EECS 129B Winter 2008 Final Project Report

Due March 17, 2008 on EEE

Team Members

Last Name	First Name
Park	Taejoon
Pham	Danh
Shu	Zhan

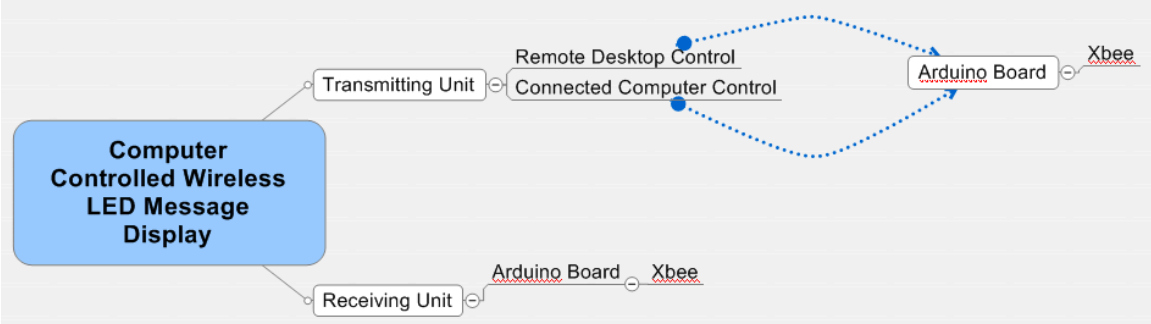
Computer Controlled Wireless LED Message Display

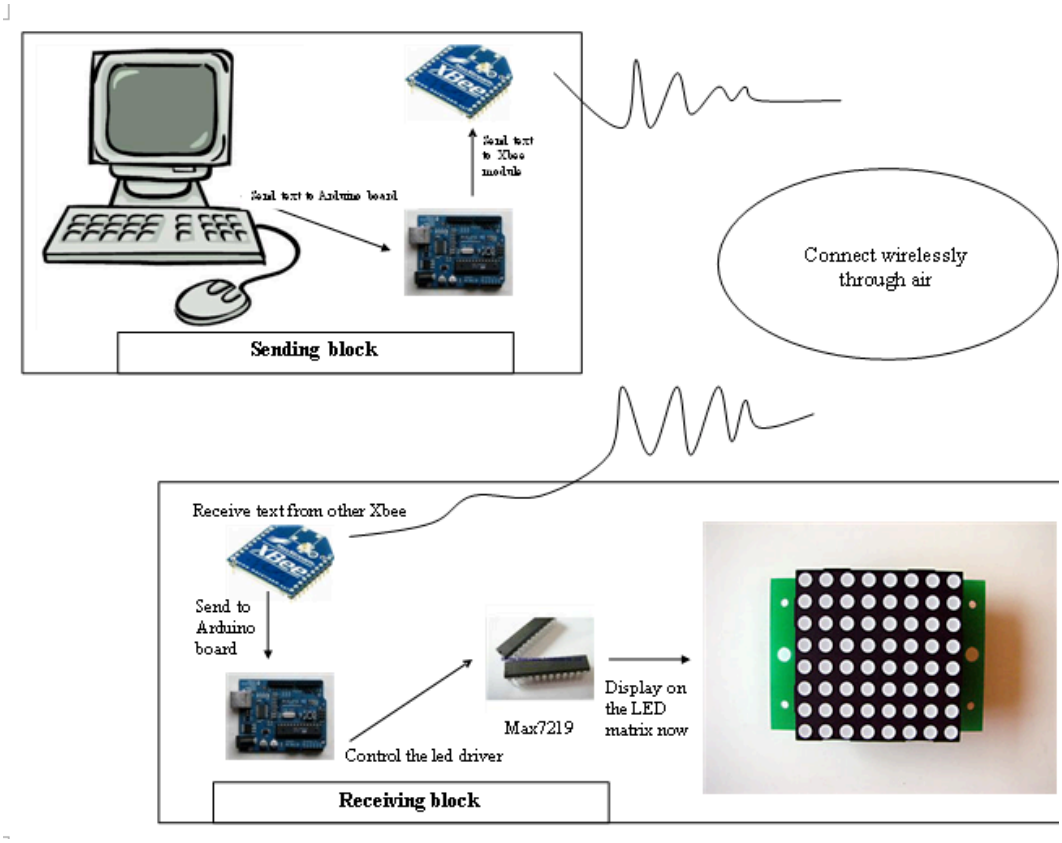
1. Abstract

Wireless communication between 2 Xbees which are connected individually to one Arduino board is set to be the goal. There will be two devices for this project. One, called the message displayer, is going to receive messages wirelessly and display them using matrix LED panels. The other device, called message transmitter, is going to deliver messages to be displayed wirelessly from a computer. The message transmitter will be connected to a computer so that users can input any message they want to display over the LED display panels on the message displayer.

One application for this product can be to guide customers in line to the available windows for serving.

2. Project overview (High level)





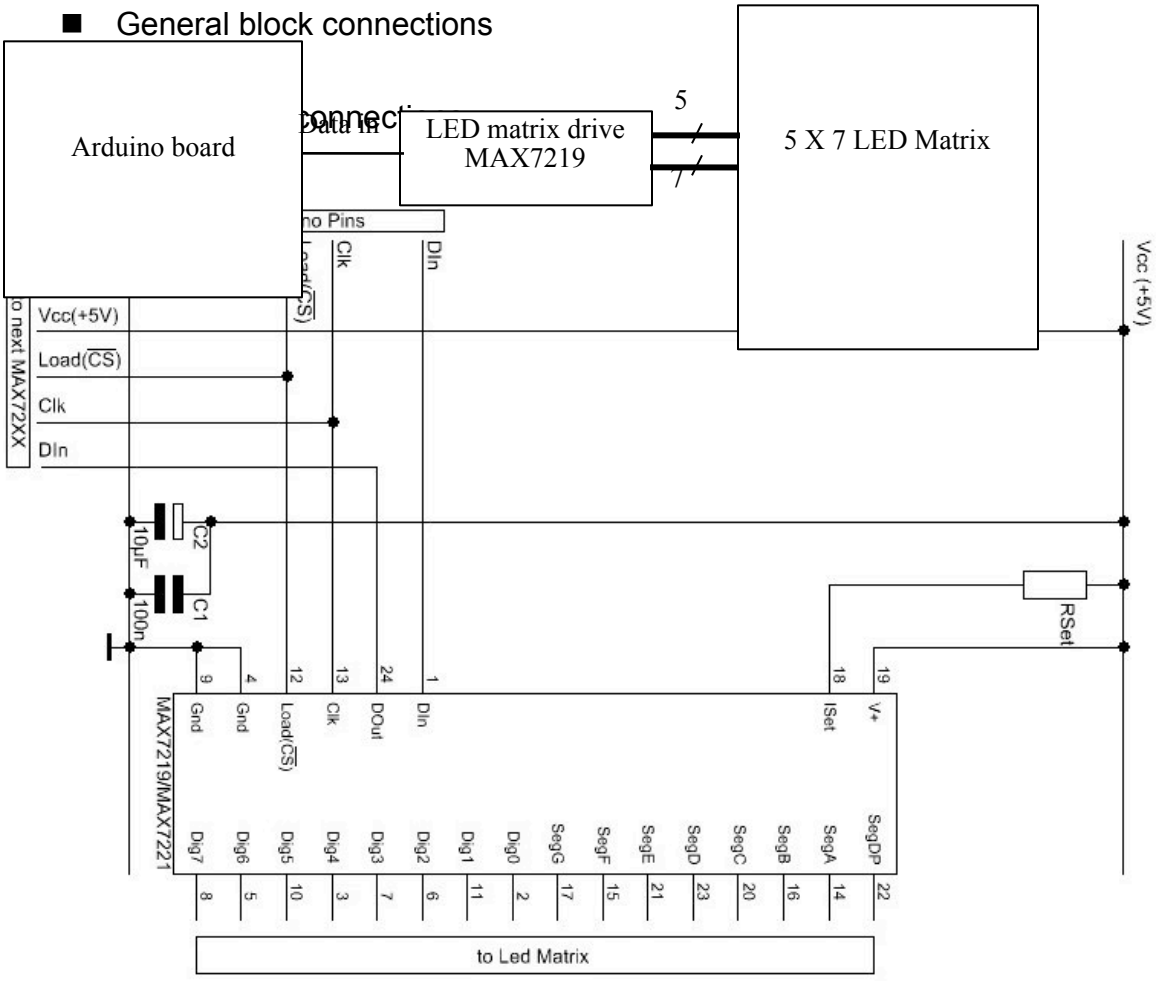
3. Project components

Mechanical engineering

This project does not involve mechanical engineering so far.

Electrical

- Connection between MAX7219 and LED dot matrix
  - General block connections



Software

- Using programming language C in Arduino board to send and receive messages through Xbees wirelessly.
- The receiving and sending unit codes are attached at the end of the report
- Receiving unit pseudo code

Setup fuction:

1. Assign arduino board pins: input and output for Xbee, single LED output
2. Start "Serial" mode
3. Assign arduino board pins: data in, clock, and load for LED matrix driver
4. Initialize functions of MAX7219

Loop function:

5. Initialize all LED matrices
6. Receive messages from Xbee
   a) Save messages into array
   b) Each time take front byte to last matrix, move other bytes to the left by one matrix
   c) When finish reading from the array, keep rotating the message until receive new message
      i. If message's size is less than or equal to 6, only push the message into the matrices, and flash it afterwards
      ii. If message's size is larger than 6, keep rotating.

Other functions used in previous functions:

7. Select and display letter/number/symbol: When a symbol can be recognized, then display it.
8. Display pattern for each letter/number/symbol
9. returnOK function: this function checks the response on the serial port.

---

4. Design description

1. The message displayer will consist of an Arduino board, Xbee, some matrix LED pannels, and any circuitry needed to connect each component.

2. The message deliver will also consist of an Arduino board, Xbee, some matrix LED pannels, and any circuitry needed to connect each component.

5. System test plan

Number	Description of Set-up	Input or Stimulus	Expected Behavior
1	Wire 1 LED matrix with Aduino board, use function for only one matrix, upload function to board	A fixed function of a display pattern	Display the pattern
2	Wire 1 LED matrix with Aduino board, use printLetter function	Letter "A", or any other letter	Display capital letter
3	Cascade 2 LED matrices, use "maxAll" function	A display pattern	2 matrices display the same pattern
4	Cascade 2 LED matrices, use "maxAll" function	Letter "A", or any other letter	2 matrices display the same letter
5	Cascade 2 LED matrices, use for loop to change the column number	Light up one column, and keep decreasing the column number	One column rolling from one side to the other
6	Cascade 2 LED matrices, use "maxOne" function	Send "A" and "B"	Display "A" on matrix 1, "B" on matrix 2
7	Cascade 2 LED matrices, use "maxOne" function and for loop to change the number of matrix	Send "A" and "B"	Display "A" on matrix 1, "B" on matrix 2; then switch them
8	Connect Xbee to arduino board, use "serial monitor" to monitor the sending message	Send "A" from arduino program to board	LED blink once indicate sending message
9	Connect LED matrices to arduino board2	Receive message and display character	LED blink twice, and matrices display same letter
10	Cascade 6 LED matrices to arduino board2	Hard code input in the function	Display message as in the code

Number	Description of Set-up	Input or Stimulus	Expected Behavior
11	Cascade 6 matrices, add rotating display function	Hard code input in the function	Display message as in the code, keep rotating
12	Cascade 6 matrices, send a more than 6 letter sentence using the laptop connected to the sending units	Messages from the keyboard connected to the laptop and sending unit	Display message as it was sent, the message should be rotating
13	Cascade 6 matrices, send less than 7 letter sentence	Messages from the keyboard connected to the laptop and sending unit	Display message as it was sent, the message should be flashing
14	Cascade 6 matrices, send every character, number, and symbol the receiving unit support.	every character, number, and symbol	Display all the characters, numbers, and symbols
15	Let it display for more than 10 minute to see whether it is stable after it displays the message	Any message less than 128 bytes	Display messages as long as we send next message
16	Send more than 128 bytes messages	Any message more than 128 bytes (letters)	Display no message

6. Project timeline

Date	Description
1/16/2008	choosing project
1/18/2008	ordering parts
1/22/2008	figuring out how LED drivers and LED matrix display work
2/3/2008	displaying a letter on the LED matrix board
2/11/2008	displaying a sentence on the LED matrix board
2/18/2008	sending, receiving, and displaying a sentence on the LED matrix board

7. Division of work

Zhan Shu

Connecting the LED matrix and LED matrix drivers circuit and program the displaying of text messages on the matrix.

Taejoon Park

Connecting the receiving block circuit and program the receiving of text messages sent from the sending block.

Danh Pham

Connecting the sending block circuit and program the sending of text messages to the receiving block.

8. Cost

Quantity	Supplier	Part Number	Description	Price ($)	Total ($)
2	www.sparkfun.com	DEV-00666	Arduino USB Board	31.46	62.92
2	www.digikey.com	XB24-AWI-001-ND	module ZIGBEE 1MW W/WIRE ANT	19.00	38.00
6	www.jameco.com	206560PS	LED dot matrix display	1.85	11.10
6	www.jameco.com	312160PS	LED display driver	8.95	53.70
2	Fry's Electronics	TW-E41-1060	breadboard	27.99	55.98
100	www.jameco.com	691260	47k resistor	0.01	1.00
2	www.jameco.com	229711	USB, A to B cable	1.55	3.10
1	www.jameco.com	36792	100' reel 22awg, solid black wire	5.49	5.49
1	www.jameco.com	36856	100' reel 22awg, solid red wire	5.49	5.49
1	www.jameco.com	36920	100' reel 22awg, solid yellow wire	5.49	5.49
1	www.jameco.com	226085	electronic tool kit	17.95	17.95
10	www.jameco.com	211086	green led	0.05	0.50
10	www.jameco.com	202471	red led	0.06	0.60
				Total	261.32

Total cost of project: $ 261.32

9. Problem encountered and comments

Sending part:

At first, we could not get the text message sent to the board. The reason was the Arduino board shares its serial input port among the USB and RX connection. Only one of these connections can be active at a time. In order to initialize the Xbee module, we have to send to it message and wait for it to reply; that means the Xbee module will reply a message to the serial RX input of the board. Therefore, USB serial input is blocked and no data can be sent to the board through USB. We solved the problem by inserting a switch. This switch only makes one type of serial connections active. So we will initialize the Xbee module first, then deactivate it and make USB connection available.

Wiring part:

After purchasing the common cathold LED matrix, we just found out that the LED matrix has common cathold "columns" but not same as the 8X8 LED matrix sample in the Arduino website. We had to redo the configuration of the columns and rows so that we can use the functions associated with the LED matrix drivers.

Receiving part:

When there was not enough delay between receiving from Xbee to Xbee, the Serial.read function could not get all the bytes sent from the sending unit. It was crucial to find right amount of delay for the receiving unit. If the delay is too small, it will display only the messages it gets during the delay and cut the rest of delay by ignoring or displaying them as next messages. If the delay is too large, the receiving unit gets to waste of time after it received full message. The other issue is that it needs more power than it can get from USB cable. When we used USB cable as a power source for the receiving unit, the receiving unit does not display the last LED matrix after 2 or 3 minutes. We had to use 110V power outlet for the receiving unit so it can have enough power to drive all the LED matrixes.

//Code for receiving uint

/* code for max 7219 from maxim,
reduced and optimised for useing more then one 7219 in a row,

_____

Code History:

The orginal code was written for the Wiring board by:
* Nicholas Zambetti and Dave Mellis /Interaction Design Institute Ivrea /Dec 2004
* http://www.potemkin.org/uploads/Wiring/MAX7219.txt

First modification by:
* Marcus Hannerstig/ K3, malm?h?skola /2006
* http://www.xlab.se | http://arduino.berlios.de

This version is by:
* tomek ness /FH-Potsdam / Feb 2007
* http://design.fh-potsdam.de/

* @acknowledgements: eric f.

General notes:
-if you are only using one max7219, then use the function maxSingle to control the little guy ---maxSingle(register (1-8), collum (0-255))

-if you are using more then one max7219, and they all should work the same, then use the function maxAll ---maxAll(register (1-8), collum (0-255))

-if you are using more than one max7219 and just want to change something at one little guy, then use the function maxOne
---maxOne(Max you wane controll (1== the first one), register (1-8), collum (0-255))

```
/* During initiation, be sure to send every part to every max7219 and then
 upload it.
For example, if you have five max7219's, you have to send the scanLimit 5 times
before you load it-- other wise not every max7219 will get the data. the
function maxInUse  keeps track of this, just tell it how many max7219 you are
using.
*/

int dataIn = 2;
int load = 3;
int clock = 4;


int maxInUse = 6;    //change this variable to set how many MAX7219's you'll use

int e = 0;           // just a varialble

                  // define max7219 registers
byte max7219_reg_noop        = 0x00;
byte max7219_reg_digit0      = 0x01;
byte max7219_reg_digit1      = 0x02;
byte max7219_reg_digit2      = 0x03;
byte max7219_reg_digit3      = 0x04;
byte max7219_reg_digit4      = 0x05;
byte max7219_reg_digit5      = 0x06;
byte max7219_reg_digit6      = 0x08;
byte max7219_reg_digit7      = 0x01;
byte max7219_reg_decodeMode  = 0x09;
byte max7219_reg_intensity   = 0x0a;
byte max7219_reg_scanLimit   = 0x0b;
byte max7219_reg_shutdown    = 0x0c;
byte max7219_reg_displayTest = 0x0f;

void putByte(byte data) {
  byte i = 8;
  byte mask;
  while(i > 0) {
```

```
    mask = 0x01 << (i - 1);      // get bitmask
    digitalWrite( clock, LOW);   // tick
    if (data & mask){            // choose bit
      digitalWrite(dataIn, HIGH);// send 1
    }else{
      digitalWrite(dataIn, LOW); // send 0
    }
    digitalWrite(clock, HIGH);   // tock
    --i;                         // move to lesser bit
  }
}

void maxSingle( byte reg, byte col) {
//maxSingle is the "easy"  function to use for a     //single max7219

  digitalWrite(load, LOW);       // begin
  putByte(reg);                  // specify register
  putByte(col);//((data & 0x01) * 256) + data >> 1); // put data
  digitalWrite(load, LOW);       // and load da shit
  digitalWrite(load,HIGH);
}

void maxAll (byte reg, byte col) {    // initialize  all  MAX7219's in the system
  int c = 0;
  digitalWrite(load, LOW);  // begin
  for ( c =1; c<= maxInUse; c++) {
  putByte(reg);  // specify register
  putByte(col);//((data & 0x01) * 256) + data >> 1); // put data
    }
  digitalWrite(load, LOW);
  digitalWrite(load,HIGH);
}

void maxOne(byte maxNr, byte reg, byte col) {
//maxOne is for adressing different MAX7219's,
//whilele having a couple of them cascaded
```

```
  int c = 0;
  digitalWrite(load, LOW);  // begin

  for ( c = maxInUse; c > maxNr; c--) {
    putByte(0);    // means no operation
    putByte(0);    // means no operation
  }

  putByte(reg);  // specify register
  putByte(col);//((data & 0x01) * 256) + data >> 1); // put data

  for ( c =maxNr-1; c >= 1; c--) {
    putByte(0);    // means no operation
    putByte(0);    // means no operation
  }

  digitalWrite(load, LOW); // and load da shit
  digitalWrite(load,HIGH);
}



//these are the functions needed for LED matrix
// you can add addtional sysmbols later if you want
/*
void printPlusSymbol(byte maxNo);       // +
void printMinusSymbol(byte maxNo);      // -
void printTimesSymbol(byte maxNo);      // *
void printDivisionSymbol(byte maxNo);   // /
void printEqualSymbol(byte maxNo);      // =
void printOpenPrenSymbol(byte maxNo);   // (
void printClosePrenSymbol(byte maxNo);  // )
*/

void printCommaSymbol(byte maxNo);       // ,
void printQuestionSymbol(byte maxNo);    // ?
void printExclamationSymbol(byte maxNo); // !
```

```
void printPeriodSymbol(byte maxNo);      // .

void printDollarSymbol(byte maxNo);    // $
void printNumSymbol(byte maxNo);       // #
void printAndSymbol(byte maxNo);       // &



void print0(byte maxNo);
void print1(byte maxNo);
void print2(byte maxNo);
void print3(byte maxNo);
void print4(byte maxNo);
void print5(byte maxNo);
void print6(byte maxNo);
void print7(byte maxNo);
void print8(byte maxNo);
void print9(byte maxNo);


void clearM(byte maxNo );
void printA(byte maxNo );
void printB(byte maxNo );
void printC(byte maxNo );
void printD(byte maxNo );
void printE(byte maxNo );
void printF(byte maxNo );
void printG(byte maxNo );
void printH(byte maxNo );
void printI(byte maxNo );
void printJ(byte maxNo );
void printK(byte maxNo );
void printL(byte maxNo );
void printM(byte maxNo );
void printN(byte maxNo );
void printO(byte maxNo );
void printP(byte maxNo );
```

```
void printQ(byte maxNo );
void printR(byte maxNo );
void printS(byte maxNo );
void printT(byte maxNo );
void printU(byte maxNo );
void printV(byte maxNo );
void printW(byte maxNo );
void printX(byte maxNo );
void printY(byte maxNo );
void printZ(byte maxNo );
void printSpace(byte maxNo );
void selectLetter(byte let,byte m);
void printError();

//functions for XBEE
char returnOK();

//variables for XBEE
int rx = 0;
int tx = 1;
int led = 13;

void blinkLED(int nTimes, int miliSecond)
{
  for (int i=0; i<nTimes; i++)
  {
    digitalWrite(led, HIGH);
    delay(miliSecond);
    digitalWrite(led, LOW);
    delay(miliSecond);
  }
}

void setup ()
{
 //this is for XBEE
 pinMode(rx, INPUT);
```

```
pinMode(tx, OUTPUT);
//pinMode(speakerOut, OUTPUT);
pinMode(led, OUTPUT);
//pinMode(switchPin, INPUT);

Serial.begin(9600);

delay(2000);
Serial.print("+++");
delay(2000);

if (returnOK() == 'T')
{
   blinkLED(2, 500);
}
else
{
  blinkLED(2,2000);
  setup();
}

//to distinguish from the first setup part
delay(2000);

Serial.print("ATID1234, CN");
if (returnOK() == 'T')
{
   blinkLED(3,500);
}
else
{
 blinkLED(3,2000);
  setup();
}


//jackie's part for LED matrix
```

```
  pinMode(dataIn, OUTPUT);
  pinMode(clock,  OUTPUT);
  pinMode(load,   OUTPUT);

  beginSerial(9600);
  blinkLED(1,1000);

//initiation of the max 7219
  maxAll(max7219_reg_scanLimit, 0x07);
  maxAll(max7219_reg_decodeMode, 0x00);  // using an led matrix (not digits)
  maxAll(max7219_reg_shutdown, 0x01);    // not in shutdown mode
  maxAll(max7219_reg_displayTest, 0x00); // no display test
   for (e=1; e<=8; e++) {    // empty registers, turn all LEDs off
    maxAll(e,0);
  }
  maxAll(max7219_reg_intensity, 0x0f & 0x0f);     // the first 0x0f is the value you
can set
                                        // range: 0x00 to 0x0f

//clear Arduino buffer to erase old stuff
    Serial.flush();
}

//allocate a byte array to store received bytes
byte st[128]={'0'};


int p,size,j,l;
void loop ()
{

    //initialize all matrices to "Space"
    byte LedMatrix[maxInUse+1];
    for(int k=0;k<=maxInUse+1;k++)
    {
         LedMatrix[k]= ' ';
         selectLetter(LedMatrix[k],(byte)k);
```

```
}
  st[1]='a';
 st[2]='m';
 st[3]='t';
 st[4]='1';
 st[5]='6';
 st[6]='8';
 size=6;


if (Serial.available() > 0)
{
    blinkLED(1,50);
    delay(260);      //wait for the whole message to come

    //put received bytes into array
    size = Serial.available();

    if(size > 6)  //7 or more
    {
    for(p=1;p<=size+maxInUse-1;p++)
    {
         st[p]= Serial.read();
         if(p>size)
              st[p]=' ';

           //put a sentence in the array
        for( j=1; j<maxInUse; j++)
          LedMatrix[j] = LedMatrix[j+1];
         LedMatrix[maxInUse] = st[p]; //take one letter from array, and put into the
last matrix

           //print
           for( l=1; l<=maxInUse; l++)
             selectLetter(LedMatrix[l], l);

           delay(500);    //how fast each letter moves
```

```
            //selectLetter(st[p],maxInUse); delay(1000);
        }
            Serial.flush();

            //keep rolling until you get another message
            while(Serial.available()==0)
            {
                for(p=1;p<=size+maxInUse-1;p++)
                {
                    //put a sentence in the LED array
                for( j=1; j<maxInUse; j++)
                  LedMatrix[j] = LedMatrix[j+1];
                    LedMatrix[maxInUse] = st[p]; //take one letter from the string array,
and put into the last matrix display array

                    //print the message to the LED displayer
                    for(l=1; l<=maxInUse; l++)
                        selectLetter(LedMatrix[l], l);

                    //delay before the message move to the next LED display
                    delay(500);
                    //selectLetter(st[p],maxInUse); delay(1000);
                }
            }

        }
        else //if message size is 1 to 6, then i will make it flash
        {
          for(p=1;p<=size;p++)
        {
             st[p]= Serial.read();
             if(p>size)
                  st[p]=' ';

              //put a sentence in the array
           for( j=1; j<maxInUse; j++)
             LedMatrix[j] = LedMatrix[j+1];
```

```
            LedMatrix[maxInUse] = st[p]; //take one letter from array, and put into the
last matrix

              //print
              for( l=1; l<=maxInUse; l++)
                selectLetter(LedMatrix[l], l);

              delay(500);    //how fast each letter moves
              //selectLetter(st[p],maxInUse); delay(1000);
          }

          //keep rolling until you get another message
          while(Serial.available()==0)
          {
              //print it
              for(l=1; l<=maxInUse; l++)
                selectLetter(LedMatrix[l], l);

              //delay before the message move to the next LED display
              delay(500);

              for(l=1; l<=maxInUse; l++)
                selectLetter(' ', l);

              delay(500);
          }
        } // the else for the sentence size <= 6 ends here

    } //the first if(serial.size >0) ends here


}       //loop() ends here



//function definitions
```

```
void selectLetter(byte let,byte m)    //m is the index of max
{

  switch (let)
    {
      case ',': printCommaSymbol(m);break;
      case '?':printQuestionSymbol(m);break;
      case '!':printExclamationSymbol(m);break;
      case '.':printPeriodSymbol(m);break;

      case '$':printDollarSymbol(m);break;
      case '#':printNumSymbol(m);break;
      case '&':printAndSymbol(m);break;

      case '0':print0(m);break;
      case '1':print1(m);break;
      case '2':print2(m);break;
      case '3':print3(m);break;
      case '4':print4(m);break;
      case '5':print5(m);break;
      case '6':print6(m);break;
      case '7':print7(m);break;
      case '8':print8(m);break;
      case '9':print9(m);break;

      case 'A': printA(m);break;
      case 'B': printB(m);break;
      case 'C': printC(m);break;
      case 'D': printD(m);break;
      case 'E': printE(m);break;
      case 'F': printF(m);break;
      case 'G': printG(m);break;
      case 'H': printH(m);break;
      case 'I': printI(m);break;
      case 'J': printJ(m);break;
      case 'K': printK(m);break;
      case 'L': printL(m);break;
```

```
case 'M': printM(m);break;
case 'N': printN(m);break;
case 'O': printO(m);break;
case 'P': printP(m);break;
case 'Q': printQ(m);break;
case 'R': printR(m);break;
case 'S': printS(m);break;
case 'T': printT(m);break;
case 'U': printU(m);break;
case 'V': printV(m);break;
case 'W': printW(m);break;
case 'X': printX(m);break;
case 'Y': printY(m);break;
case 'Z': printZ(m);break;

case 'a': printA(m);break;
case 'b': printB(m);break;
case 'c': printC(m);break;
case 'd': printD(m);break;
case 'e': printE(m);break;
case 'f': printF(m);break;
case 'g': printG(m);break;
case 'h': printH(m);break;
case 'i': printI(m);break;
case 'j': printJ(m);break;
case 'k': printK(m);break;
case 'l': printL(m);break;
case 'm': printM(m);break;
case 'n': printN(m);break;
case 'o': printO(m);break;
case 'p': printP(m);break;
case 'q': printQ(m);break;
case 'r': printR(m);break;
case 's': printS(m);break;
case 't': printT(m);break;
case 'u': printU(m);break;
case 'v': printV(m);break;
```

```
      case 'w': printW(m);break;
      case 'x': printX(m);break;
      case 'y': printY(m);break;
      case 'z': printZ(m);break;

      case ' ': printSpace(m);break;
      default: printSpace(m);break;
    }
}


void clearM(byte maxNo)
{
   for(int i=1;i<=5;i++)
   maxOne(maxNo,i,0);
}

void clearAll()
{
  for(int i=1;i<=5;i++)
  maxAll(i,0);
}


//////////fucntions for printing symbols

/*
void printPlusSymbol(byte maxNo);       // +
void printMinusSymbol(byte maxNo);      // -
void printTimesSymbol(byte maxNo);      // *
void printDivisionSymbol(byte maxNo);   // /
void printEqualSymbol(byte maxNo);      // =
void printOpenPrenSymbol(byte maxNo);   // (
void printClosePrenSymbol(byte maxNo);  // )
*/

void printCommaSymbol(byte maxNo)      // ,
```

```
{
   maxOne(maxNo,1,13);
   maxOne(maxNo,2,14);
   maxOne(maxNo,3,0);
   maxOne(maxNo,4,0);
   maxOne(maxNo,5,0);
}
void printQuestionSymbol(byte maxNo)     // ?
{
   maxOne(maxNo,1,32);
   maxOne(maxNo,2,64);
   maxOne(maxNo,3,69);
   maxOne(maxNo,4,72);
   maxOne(maxNo,5,48);
}
void printExclamationSymbol(byte maxNo)  // !
{
   maxOne(maxNo,1,0);
   maxOne(maxNo,2,0);
   maxOne(maxNo,3,125);
   maxOne(maxNo,4,0);
   maxOne(maxNo,5,0);
}
void printPeriodSymbol(byte maxNo)       // .
{
   maxOne(maxNo,1,3);
   maxOne(maxNo,2,3);
   maxOne(maxNo,3,0);
   maxOne(maxNo,4,0);
   maxOne(maxNo,5,0);
}

void printDollarSymbol(byte maxNo)    // $
{
   maxOne(maxNo,1,50);
   maxOne(maxNo,2,73);
   maxOne(maxNo,3,127);
```

```
    maxOne(maxNo,4,73);
    maxOne(maxNo,5,38);
}
void printNumSymbol(byte maxNo)       // #
{
    maxOne(maxNo,1,20);
    maxOne(maxNo,2,127);
    maxOne(maxNo,3,20);
    maxOne(maxNo,4,127);
    maxOne(maxNo,5,20);
}
void printAndSymbol(byte maxNo)       // &
{
    maxOne(maxNo,1,34);
    maxOne(maxNo,2,85);
    maxOne(maxNo,3,77);
    maxOne(maxNo,4,82);
    maxOne(maxNo,5,37);
}

//////////functions for printing numbers
void print0(byte maxNo)
{
    maxOne(maxNo,1,62);
    maxOne(maxNo,2,69);
    maxOne(maxNo,3,73);
    maxOne(maxNo,4,81);
    maxOne(maxNo,5,62);
}
void print1(byte maxNo)
{
    maxOne(maxNo,1,0);
    maxOne(maxNo,2,33);
    maxOne(maxNo,3,127);
    maxOne(maxNo,4,1);
    maxOne(maxNo,5,0);
}
```

```
void print2(byte maxNo)
{
   maxOne(maxNo,1,33);
   maxOne(maxNo,2,67);
   maxOne(maxNo,3,69);
   maxOne(maxNo,4,73);
   maxOne(maxNo,5,49);
}
void print3(byte maxNo)
{
   maxOne(maxNo,1,34);
   maxOne(maxNo,2,65);
   maxOne(maxNo,3,73);
   maxOne(maxNo,4,73);
   maxOne(maxNo,5,54);
}
void print4(byte maxNo)
{
   maxOne(maxNo,1,12);
   maxOne(maxNo,2,20);
   maxOne(maxNo,3,36);
   maxOne(maxNo,4,127);
   maxOne(maxNo,5,4);
}
void print5(byte maxNo)
{
   maxOne(maxNo,1,121);
   maxOne(maxNo,2,73);
   maxOne(maxNo,3,73);
   maxOne(maxNo,4,73);
   maxOne(maxNo,5,70);
}
void print6(byte maxNo)
{
   maxOne(maxNo,1,62);
   maxOne(maxNo,2,73);
   maxOne(maxNo,3,73);
```

```
    maxOne(maxNo,4,73);
    maxOne(maxNo,5,46);
}
void print7(byte maxNo)
{
    maxOne(maxNo,1,64);
    maxOne(maxNo,2,64);
    maxOne(maxNo,3,71);
    maxOne(maxNo,4,72);
    maxOne(maxNo,5,112);
}
void print8(byte maxNo)
{
    maxOne(maxNo,1,54);
    maxOne(maxNo,2,73);
    maxOne(maxNo,3,73);
    maxOne(maxNo,4,73);
    maxOne(maxNo,5,54);
}
void print9(byte maxNo)
{
    maxOne(maxNo,1,50);
    maxOne(maxNo,2,73);
    maxOne(maxNo,3,73);
    maxOne(maxNo,4,73);
    maxOne(maxNo,5,62);
}

///////////////functions for printing letters
void printA(byte maxNo )  //test OK
{
    maxOne(maxNo,1,31);
    maxOne(maxNo,2,40);
    maxOne(maxNo,3,72);
    maxOne(maxNo,4,40);
    maxOne(maxNo,5,31);
}
```

```
void printB(byte maxNo )  //test OK
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,73);
   maxOne(maxNo,3,73);
   maxOne(maxNo,4,73);
   maxOne(maxNo,5,54);
}

void printC(byte maxNo)
{
   maxOne(maxNo,1,28);
   maxOne(maxNo,2,34);
   maxOne(maxNo,3,65);
   maxOne(maxNo,4,65);
   maxOne(maxNo,5,34);

}

void printD(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,65);
   maxOne(maxNo,3,65);
   maxOne(maxNo,4,34);
   maxOne(maxNo,5,28);

}

void printE(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,73);
   maxOne(maxNo,3,73);
   maxOne(maxNo,4,73);
   maxOne(maxNo,5,65);
```

```
}

void printF(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,72);
   maxOne(maxNo,3,72);
   maxOne(maxNo,4,72);
   maxOne(maxNo,5,64);

}

void printG(byte maxNo)
{
   maxOne(maxNo,1,30);
   maxOne(maxNo,2,33);
   maxOne(maxNo,3,69);
   maxOne(maxNo,4,69);
   maxOne(maxNo,5,38);

}

void printH(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,8);
   maxOne(maxNo,3,8);
   maxOne(maxNo,4,8);
   maxOne(maxNo,5,127);

}

void printI(byte maxNo)
{
   maxOne(maxNo,1,0);
   maxOne(maxNo,2,65);
```

```
    maxOne(maxNo,3,127);
    maxOne(maxNo,4,65);
    maxOne(maxNo,5,0);

}

void printJ(byte maxNo)
{
    maxOne(maxNo,1,6);
    maxOne(maxNo,2,65);
    maxOne(maxNo,3,127);
    maxOne(maxNo,4,64);
    maxOne(maxNo,5,0);

}

void printK(byte maxNo)
{
    maxOne(maxNo,1,127);
    maxOne(maxNo,2,8);
    maxOne(maxNo,3,20);
    maxOne(maxNo,4,34);
    maxOne(maxNo,5,65);

}

void printL(byte maxNo)
{
    maxOne(maxNo,1,127);
    maxOne(maxNo,2,1);
    maxOne(maxNo,3,1);
    maxOne(maxNo,4,1);
    maxOne(maxNo,5,1);

}

void printM(byte maxNo)
```

```
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,64);
   maxOne(maxNo,3,62);
   maxOne(maxNo,4,64);
   maxOne(maxNo,5,127);

}

void printN(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,48);
   maxOne(maxNo,3,8);
   maxOne(maxNo,4,6);
   maxOne(maxNo,5,127);

}

void printO(byte maxNo)
{
   maxOne(maxNo,1,62);
   maxOne(maxNo,2,65);
   maxOne(maxNo,3,65);
   maxOne(maxNo,4,65);
   maxOne(maxNo,5,62);

}

void printP(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,72);
   maxOne(maxNo,3,72);
   maxOne(maxNo,4,72);
   maxOne(maxNo,5,48);
```

```
}

void printQ(byte maxNo)
{
   maxOne(maxNo,1,62);
   maxOne(maxNo,2,65);
   maxOne(maxNo,3,69);
   maxOne(maxNo,4,66);
   maxOne(maxNo,5,61);

}

void printR(byte maxNo)
{
   maxOne(maxNo,1,127);
   maxOne(maxNo,2,72);
   maxOne(maxNo,3,76);
   maxOne(maxNo,4,74);
   maxOne(maxNo,5,49);

}

void printS(byte maxNo)
{
   maxOne(maxNo,1,34);
   maxOne(maxNo,2,81);
   maxOne(maxNo,3,73);
   maxOne(maxNo,4,73);
   maxOne(maxNo,5,38);

}

void printT(byte maxNo)
{
   maxOne(maxNo,1,64);
   maxOne(maxNo,2,64);
   maxOne(maxNo,3,127);
```

```
    maxOne(maxNo,4,64);
    maxOne(maxNo,5,64);

}

void printU(byte maxNo)
{
    maxOne(maxNo,1,126);
    maxOne(maxNo,2,1);
    maxOne(maxNo,3,1);
    maxOne(maxNo,4,1);
    maxOne(maxNo,5,126);

}

void printV(byte maxNo)
{
    maxOne(maxNo,1,124);
    maxOne(maxNo,2,2);
    maxOne(maxNo,3,1);
    maxOne(maxNo,4,2);
    maxOne(maxNo,5,124);

}

void printW(byte maxNo)
{
    maxOne(maxNo,1,127);
    maxOne(maxNo,2,1);
    maxOne(maxNo,3,31);
    maxOne(maxNo,4,1);
    maxOne(maxNo,5,127);

}

void printX(byte maxNo)
{
```

```
    maxOne(maxNo,1,99);
    maxOne(maxNo,2,20);
    maxOne(maxNo,3,8);
    maxOne(maxNo,4,20);
    maxOne(maxNo,5,99);

}

void printY(byte maxNo)
{
    maxOne(maxNo,1,64);
    maxOne(maxNo,2,32);
    maxOne(maxNo,3,31);
    maxOne(maxNo,4,32);
    maxOne(maxNo,5,64);

}

void printZ(byte maxNo)
{
    maxOne(maxNo,1,67);
    maxOne(maxNo,2,69);
    maxOne(maxNo,3,73);
    maxOne(maxNo,4,81);
    maxOne(maxNo,5,97);

}

void printSpace(byte maxNo)
{
    maxOne(maxNo,1,0);
    maxOne(maxNo,2,0);
    maxOne(maxNo,3,8);
    maxOne(maxNo,4,0);
    maxOne(maxNo,5,0);

}
```

```
void printError()
{
   maxAll(3,127);
}
```

```
char returnOK () {
  // this function checks the response on the serial port to see if it was an "OK" or not
  char incomingChar[3];
  char okString[] = "OK";
  char result = 'n';
  int startTime = millis();

   while (millis() - startTime < 2000 && result == 'n') {   // use a timeout of 10 seconds
    if (Serial.available() > 1)
    {
      // read three incoming bytes which should be "O", "K", and a linefeed:
      for (int i=0; i<3; i++) {
        incomingChar[i] = Serial.read();
      }

      if ( strstr(incomingChar, okString) != NULL )
      { // check to see if the respose is "OK"
       //if (incomingChar[0] == 'O' && incomingChar[1] == 'K') { // check to see if the first two characters are "OK"
        result = 'T'; // return T if "OK" was the response
      }

      else
      {
        result = 'F'; // otherwise return F
      }
```

```
    }
  }
  return result;
}
```

//The code for sending unit

```
char returnOK();

int ledPin = 13; // LED connected to digital pin 13
int incomingByte = 0;
int rx = 0;
int tx = 1;
int switchPin = 7;
void setup()                    // run once, when the sketch starts
{
 pinMode(ledPin, OUTPUT);// sets the digital pin as output
 pinMode(rx, INPUT);
 pinMode(tx, OUTPUT);
 pinMode(switchPin, INPUT);



 Serial.begin(9600);
 delay(2000);

 Serial.print("+++");
 delay(2000);

 if (returnOK() == 'T')
 {
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
 }

 else
 {

   digitalWrite(13, HIGH);
   delay(10000);
   digitalWrite(13, LOW);
   delay(10000);
  setup();
 }

 Serial.print("ATID1234, CN");
```

```
 if (returnOK() == 'T')
 {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
 }
 else
 {
 digitalWrite(13, HIGH);
 delay(10000);
 digitalWrite(13, LOW);
 delay(10000);
 setup();
 }


}




void loop()                    // run over and over again
{
  // send data only when you receive data:
      if (Serial.available() > 0) {
            // read the incoming byte:
            incomingByte = Serial.read();
             digitalWrite(13,HIGH);
             delay(1);
             digitalWrite(13,LOW);
             delay(1);
             //say what you got:
            Serial.print(incomingByte,BYTE);

      }

}



char returnOK () {
 // this function checks the response on the serial port to see if it was
```

```
an "OK" or not
 char incomingChar[3];
 char okString[] = "OK";
 char result = 'n';
 int startTime = millis();

 while (millis() - startTime < 2000 && result == 'n') {  // use a timeout
of 10 seconds
   if (Serial.available() > 1)
   {
    // read three incoming bytes which should be "O", "K", and a linefeed:
    for (int i=0; i<3; i++) {
      incomingChar[i] = Serial.read();
    }

    if ( strstr(incomingChar, okString) != NULL )
    { // check to see if the respose is "OK"
    //if (incomingChar[0] == 'O' && incomingChar[1] == 'K') { // check
to see if the first two characters are "OK"
      result = 'T'; // return T if "OK" was the response
    }

    else
    {
      result = 'F'; // otherwise return F
    }

  }
 }
 return result;
}
```