EL - 203

EHD Project Report

LED Matrix With Game Controller and Scrolling Text

Team Number: 1.3

Group Members:

Sr. No	**Student ID**	**Name**
1	201501012	KEYA DESAI
2	201501013	KOLI POOJA TUKARAM
3	201501014	NIKITA VERMA
4	201501016	SALONI MUNDRA
5	201501017	RAHUL PATEL

Contents

Abstract

The project LED Matrix Display is concerned with the construction of a two dimensional arrangement of LEDs in a rectangular arrangement for the purpose of displaying the English Alphabet, decimal numerals and play games like snake, tetris, etc.

In order to showcase the vast utility of an LED screen, our group decided to do two activities with the same. For the first part of the project, we use a GSM SIM300 module to receive a message in the form of normal text from the user and display it on the LED matrix. For the second part, we implement the classic Nokia Snakes game on the screen. The snake game is played with the help of a controller made using four push buttons which direct the snake in search of food enabling it to grow.

Introduction

Nowadays LED Matrix displays are extensively used on streets, buildings, parks and other public places. Since they are cheaper and more reliable than LCDs and other expensive display devices, they have become a basic way of displaying information visual. Considering the growing popularity of the LED Matrix displays, we decided to build one on our own. To show its versatility, we tried to simulate two applications on it. We also learnt the principles of multiplexing which helped in saving pins on the Arduino board. We also learnt the basics of electronics and microcontroller boards.
The two applications that this project focuses on are:

1. Notice display board controlled by GSM module.
2. Classic Snakes game from Nokia.

Components

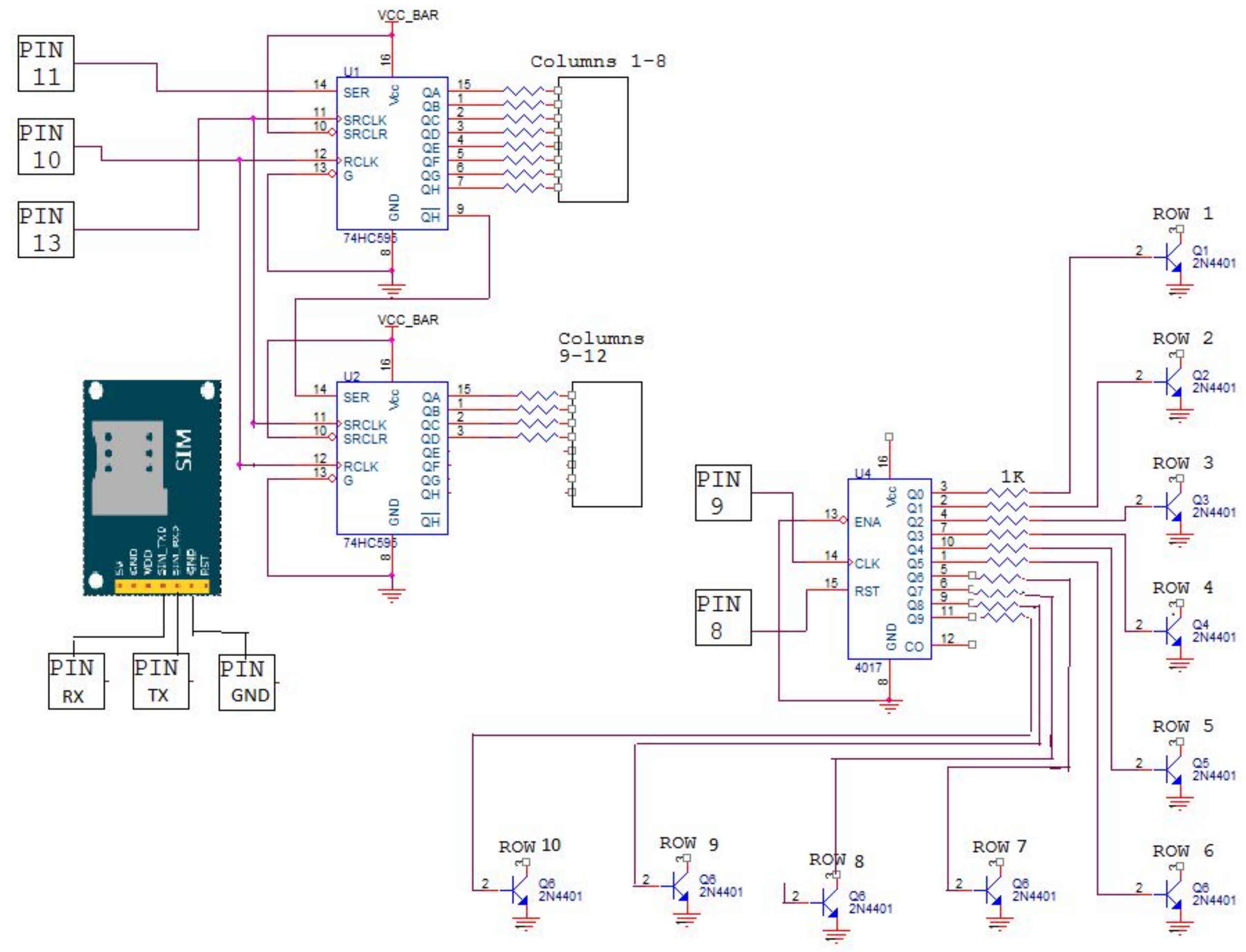
Hardware Specifications:

1. Arduino Uno
2. GSM SIM 300 module
3. 4017 decade counter
4. 2N3904 transistors
5. 74HC595 shift registers
6. 5 mm diffused LEDs
7. Breadboard
8. Header pins
9. Resistors
10. Sim card
11. Connecting Wires
12. Push buttons
13. Soldering Rod and wires

Software Specifications:

1. Arduino IDE

Circuit Diagram



Working

The LEDs are arranged in a 10X12 matrix consisting of 120 dots or pixels. Each pixel is represented by one LED. All the anodes of the same row are connected to one pin and all the cathodes of the same column are connected to another pin. On applying a positive voltage to the row 1 pin and negative to column 1 pin first pixel turns on. If we apply negative to column 2 pin then the second pixel turns on. Like this each pixel can be controlled by hanging the supply pins.
The row pins(10) are given as input to the decade counter and the column pins(12) are given as input to the shift register. 2N3904 transistors are used to drive the decade counter and appropriate connections are made as shown in the circuit diagram. Usage of shift registers and decade counter promotes multiplexing and reduces the input pins to the arduino to five.

The setup uses persistence of vision to draw an image on the array. Each LED required to draw the image is turned on one at a time, and looped very fast, giving the illusion that all LEDs are on at the same time.

For the text message display, the user sends a text message to the SIM inserted in the GSM Module. On receiving the message from the module, the Arduino extracts the content of the text. This text is then displayed on the LED matrix with the help of shift registers and a decade counter with a scrolling effect.

The snakes game is hard coded into the microcontroller and a game controller with four push buttons is created to control the movement of the snake in the four directions. The led matrix is visualised as a 2D matrix and appropriate code is written to drive the game.

Challenges

1. Soldering the LED matrix
2. Configuring the GSM module
3. Extracting the relevant string from the GSM input.

Future Scope

1. With the use of OpenCV and other tools, facial expressions can be classified as happy or sad and displayed as emojis.
2. It can be used to display musical histograms.
3. This can be extended to any application that involves a visual display.

References

1. https://diyhacking.com/arduino-led-matrix/
2. http://extremeelectronics.co.in/avr-tutorials/sending-an-receiving-sms-using-sim300-gsm-module/
3. http://www.instructables.com/id/Make-a-24X6-LED-matrix/

Appendix

Codes:

1. GSM

```
int8_t answer,ans;
int counter,flag=0,co;
long previous;
char aux_str[100],Basic_str[100],SMS[100],SM[50];

byte S[][10]=
{{B00000000,B00111100,B01000010,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010,B00000000},
{B00000000,B01111100,B01000010,B01000010,B01111100,B01000010,B01000010,B01000010,B01111100,B00000000},
{B00000000,B00111110,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B00111110,B00000000},
{B00000000,B01111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B01111100,B00000000},
{B00000000,B01111110,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000,B01111110,B00000000},
{B00000000,B01111110,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000000,B01000111,B01000010,B01000010,B00111100,B00000000},
{B00000000,B01000010,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010,B01000010,B00000000},
{B00000000,B00111000,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00111000,B00000000},
{B00000000,B00011100,B00001000,B00001000,B00001000,B00001000,B01001000,B01001000,B00110000,B00000000},
{B00000000,B01000100,B01001000,B01010000,B01100000,B01010000,B01001000,B01000100,B01000010,B00000000},
{B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01111110,B00000000},
{B00000000,B01000100,B10101010,B10010010,B10010010,B10000010,B10000010,B10000010,B10000010,B00000000},
{B00000000,B01000010,B01100010,B01010010,B01001010,B01001010,B01001010,B01000110,B01000010,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01111100,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000110,B00111110,B00000001},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01111100,B01000100,B01000010,B01000010,B00000000},
{B00000000,B00111100,B01000010,B01000000,B01000000,B00111100,B00000010,B01000010,B00111100,B00000000},
{B00000000,B11111110,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00100100,B00011000,B00000000},
{B00000000,B10000010,B10000010,B10000010,B10000010,B10010010,B10010010,B10101010,B01000100,B00000000},
{B00000000,B01000010,B01000010,B00100100,B00011000,B00011000,B00100100,B01000010,B01000010,B00000000},
{B00000000,B10000010,B01000100,B00101000,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B01111110,B00000010,B00000100,B00001000,B00010000,B00100000,B01000000,B01111110,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00111000,B01000100,B01000101,B01000101,B00111010,B00000000},
{B00000000,B00000000,B00100000,B00100000,B00100000,B00111100,B00100010,B00100010,B00111100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00111100,B01000000,B01000000,B01000000,B00111100,B00000000},
{B00000000,B00000000,B00000100,B00000100,B00000100,B00111100,B01000100,B01000100,B00111100,B00000000},
{B00000000,B00000000,B00000000,B00111000,B01000100,B01000100,B01111000,B01000000,B00111100,B00000000},
{B00000000,B00011000,B00100100,B00100000,B00100000,B01110000,B00100000,B00100000,B00100000,B00000000},
{B00000000,B00011100,B00100010,B00100010,B00011110,B00000010,B00000010,B00010010,B00001100,B00000000},
{B00000000,B01000000,B01000000,B01000000,B01000000,B01111000,B01000100,B01000100,B01000100,B00000000},
{B00000000,B00000000,B00010000,B00000000,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B00000000,B00010000,B00000000,B00010000,B00010000,B00010000,B01010000,B00110000,B00000000},
{B00000000,B00000000,B00000000,B01001000,B01010000,B01100000,B01100000,B01010000,B01001000,B00000000},
{B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00000000,B00000000,B00110100,B01001010,B01001010,B01001010,B01001010,B01001010,B00000000},
{B00000000,B00000000,B00000000,B01111000,B01000100,B01000100,B01000100,B01000100,B01000100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00011100,B00100010,B00100010,B00100010,B00011100,B00000000},
{B00000000,B00000000,B00000000,B00011100,B00100010,B00100010,B00111100,B00100000,B00100000,B00100000},
{B00000000,B00000000,B00000000,B00111000,B01000100,B01000100,B00111100,B00000100,B00000100,B00000100},
{B00000000,B00000000,B00000000,B00111000,B01000000,B01000000,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00000000,B00111000,B01000100,B01000000,B00111000,B00000100,B01000100,B00111000,B00000000},
{B00000000,B00100000,B00100000,B00100000,B01111000,B00100000,B00100000,B00100010,B00011100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B01000100,B01000100,B01000100,B01000100,B00111000,B00000000},
{B00000000,B00000000,B00000000,B01000100,B01000100,B01000100,B01000100,B00101000,B00010000,B00000000},
{B00000000,B00000000,B00000000,B00000000,B01000100,B01000100,B01010100,B01010100,B00101000,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00000000,B00100100,B00011000,B00011000,B00100100,B00000000},
{B00000000,B00000000,B01000100,B01000100,B00111100,B00000100,B00000100,B00000100,B00111000,B00000000},
{B00000000,B00000000,B00000000,B00000000,B01111100,B00001000,B00010000,B00100000,B01111100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000},
{B00000000,B00111100,B01000110,B01001010,B01001010,B01001010,B01010010,B01100010,B00111100,B00000000},
{B00000000,B00001000,B00011000,B00001000,B00001000,B00001000,B00001000,B00001000,B00011100,B00000000},
{B00000000,B00111100,B01000010,B00000100,B00001000,B00010000,B00100000,B01000000,B01111110,B00000000},
{B00000000,B01111110,B00000010,B00000010,B00011100,B00000010,B00000010,B01000010,B00111100,B00000000},
{B00000000,B00000100,B00001100,B00010100,B00100100,B01000100,B01111110,B00000100,B00000100,B00000000},
{B00000000,B01111110,B01000000,B01000000,B00111100,B00000010,B00000010,B00000010,B01111100,B00000000},
{B00000000,B00111100,B01000000,B01000000,B01111100,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B01111110,B00000010,B00000100,B00001000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B00111100,B01000010,B01000010,B00111100,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B00111110,B00000010,B00000010,B00111100,B00000000},
{B00000000,B00000000,B01000010,B00100100,B00011000,B00011000,B00100100,B01000010,B00000000,B00000000}};

int latchPin = 10;
int clockPin = 13;
int dataPin = 11;
int clock = 9;
int Reset = 8;
int latchPinPORTB = latchPin - 8;
int clockPinPORTB = clockPin - 8;
int dataPinPORTB = dataPin - 8;
int i = 0;
long scrolling_word[50];
int array_turn=0;


byte your_text[50][10];

void setup(){

  //Serial.begin(9600);
```

```
//pinMode(LED_BUILTIN, OUTPUT);
 Serial.begin(9600);
 Serial.println("Starting...");
 power_on();

 while (sendATcommand("AT+CREG?", "+CREG: 0,1", 2000) == 0);
  sendATcommand("AT+CMGF=1", "OK", 20000);

  for(int i=1;i<5;i++)
  {
   snprintf(aux_str, sizeof(aux_str), "AT+CMGD=%d",i);
   sendATcommand(aux_str, "OK", 2000);
  }


  pinMode(dataPin,OUTPUT);

 pinMode(clockPin,OUTPUT);

 pinMode(latchPin,OUTPUT);

 pinMode(clock,OUTPUT);

 pinMode(Reset,OUTPUT);

 digitalWrite(Reset,HIGH);

 digitalWrite(Reset,LOW);
 setupSPI();

}

 void loop() {
     while( Serial.available() > 0)
    Serial.read();
     memset(SM, '\0', 50);
  delay(100);
  snprintf(aux_str, sizeof(aux_str), "AT+CMGR=1");
      ans=sendATcommand(aux_str, "+CMGR", 2000);
      co=0;
      counter = 0;
      answer = 0;
      memset(SMS, '\0', 100);
      memset(SM, '\0', 50);// Initialize the string
      previous = millis();

      if(ans==1)
      {
       do
```

```
        {
            if(Serial.available() != 0)
            {
                SMS[counter] = Serial.read();
                counter++;
              if(strstr(SMS, "OK") != NULL)
                  answer = 1;
            }
          }while((answer == 0) && ((millis() - previous) < 2000));

          SMS[counter-3] = '\0';
          for(int i=1;i<5;i++)
          {
            snprintf(aux_str, sizeof(aux_str), "AT+CMGD=%d",i);
            sendATcommand(aux_str, "OK", 2000);
          }
      }

      //if (SMS[0]=='=') memset(SMS, '\0', 100);
      int cnt=0;
      int index=0;
      while (index<100 && cnt<2)
      {
        if (SMS[index]=='\n') cnt++;
        index++;
      }
      //SM=SMS+index;
      Serial.print(SMS+index);
      //int j=0;
      for(int i=index;SMS[i]!='\0';i++){
      SM[co]=SMS[i];
      co++;
      }
      Serial.println("Print message: ");
      for(int i=0;i<co;i++)
      Serial.print(SM[i]);
      Serial.println();

  delay(100);


     if(co!=0){
     ParseText();
      delay(1000);
      for(int i=0;i<2;i++)
     display_word(1,your_text,co,15);// calls for the display_pattern function and says that int loop = 15(if you do more loop the
pattern whould scrole slower).
      memset(SM, '\0', 50);
      //delay(100);
      co=0;
     }
     delay(100);
     }

void ParseText(){
  //int j=0;
  for(int i=0;i<co;i++){
    if(SM[i]>='A' && SM[i]<='Z'){
     int x=SM[i]-'A';
     //ind[i]=x;
     for(int j=0;j<10;j++){
      your_text[i][j]=S[x][j];
     }

    }
    else if(SM[i]>='a' && SM[i]<='z'){
     int x=26+SM[i]-'a';
     for(int j=0;j<10;j++){
      your_text[i][j]=S[x][j];
     }
    }
    else if(SM[i]==' '){
     int x=52;
     for(int j=0;j<10;j++){
      your_text[i][j]=S[x][j];
     }
    }
    else if(SM[i]>='0' && SM[i]<='9'){
     int x=53+SM[i]-'0';
    for(int j=0;j<10;j++){
      your_text[i][j]=S[x][j];
     }
    }
  }

}
/*void Convert(){
  for(int i=0;i<counter;i++){
    your_text
  }
}*/

void display_word(int loops,byte word_print[][10],int num_patterns,int delay_length){// this function displays your symbols

  i = 0;// resets the counter fot the 4017

  for(int g=0;g<10;g++)//resets the the long int where your word goes

    scrolling_word[g] = 0;
  for(int x=0;x<num_patterns;x++){//main loop, goes over your symbols

   for(int r=0;r<10;r++)//puts the buildes the first symbol

      scrolling_word[r] |= word_print[x][r];

    for (int z=0;z<10;z++){//the sctolling action

        for(int p=0;p<10;p++)

          scrolling_word[p] = scrolling_word[p] << 1;

// end of the scrolling funcion

      for(int t=0;t<delay_langth;t++){// delay function
        for(int y=0;y<10;y++){// scaning the display

          if(i == 10){// counting up to 6 with the 4017

            digitalWrite(Reset,HIGH);

            digitalWrite(Reset,LOW);

            i = 0;

          }

          latchOff();

          spi_transfer(make_word(0x01000000,y));// sending the data

          spi_transfer(make_word(0x00010000,y));

          spi_transfer(make_word(0x00000100,y));

          latchOn();

          delayMicroseconds(800);//waiting a bit

          latchOff();

          spi_transfer(0);// clearing the data

          spi_transfer(0);

          spi_transfer(0);

          latchOn();

          digitalWrite(clock,HIGH);//counting up with the 4017
        digitalWrite(clock,LOW);

        i++;

      }

    }

  }

 }

 finish_scroll(delay_langth);

}


void finish_scroll(int delay_scroll){// this function is the same as the funcion above, it just finishing scrolling

 for (int n=0;n<12;n++){

      for(int h=0;h<10;h++)

        scrolling_word[h] = scrolling_word[h] << 1;

     for(int w=0;w<delay_scroll;w++){

      for(int k=0;k<10;k++){

        if(i == 10){

          digitalWrite(Reset,HIGH);

          digitalWrite(Reset,LOW);

          i = 0;

        }

        latchOff();

        spi_transfer(make_word(0x01000000,k));

        spi_transfer(make_word(0x00010000,k));

        spi_transfer(make_word(0x00000100,k));

        latchOn();
```

```
         delayMicroseconds(800);

         latchOff();

         spi_transfer(0);

         spi_transfer(0);

         spi_transfer(0);

         latchOn();

         digitalWrite(clock,HIGH);

         digitalWrite(clock,LOW);

         i++;

       }

     }

   }

}


byte make_word (long posistion,byte turn){

  byte dummy_word = 0;

  for(int q=0;q<8;q++){

    if(scrolling_word[turn] & (posistion<<q))

      dummy_word |= 0x01<<q;

  }

  return dummy_word;

}
void latchOn(){

  bitSet(PORTB,latchPinPORTB);

}
```

```
void latchOff(){

 bitClear(PORTB,latchPinPORTB);

}

void setupSPI(){

 byte clr;

 SPCR |= ( (1<<SPE) | (1<<MSTR) ); // enable SPI as master

 //SPCR |= ( (1<<SPR1) | (1<<SPR0) ); // set prescaler bits

 SPCR &= ~( (1<<SPR1) | (1<<SPR0) ); // clear prescaler bits

 clr=SPSR; // clear SPI status reg

 clr=SPDR; // clear SPI data reg

 SPSR |= (1<<SPI2X); // set prescaler bits

 //SPSR &= ~(1<<SPI2X); // clear prescaler bits



 delay(40);

}

byte spi_transfer(byte data)

{

 SPDR = data;        // Start the transmission

 while (!(SPSR & (1<<SPIF)))     // Wait the end of the transmission

 {

 };

 return SPDR;        // return the received byte, we don't need that

}
void power_on()
{
 uint8_t answer=0;
```

```
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0)
  {
    while(answer == 0)
      answer = sendATcommand("AT", "OK", 2000);
  }
}

int8_t sendATcommand(const char* ATcommand, const char* expected_answer, unsigned int timeout)
{
  uint8_t x=0,  answer=0;
  char response[100];
  unsigned long previous;
  memset(response, '\0', 100);    // Initialize the string
  delay(100);
  while( Serial.available() > 0)
        Serial.read();    // Clean the input buffer

  Serial.println(ATcommand);    // Send the AT command

  x = 0;
  previous = millis();

      do
  {
          if(Serial.available() != 0)
    {
                response[x] = Serial.read();
                x++;
                // check if the desired answer  is in the response of the module
                if (strstr(response, expected_answer) != NULL)
                        answer = 1;
          }
        } while((answer == 0) && ((millis() - previous) < timeout));

  return answer;
}
```

2. Snakes

```
byte S[][10]=
{{B00000000,B00111100,B01000010,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010,B00000000},
{B00000000,B01111100,B01000010,B01000010,B01111100,B01000010,B01000010,B01000010,B01111100,B00000000},
{B00000000,B00111110,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B00111110,B00000000},
{B00000000,B01111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B01111100,B00000000},
{B00000000,B01111110,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000,B01111110,B00000000},
{B00000000,B01111110,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000000,B01000111,B01000010,B01000010,B00111100,B00000000},
{B00000000,B01000010,B01000010,B01000010,B01111110,B01000010,B01000010,B01000010,B01000010,B00000000},
{B00000000,B00111000,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00111000,B00000000},
{B00000000,B00011100,B00001000,B00001000,B00001000,B00001000,B01001000,B01001000,B00110000,B00000000},
{B00000000,B01000100,B01001000,B01010000,B01100000,B01010000,B01001000,B01000100,B01000010,B00000000},
{B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01111110,B00000000},
{B00000000,B01000100,B10101010,B10010010,B10010010,B10000010,B10000010,B10000010,B10000010,B00000000},
{B00000000,B01000010,B01100010,B01010010,B01001010,B01001010,B01001010,B01000110,B01000010,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01111100,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000110,B00111110,B00000001},
{B00000000,B00111100,B01000010,B01000010,B01000010,B01111100,B01000100,B01000010,B01000010,B00000000},
{B00000000,B00111100,B01000010,B01000000,B01000000,B00111100,B00000010,B01000010,B00111100,B00000000},
{B00000000,B11111110,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00100100,B00011000,B00000000},
{B00000000,B10000010,B10000010,B10000010,B10000010,B10010010,B10010010,B10101010,B01000100,B00000000},
{B00000000,B01000010,B01000010,B00100100,B00011000,B00011000,B00100100,B01000010,B01000010,B00000000},
{B00000000,B10000010,B01000100,B00101000,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B01111110,B00000010,B00000100,B00001000,B00010000,B00100000,B01000000,B01111110,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00111000,B01000100,B01000101,B01000101,B00111010,B00000000},
{B00000000,B00000000,B00100000,B00100000,B00100000,B00111100,B00100010,B00100010,B00111100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00111100,B01000000,B01000000,B01000000,B00111100,B00000000},
{B00000000,B00000000,B00000100,B00000100,B00000100,B00111100,B01000100,B01000100,B00111100,B00000000},
{B00000000,B00000000,B00000000,B00111000,B01000100,B01000100,B01111000,B01000000,B00111100,B00000000},
{B00000000,B00011000,B00100100,B00100000,B00100000,B01110000,B00100000,B00100000,B00100000,B00000000},
{B00000000,B00011100,B00100010,B00100010,B00011110,B00000010,B00000010,B00010010,B00001100,B00000000},
{B00000000,B01000000,B01000000,B01000000,B01000000,B01111000,B01000100,B01000100,B01000100,B00000000},
{B00000000,B00000000,B00010000,B00000000,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B00000000,B00010000,B00000000,B00010000,B00010000,B00010000,B01010000,B00110000,B00000000},
{B00000000,B00000000,B00000000,B01001000,B01010000,B01100000,B01100000,B01010000,B01001000,B00000000},
{B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00000000,B00000000,B00110100,B01001010,B01001010,B01001010,B01001010,B01001010,B00000000},
{B00000000,B00000000,B00000000,B01111000,B01000100,B01000100,B01000100,B01000100,B01000100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00011100,B00100010,B00100010,B00100010,B00011100,B00000000},
{B00000000,B00000000,B00000000,B00011100,B00100010,B00100010,B00111100,B00100000,B00100000,B00100000},
{B00000000,B00000000,B00000000,B00111000,B01000100,B01000100,B00111100,B00000100,B00000100,B00000100},
{B00000000,B00000000,B00000000,B00111000,B01000000,B01000000,B01000000,B01000000,B01000000,B00000000},
{B00000000,B00000000,B00111000,B01000100,B01000000,B00111000,B00000100,B01000100,B00111000,B00000000},
{B00000000,B00100000,B00100000,B00100000,B01111000,B00100000,B00100000,B00100010,B00011100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B01000100,B01000100,B01000100,B01000100,B00111000,B00000000},
{B00000000,B00000000,B00000000,B01000100,B01000100,B01000100,B01000100,B00101000,B00010000,B00000000},
{B00000000,B00000000,B00000000,B00000000,B01000100,B01000100,B01010100,B01010100,B00101000,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00000000,B00100100,B00011000,B00011000,B00100100,B00000000},
{B00000000,B00000000,B01000100,B01000100,B00111100,B00000100,B00000100,B00000100,B00111000,B00000000},
{B00000000,B00000000,B00000000,B00000000,B01111100,B00001000,B00010000,B00100000,B01111100,B00000000},
{B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000,B00000000},
{B00000000,B00111100,B01000110,B01001010,B01001010,B01001010,B01010010,B01100010,B00111100,B00000000},
{B00000000,B00001000,B00011000,B00001000,B00001000,B00001000,B00001000,B00001000,B00011100,B00000000},
{B00000000,B00111100,B01000010,B00000100,B00001000,B00010000,B00100000,B01000000,B01111110,B00000000},
{B00000000,B01111110,B00000010,B00000010,B00011100,B00000010,B00000010,B01000010,B00111100,B00000000},
{B00000000,B00000100,B00001100,B00010100,B00100100,B01000100,B01111110,B00000100,B00000100,B00000000},
{B00000000,B01111110,B01000000,B01000000,B00111100,B00000010,B00000010,B00000010,B01111100,B00000000},
{B00000000,B00111100,B01000000,B01000000,B01111100,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B01111110,B00000010,B00000100,B00001000,B00010000,B00010000,B00010000,B00010000,B00000000},
{B00000000,B00111100,B01000010,B01000010,B00111100,B01000010,B01000010,B01000010,B00111100,B00000000},
{B00000000,B00111100,B01000010,B01000010,B01000010,B00111110,B00000010,B00000010,B00111100,B00000000},
{B00000000,B00000000,B01000010,B00100100,B00011000,B00011000,B00100100,B01000010,B00000000,B00000000}};
struct snakes
{
  int x;
  int y;
};

int latchPin = 10;
int clockPin = 13;
int dataPin = 11;
int clck = 9;
int reset = 8;
int upb=4;
int downb=6;
int leftb=7;
int rightb=12;
int x,y;
int foodx,foody;
int score=0;

int latchPinPORTB = latchPin - 8;
int clockPinPORTB = clockPin - 8;
int dataPinPORTB = dataPin - 8;

int i = 0;

long scrolling_word[10];

int array_turn=0;
```

```
byte your_text[25][10];
int co=0;
char SM[25];
snakes body[13*11];
int snakex=5,snakey=9;
boolean up=false,down=false,left=false,right=false;
boolean ustate=false,dstate=false,lstate=false,rstate=false;
boolean matrix[12][10];
uint16_t displaymatrix[10];
int len=1;

void setup()
{
  Serial.begin(9600);
  pinMode(dataPin,OUTPUT);
  pinMode(clockPin,OUTPUT);
  pinMode(latchPin,OUTPUT);
  pinMode(clck,OUTPUT);
  pinMode(reset,OUTPUT);
  pinMode(upb,INPUT);
  pinMode(downb,INPUT);
  pinMode(leftb,INPUT);
  pinMode(rightb,INPUT);

  digitalWrite(reset,HIGH);
  digitalWrite(reset,LOW);
  randomSeed(analogRead(0));
  //setupSPI();

}

void loop()
{

  Serial.println("In Loop");
  resetmatrix();
  digitalWrite(reset,HIGH);
  digitalWrite(reset,LOW);
  snakex=5,snakey=9;
  body[0].x=snakex;
  body[0].y=snakey;
  len=1;
  spawnfood();

  left=right=up=down=false;

  ustate=false,dstate=false,lstate=false,rstate=false;
```

```
 Serial.println("Loop 2");
 while (!(up=digitalRead(upb)) && !(down=digitalRead(downb)) && !(left=digitalRead(leftb)) &&
!(right=digitalRead(rightb)))
 {
  refreshmatrix();
  disp();
 }

 Serial.println("start");
 while (true)
 {
  refreshmatrix();
  disp();

  if (selfeaten()){
   Serial.println("Final Score: ");
    Serial.println(score);
   delay(100);
   game_over();
   //delay(1000);
    score=0;
    break;
  }

  ustate=digitalRead(upb);
  dstate=digitalRead(downb);
  lstate=digitalRead(leftb);
  rstate=digitalRead(rightb);

  //Serial.println("Inside");
  if (ustate || dstate || lstate || rstate)
  {
   if ((ustate && !down) || (dstate && !up) || (lstate && !right) || (rstate && !left))
   {
    up=ustate;
    down=dstate;
    left=lstate;
    right=rstate;
   }
  }

  snakey+=up;
  snakey-=down;
  snakex+=right;
  snakex-=left;

  snakex=(12+snakex)%12;
  snakey=(10+snakey)%10;
```

```
    if (snakex==foodx && snakey==foody)
    {
      score++;
      Serial.println(score);
      spawnfood();
    }
    else  removelast();
    addnew();

    resetmatrix();
  }
}

void spawnfood()
{
  do
  {
    foodx=random(12);
    foody=random(10);
  } while (eaten(foodx,foody));
}void game_over(){

  SM[0]='S';
  SM[1]='C';
  SM[2]='O';
  SM[3]='R';
  SM[4]='E';
 // SM[5]=' ';
  int i=5;

  String myString="";
  myString+=score;
  char te[10];
  itoa(score,te,10);
  Serial.println("String Score: ");
  Serial.println(te);
  int n=score;
  int cx=0;
  while (n)
  {
    n=n/10;
    cx++;
  }

  //int n=myString.len();
  for(int j=0;j<cx;j++){
    Serial.println(te[j]);
```

```
   SM[i]=te[j];
   i++;
  }
  co=5+cx;
  ParseText();
  delay(100);
  display_word(1,your_text,co,15);
}
void shftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint16_t val)
{
  uint8_t i;

  for (i=0;i<16;i++)
  {
   if (bitOrder==LSBFIRST) digitalWrite(dataPin,!!(val & (1<<i)));
   else  digitalWrite(dataPin,!!(val & (1<<(15-i))));

   digitalWrite(clockPin,HIGH);
   digitalWrite(clockPin,LOW);
  }
}
void display_word(int loops,byte word_print[][10],int num_patterns,int delay_length)
{
  for (int x=0;x<num_patterns;x++)
  {
   for (int z=0;z<12;z++)
   {
    for (int t=0;t<delay_length;t++)
    {
     for (int y=0;y<10;y++)
     {
      byte temp=word_print[x][y];
      byte temp_2=word_print[x+1][y];
      digitalWrite(latchPin,LOW);
      if (x==num_patterns-1)  shftOut(dataPin,clockPin, MSBFIRST,temp<<z);
      else  shftOut(dataPin,clockPin, MSBFIRST,((temp<<z)|(temp_2>>(12-z))));
      digitalWrite(latchPin,HIGH);
      delayMicroseconds(800);
      digitalWrite(latchPin,LOW);
      shftOut(dataPin,clockPin,MSBFIRST,0);
      digitalWrite(latchPin,HIGH);
      digitalWrite(clck,HIGH);
      digitalWrite(clck,LOW);
     }
    }
   }
  }
}
```

```
void ParseText(){
  //int j=0;
  for(int i=0;i<co;i++){
    if(SM[i]>='A' && SM[i]<='Z'){
      int x=SM[i]-'A';
      //ind[i]=x;
      for(int j=0;j<10;j++){
        your_text[i][j]=S[x][j];
      }

    }
    else if(SM[i]>='a' && SM[i]<='z'){
      int x=26+SM[i]-'a';
      for(int j=0;j<10;j++){
        your_text[i][j]=S[x][j];
      }
    }
    else if(SM[i]==' '){
      int x=52;
      for(int j=0;j<10;j++){
        your_text[i][j]=S[x][j];
      }
    }
    else if(SM[i]>='0' && SM[i]<='9'){
      int x=53+SM[i]-'0';
     for(int j=0;j<10;j++){
        your_text[i][j]=S[x][j];
      }
    }
  }

}
boolean eaten(int tx, int ty)
{
  for (int i=0;i<len;i++)
  {
    if (body[i].x==tx && body[i].y==ty) return true;
  }
  return false;
}

boolean selfeaten()
{
  for (int i=0;i<len-1;i++)
  {
    if (body[i].x==snakex && body[i].y==snakey) return true;
  }
  return false;
}
void resetmatrix()
{
  for (int x=0;x<12;x++)
  {
    for (int y=0;y<10;y++)
    {
      matrix[x][y]=false;
      displaymatrix[y]=0;
    }
  }
}

void refreshmatrix()
{
  int i=0;

  for (;i<len;i++) matrix[body[i].x][body[i].y]=true;

  matrix[foodx][foody]=true;
}


void disp()
{
  //digitalWrite(reset,HIGH);
  //digitalWrite(reset,LOW);
  for (int i=0;i<12;i++)
  {
    for (int j=0;j<10;j++)
    {
      if (matrix[i][j])
      displaymatrix[j]|=(1<<i);
    }
  }

  for (int j=0;j<25;j++)
  {
    for (int i=0;i<10;i++)
    {
      digitalWrite(latchPin,LOW);
      shftOut(dataPin,clockPin,MSBFIRST,displaymatrix[i]);
      digitalWrite(latchPin,HIGH);

      delayMicroseconds(10);

      digitalWrite(latchPin,LOW);
      shftOut(dataPin,clockPin,MSBFIRST,0);
      digitalWrite(latchPin,HIGH);

      digitalWrite(clck,HIGH);
      digitalWrite(clck,LOW);
    }
  }
}

void addnew()
{
  body[len].y=snakey;
  body[len].x=snakex;

  len++;
}

void removelast()
{
  int i=-1;

  for (i=1;i<len;i++)
  {
    body[i-1].x=body[i].x;
    body[i-1].y=body[i].y;
  }

  len--;
}
```