};

**const** **unsigned** **char** scrollText58[] PROGMEM ={"["};

**const** **unsigned** **char** scrollText59[] PROGMEM ={"]"};

**const** **unsigned** **char** scrollText60[] PROGMEM ={"^"};

**const** **unsigned** **char** scrollText61[] PROGMEM ={"\_"};

**const** **unsigned** **char** scrollText62[] PROGMEM ={"´"};

**const** **unsigned** **char** scrollText63[] PROGMEM ={"a"};

**const** **unsigned** **char** scrollText64[] PROGMEM ={"b"};

**const** **unsigned** **char** scrollText65[] PROGMEM ={"c"};

**const** **unsigned** **char** scrollText66[] PROGMEM ={"d"};

**const** **unsigned** **char** scrollText67[] PROGMEM ={"e"};

**const** **unsigned** **char** scrollText68[] PROGMEM ={"f"};

**const** **unsigned** **char** scrollText69[] PROGMEM ={"g"};

**const** **unsigned** **char** scrollText70[] PROGMEM ={"h"};

**const** **unsigned** **char** scrollText71[] PROGMEM ={"i"};

**const** **unsigned** **char** scrollText72[] PROGMEM ={"j"};

**const** **unsigned** **char** scrollText73[] PROGMEM ={"k"};

**const** **unsigned** **char** scrollText74[] PROGMEM ={"l"};

**const** **unsigned** **char** scrollText75[] PROGMEM ={"m"};

**const** **unsigned** **char** scrollText76[] PROGMEM ={"n"};

**const** **unsigned** **char** scrollText77[] PROGMEM ={"o"};

**const** **unsigned** **char** scrollText78[] PROGMEM ={"p"};

**const** **unsigned** **char** scrollText79[] PROGMEM ={"q"};

**const** **unsigned** **char** scrollText80[] PROGMEM ={"r"};

**const** **unsigned** **char** scrollText81[] PROGMEM ={"s"};

**const** **unsigned** **char** scrollText82[] PROGMEM ={"t"};

**const** **unsigned** **char** scrollText83[] PROGMEM ={"u"};

**const** **unsigned** **char** scrollText84[] PROGMEM ={"v"};

**const** **unsigned** **char** scrollText85[] PROGMEM ={"w"};

**const** **unsigned** **char** scrollText86[] PROGMEM ={"x"};

**const** **unsigned** **char** scrollText87[] PROGMEM ={"y"};

**const** **unsigned** **char** scrollText88[] PROGMEM ={"z"};

**const** **unsigned** **char** scrollText89[] PROGMEM ={"{"};

**const** **unsigned** **char** scrollText90[] PROGMEM ={"|"};

**const** **unsigned** **char** scrollText91[] PROGMEM ={"}"};

**const** **unsigned** **char** scrollText92[] PROGMEM ={"~"};

//We start our setup

**void** setup(){

Serial.begin(9600); //Start the serial comunication fot the bluetooth module

for (**int** x=0; x<numDevices; x++){

lc.shutdown(x,false); //The MAX72XX is in power-saving mode on startup

lc.setIntensity(x,8); // Set the brightness to default value

lc.clearDisplay(x); // and clear the display

}

}

//We start the infinite loop

**void** loop(){

//We start scrooling the initial text

if(initial==0)

{

scrollMessage(initialText);

}

while(Serial.available() > 0) // Don't read unless you receive something new

{

message = Serial.readString(); //Store the bluetooth received text

initial=1;

}

//Go letter by letter and send the dots vector for the LEDs to the drivers

for (**int** i=0; i < message.length(); i++)

{

if (message[i] == ' ')

{

scrollMessage(scrollText0);

}

if (message[i] == '!')

{

scrollMessage(scrollText1);

}

if (message[i] == '"')

{

scrollMessage(scrollText2);

}

if (message[i] == '#')

{

scrollMessage(scrollText3);

}

if (message[i] == '&')

{

scrollMessage(scrollText4);

}

if (message[i] == '%')

{

scrollMessage(scrollText5);

}

if (message[i] == '&')

{

scrollMessage(scrollText5);

}

if (message[i] == '(')

{

scrollMessage(scrollText8);

}

if (message[i] == ')')

{

scrollMessage(scrollText9);

}

if (message[i] == '\*')

{

scrollMessage(scrollText10);

}

if (message[i] == '+')

{

scrollMessage(scrollText11);

}

if (message[i] == ',')

{

scrollMessage(scrollText12);

}

if (message[i] == '-')

{

scrollMessage(scrollText13);

}

if (message[i] == '.')

{

scrollMessage(scrollText14);

}

if (message[i] == '/')

{

scrollMessage(scrollText15);

}

if (message[i] == '0')

{

scrollMessage(scrollText16);

}

if (message[i] == '1')

{

scrollMessage(scrollText17);

}

if (message[i] == '2')

{

scrollMessage(scrollText18);

}

if (message[i] == '3')

{

scrollMessage(scrollText19);

}

if (message[i] == '4')

{

scrollMessage(scrollText20);

}

if (message[i] == '5')

{

scrollMessage(scrollText21);

}

if (message[i] == '6')

{

scrollMessage(scrollText22);

}

if (message[i] == '7')

{

scrollMessage(scrollText23);

}

if (message[i] == '8')

{

scrollMessage(scrollText24);

}

if (message[i] == '9')

{

scrollMessage(scrollText25);

}

if (message[i] == ':')

{

scrollMessage(scrollText26);

}

if (message[i] == '<')

{

scrollMessage(scrollText27);

}

if (message[i] == '=')

{

scrollMessage(scrollText28);

}

if (message[i] == '>')

{

scrollMessage(scrollText29);

}

if (message[i] == '?')

{

scrollMessage(scrollText30);

}

if (message[i] == '@')

{

scrollMessage(scrollText31);

}

if (message[i] == 'A')

{

scrollMessage(scrollText32);

}

if (message[i] == 'B')

{

scrollMessage(scrollText33);

}

if (message[i] == 'C')

{

scrollMessage(scrollText34);

}

if (message[i] == 'D')

{

scrollMessage(scrollText35);

}

if (message[i] == 'E')

{

scrollMessage(scrollText36);

}

if (message[i] == 'F')

{

scrollMessage(scrollText37);

}

if (message[i] == 'G')

{

scrollMessage(scrollText38);

}

if (message[i] == 'H')

{

scrollMessage(scrollText39);

}

if (message[i] == 'I')

{

scrollMessage(scrollText40);

}

if (message[i] == 'J')

{

scrollMessage(scrollText41);

}

if (message[i] == 'K')

{

scrollMessage(scrollText42);

}

if (message[i] == 'L')

{

scrollMessage(scrollText43);

}

if (message[i] == 'M')

{

scrollMessage(scrollText44);

}

if (message[i] == 'N')

{

scrollMessage(scrollText45);

}

if (message[i] == 'O')

{

scrollMessage(scrollText46);

}

if (message[i] == 'P')

{

scrollMessage(scrollText47);

}

if (message[i] == 'Q')

{

scrollMessage(scrollText48);

}

if (message[i] == 'R')

{

scrollMessage(scrollText49);

}

if (message[i] == 'S')

{

scrollMessage(scrollText50);

}

if (message[i] == 'T')

{

scrollMessage(scrollText51);

}

if (message[i] == 'U')

{

scrollMessage(scrollText52);

}

if (message[i] == 'V')

{

scrollMessage(scrollText53);

}

if (message[i] == 'W')

{

scrollMessage(scrollText54);

}

if (message[i] == 'X')

{

scrollMessage(scrollText55);

}

if (message[i] == 'Y')

{

scrollMessage(scrollText56);

}

if (message[i] == 'Z')

{

scrollMessage(scrollText57);

}

if (message[i] == '[')

{

scrollMessage(scrollText58);

}

if (message[i] == '[')

{

scrollMessage(scrollText59);

}

if (message[i] == '^')

{

scrollMessage(scrollText60);

}

if (message[i] == '\_')

{

scrollMessage(scrollText61);

}

if (message[i] == 'a')

{

scrollMessage(scrollText63);

}

if (message[i] == 'b')

{

scrollMessage(scrollText64);

}

if (message[i] == 'c')

{

scrollMessage(scrollText65);

}

if (message[i] == 'd')

{

scrollMessage(scrollText66);

}

if (message[i] == 'e')

{

scrollMessage(scrollText67);

}

if (message[i] == 'f')

{

scrollMessage(scrollText68);

}

if (message[i] == 'g')

{

scrollMessage(scrollText69);

}

if (message[i] == 'h')

{

scrollMessage(scrollText70);

}

if (message[i] == 'i')

{

scrollMessage(scrollText71);

}

if (message[i] == 'j')

{

scrollMessage(scrollText72);

}

if (message[i] == 'k')

{

scrollMessage(scrollText73);

}

if (message[i] == 'l')

{

scrollMessage(scrollText74);

}

if (message[i] == 'm')

{

scrollMessage(scrollText75);

}

if (message[i] == 'n')

{

scrollMessage(scrollText76);

}

if (message[i] == 'o')

{

scrollMessage(scrollText77);

}

if (message[i] == 'p')

{

scrollMessage(scrollText78);

}

if (message[i] == 'q')

{

scrollMessage(scrollText79);

}

if (message[i] == 'r')

{

scrollMessage(scrollText80);

}

if (message[i] == 's')

{

scrollMessage(scrollText81);

}

if (message[i] == 't')

{

scrollMessage(scrollText82);

}

if (message[i] == 'u')

{

scrollMessage(scrollText83);

}

if (message[i] == 'v')

{

scrollMessage(scrollText84);

}

if (message[i] == 'w')

{

scrollMessage(scrollText85);

}

if (message[i] == 'x')

{

scrollMessage(scrollText86);

}

if (message[i] == 'y')

{

scrollMessage(scrollText87);

}

if (message[i] == 'z')

{

scrollMessage(scrollText88);

}

if (message[i] == '{')

{

scrollMessage(scrollText89);

}

if (message[i] == '|')

{

scrollMessage(scrollText90);

}

if (message[i] == '}')

{

scrollMessage(scrollText91);

}

if (message[i] == '~')

{

scrollMessage(scrollText92);

}

}

}

//////////////////////////////////////////////////////EDIT THE CHARACTERS///////////////////////////////////////////

**const** **unsigned** **char** font5x7 [] PROGMEM = { //Numeric Font Matrix (Arranged as 7x font data + 1x kerning data)

B00000000, //Space (Char 0x20)

B00000000,

B00000000,

B00000000,

B00000000,

B00000000,

B00000000,

3,//this number gives the empty space column amount between the characters

B10000000, //!

B10000000,

B10000000,

B10000000,

B00000000,

B00000000,

B10000000,

2,

B10100000, //"

B10100000,

B10100000,

B00000000,

B00000000,

B00000000,

B00000000,

4,

B01010000, //#

B01010000,

B11111000,

B01010000,

B11111000,

B01010000,

B01010000,

6,

B00100000, //$

B01111000,

B10100000,

B01110000,

B00101000,

B11110000,

B00100000,

6,

B11000000, //%

B11001000,

B00010000,

B00100000,

B01000000,

B10011000,

B00011000,

6,

B01100000, //&

B10010000,

B10100000,

B01000000,

B10101000,

B10010000,

B01101000,

6,

B11000000, //'

B01000000,

B10000000,

B00000000,

B00000000,

B00000000,

B00000000,

3,

B00100000, //(

B01000000,

B10000000,

B10000000,

B10000000,

B01000000,

B00100000,

4,

B10000000, //)

B01000000,

B00100000,

B00100000,

B00100000,

B01000000,

B10000000,

4,

B00000000, //\*

B00100000,

B10101000,

B01110000,

B10101000,

B00100000,

B00000000,

6,

B00000000, //+

B00100000,

B00100000,

B11111000,

B00100000,

B00100000,

B00000000,

6,

B00000000, //,

B00000000,

B00000000,

B00000000,

B11000000,

B01000000,

B10000000,

3,

B00000000, //-

B00000000,

B11111000,

B00000000,

B00000000,

B00000000,

B00000000,

6,

B00000000, //.

B00000000,

B00000000,

B00000000,

B00000000,

B11000000,

B11000000,

3,

B00000000, ///

B00001000,

B00010000,

B00100000,

B01000000,

B10000000,

B00000000,

6,

B01110000, //0

B10001000,

B10011000,

B10101000,

B11001000,

B10001000,

B01110000,

6,

B01000000, //1

B11000000,

B01000000,

B01000000,

B01000000,

B01000000,

B11100000,

4,

B01110000, //2

B10001000,

B00001000,

B00010000,

B00100000,

B01000000,

B11111000,

6,

B11111000, //3

B00010000,

B00100000,

B00010000,

B00001000,

B10001000,

B01110000,

6,

B00010000, //4

B00110000,

B01010000,

B10010000,

B11111000,

B00010000,

B00010000,

6,

B11111000, //5

B10000000,

B11110000,

B00001000,

B00001000,

B10001000,

B01110000,

6,

B00110000, //6

B01000000,

B10000000,

B11110000,

B10001000,

B10001000,

B01110000,

6,

B11111000, //7

B10001000,

B00001000,

B00010000,

B00100000,

B00100000,

B00100000,

6,

B01110000, //8

B10001000,

B10001000,

B01110000,

B10001000,

B10001000,

B01110000,

6,

B01110000, //9

B10001000,

B10001000,

B01111000,

B00001000,

B00010000,

B01100000,

6,

B00000000, //:

B11000000,

B11000000,

B00000000,

B11000000,

B11000000,

B00000000,

3,

B00000000, //;

B11000000,

B11000000,

B00000000,

B11000000,

B01000000,

B10000000,

3,

B00010000, //<

B00100000,

B01000000,

B10000000,

B01000000,

B00100000,

B00010000,

5,

B00000000, //=

B00000000,

B11111000,

B00000000,

B11111000,

B00000000,

B00000000,

6,

B10000000, //>

B01000000,

B00100000,

B00010000,

B00100000,

B01000000,

B10000000,

5,

B01110000, //?

B10001000,

B00001000,

B00010000,

B00100000,

B00000000,

B00100000,

6,

B01110000, //@

B10001000,

B00001000,

B01101000,

B10101000,

B10101000,

B01110000,

6,

B01110000, //A

B10001000,

B10001000,

B10001000,

B11111000,

B10001000,

B10001000,

6,

B11110000, //B

B10001000,

B10001000,

B11110000,

B10001000,

B10001000,

B11110000,

6,

B01110000, //C

B10001000,

B10000000,

B10000000,

B10000000,

B10001000,

B01110000,

6,

B11100000, //D

B10010000,

B10001000,

B10001000,

B10001000,

B10010000,

B11100000,

6,

B11111000, //E

B10000000,

B10000000,

B11110000,

B10000000,

B10000000,

B11111000,

6,

B11111000, //F

B10000000,

B10000000,

B11110000,

B10000000,

B10000000,

B10000000,

6,

B01110000, //G

B10001000,

B10000000,

B10111000,

B10001000,

B10001000,

B01111000,

6,

B10001000, //H

B10001000,

B10001000,

B11111000,

B10001000,

B10001000,

B10001000,

6,

B11100000, //I

B01000000,

B01000000,

B01000000,

B01000000,

B01000000,

B11100000,

4,

B00111000, //J

B00010000,

B00010000,

B00010000,

B00010000,

B10010000,

B01100000,

6,

B10001000, //K

B10010000,

B10100000,

B11000000,

B10100000,

B10010000,

B10001000,

6,

B10000000, //L

B10000000,

B10000000,

B10000000,

B10000000,

B10000000,

B11111000,

6,

B10001000, //M

B11011000,

B10101000,

B10101000,

B10001000,

B10001000,

B10001000,

6,

B10001000, //N

B10001000,

B11001000,

B10101000,

B10011000,

B10001000,

B10001000,

6,

B01110000, //O

B10001000,

B10001000,

B10001000,

B10001000,

B10001000,

B01110000,

6,

B11110000, //P

B10001000,

B10001000,

B11110000,

B10000000,

B10000000,

B10000000,

6,

B01110000, //Q

B10001000,

B10001000,

B10001000,

B10101000,

B10010000,

B01101000,

6,

B11110000, //R

B10001000,

B10001000,

B11110000,

B10100000,

B10010000,

B10001000,

6,

B01111000, //S

B10000000,

B10000000,

B01110000,

B00001000,

B00001000,

B11110000,

6,

B11111000, //T

B00100000,

B00100000,

B00100000,

B00100000,

B00100000,

B00100000,

6,

B10001000, //U

B10001000,

B10001000,

B10001000,

B10001000,

B10001000,

B01110000,

6,

B10001000, //V

B10001000,

B10001000,

B10001000,

B10001000,

B01010000,

B00100000,

6,

B10001000, //W

B10001000,

B10001000,

B10101000,

B10101000,

B10101000,

B01010000,

6,

B10001000, //X

B10001000,

B01010000,

B00100000,

B01010000,

B10001000,

B10001000,

6,

B10001000, //Y

B10001000,

B10001000,

B01010000,

B00100000,

B00100000,

B00100000,

6,

B11111000, //Z

B00001000,

B00010000,

B00100000,

B01000000,

B10000000,

B11111000,

6,

B11100000, //[

B10000000,

B10000000,

B10000000,

B10000000,

B10000000,

B11100000,

4,

B00000000, //(Backward Slash)

B10000000,

B01000000,

B00100000,

B00010000,

B00001000,

B00000000,

6,

B11100000, //]

B00100000,

B00100000,

B00100000,

B00100000,

B00100000,

B11100000,

4,

B00100000, //^

B01010000,

B10001000,

B00000000,

B00000000,

B00000000,

B00000000,

6,

B00000000, //\_

B00000000,

B00000000,

B00000000,

B00000000,

B00000000,

B11111000,

6,

B10000000, //`

B01000000,

B00100000,

B00000000,

B00000000,

B00000000,

B00000000,

4,

B00000000, //a

B00000000,

B01110000,

B00001000,

B01111000,

B10001000,

B01111000,

6,

B10000000, //b

B10000000,

B10110000,

B11001000,

B10001000,

B10001000,

B11110000,

6,

B00000000, //c

B00000000,

B01110000,

B10001000,

B10000000,

B10001000,

B01110000,

6,

B00001000, //d

B00001000,

B01101000,

B10011000,

B10001000,

B10001000,

B01111000,

6,

B00000000, //e

B00000000,

B01110000,

B10001000,

B11111000,

B10000000,

B01110000,

6,

B00110000, //f

B01001000,

B01000000,

B11100000,

B01000000,

B01000000,

B01000000,

6,

B00000000, //g

B01111000,

B10001000,

B10001000,

B01111000,

B00001000,

B01110000,

6,

B10000000, //h

B10000000,

B10110000,

B11001000,

B10001000,

B10001000,

B10001000,

6,

B01000000, //i

B00000000,

B11000000,

B01000000,

B01000000,

B01000000,

B11100000,

4,

B00010000, //j

B00000000,

B00110000,

B00010000,

B00010000,

B10010000,

B01100000,

5,

B10000000, //k

B10000000,

B10010000,

B10100000,

B11000000,

B10100000,

B10010000,

5,

B11000000, //l

B01000000,

B01000000,

B01000000,

B01000000,

B01000000,

B11100000,

4,

B00000000, //m

B00000000,

B11010000,

B10101000,

B10101000,

B10001000,

B10001000,

6,

B00000000, //n

B00000000,

B10110000,

B11001000,

B10001000,

B10001000,

B10001000,

6,

B00000000, //o

B00000000,

B01110000,

B10001000,

B10001000,

B10001000,

B01110000,

6,

B00000000, //p

B00000000,

B11110000,

B10001000,

B11110000,

B10000000,

B10000000,

6,

B00000000, //q

B00000000,

B01101000,

B10011000,

B01111000,

B00001000,

B00001000,

6,

B00000000, //r

B00000000,

B10110000,

B11001000,

B10000000,

B10000000,

B10000000,

6,

B00000000, //s

B00000000,

B01110000,

B10000000,

B01110000,

B00001000,

B11110000,

6,

B01000000, //t

B01000000,

B11100000,

B01000000,

B01000000,

B01001000,

B00110000,

6,

B00000000, //u

B00000000,

B10001000,

B10001000,

B10001000,

B10011000,

B01101000,

6,

B00000000, //v

B00000000,

B10001000,

B10001000,

B10001000,

B01010000,

B00100000,

6,

B00000000, //w

B00000000,

B10001000,

B10101000,

B10101000,

B10101000,

B01010000,

6,

B00000000, //x

B00000000,

B10001000,

B01010000,

B00100000,

B01010000,

B10001000,

6,

B00000000, //y

B00000000,

B10001000,

B10001000,

B01111000,

B00001000,

B01110000,

6,

B00000000, //z

B00000000,

B11111000,

B00010000,

B00100000,

B01000000,

B11111000,

6,

B00100000, //{

B01000000,

B01000000,

B10000000,

B01000000,

B01000000,

B00100000,

4,

B10000000, //|

B10000000,

B10000000,

B10000000,

B10000000,

B10000000,

B10000000,

2,

B10000000, //}

B01000000,

B01000000,

B00100000,

B01000000,

B01000000,

B10000000,

4,

B00000000, //~

B00000000,

B00000000,

B01101000,

B10010000,

B00000000,

B00000000,

6,

B01100000, // (Char 0x7F)

B10010000,

B10010000,

B01100000,

B00000000,

B00000000,

B00000000,

5,

B00000000, // smiley

B01100000,

B01100110,

B00000000,

B10000001,

B01100110,

B00011000,

5

};

//This are the functions that send the serial data of each

//vectors to the MAX7219 drivers

**void** scrollFont() {

for (**int** counter=0x20;counter<0x80;counter++){

loadBufferLong(counter);

delay(500);

}

}

// Scroll Message

**void** scrollMessage(**const** **unsigned** **char** \* messageString) {

**int** counter = 0;

**int** myChar=0;

do {

// read back a char

myChar = pgm\_read\_byte\_near(messageString + counter);

if (myChar != 0){

loadBufferLong(myChar);

}

counter++;

}

while (myChar != 0);

}

// Load character into scroll buffer

**void** loadBufferLong(**int** ascii){

if (ascii >= 0x20 && ascii <=0x7f){

for (**int** a=0;a<7;a++){ // Loop 7 times for a 5x7 font

**unsigned** **long** c = pgm\_read\_byte\_near(font5x7 + ((ascii - 0x20) \* 8) + a); // Index into character table to get row data

**unsigned** **long** x = bufferLong [a\*2]; // Load current scroll buffer

x = x | c; // OR the new character onto end of current

bufferLong [a\*2] = x; // Store in buffer

}

**byte** count = pgm\_read\_byte\_near(font5x7 +((ascii - 0x20) \* 8) + 7); // Index into character table for kerning data

for (**byte** x=0; x<count;x++){

rotateBufferLong();

printBufferLong();

delay(scrollDelay);

}

}

}

// Rotate the buffer

**void** rotateBufferLong(){

for (**int** a=0;a<7;a++){ // Loop 7 times for a 5x7 font

**unsigned** **long** x = bufferLong [a\*2]; // Get low buffer entry

**byte** b = bitRead(x,31); // Copy high order bit that gets lost in rotation

x = x<<1; // Rotate left one bit

bufferLong [a\*2] = x; // Store new low buffer

x = bufferLong [a\*2+1]; // Get high buffer entry

x = x<<1; // Rotate left one bit

bitWrite(x,0,b); // Store saved bit

bufferLong [a\*2+1] = x; // Store new high buffer

}

}

// Display Buffer on LED matrix

**void** printBufferLong(){

for (**int** a=0;a<7;a++){ // Loop 7 times for a 5x7 font

**unsigned** **long** x = bufferLong [a\*2+1]; // Get high buffer entry

**byte** y = x; // Mask off first character

lc.setRow(3,a,y); // Send row to relevent MAX7219 chip

x = bufferLong [a\*2]; // Get low buffer entry

y = (x>>24); // Mask off second character

lc.setRow(2,a,y); // Send row to relevent MAX7219 chip

y = (x>>16); // Mask off third character

lc.setRow(1,a,y); // Send row to relevent MAX7219 chip

y = (x>>8); // Mask off forth character

lc.setRow(0,a,y); // Send row to relevent MAX7219 chip

}

}