```
/*
 *
 * Tutorial sketch for use of character OLED slim display family by Newhaven with Ardui
Uno, without
 * using any library.  Models: NHD0420CW-Ax3, NHD0220CW-Ax3, NHD0216CW-Ax3. Controller:
US2066
 * in this example, the display is connected to Arduino via SPI interface.
 *
 * Displays on the OLED alternately a 4-line message and a sequence of character "block"
 * This sketch assumes the use of a 4x20 display; if different, modify the values of the
two variables
 * ROW_N e COLUMN_N.
 * The sketch uses the minimum possible of Arduino's pins; if you intend to use also /RE
or /CS lines,
 * the related instructions are already present, it's sufficient to remove the comment
markers.
 *
 * The circuit:
 * OLED pin 1 (Vss)     to Arduino pin ground
 * OLED pin 2 (VDD)     to Arduino pin 3V
 * OLED pin 3 (REGVDD) to Arduino pin 3V
 * OLED pin 4 to 6      to Vss ground
 * OLED pin 7 (SCLK)    to Arduino pin D13 (SCK)
 * OLED pin 8 (SID)     to Arduino pin D11 (MOSI)
 * OLED pin 9 (SOD)     to Arduino pin D12 (MISO) (optional, can be not connected)
 * OLED pin 10 to 14    to Vss ground
 * OLED pin 15 (/CS)    to Vss ground  (or to Arduino pin D2, in case of use of more than
one display)
 * OLED pin 16 (/RES)   to Arduino pin Reset or VDD 5V (or to Arduino pin D3, to control
reset by sw)
 * OLED pin 17 (BS0)    to Vss ground
 * OLED pin 18 (BS1)    to Vss ground
 * OLED pin 19 (BS2)    to Vss ground
 * OLED pin 20 (Vss)    to Vss ground
 *
 *
 * This example code is in the public domain.
 * Line 1: 0x80
 * Line 2: 0xA0
 * Line 3: 0xC0
 * Line 4: 0xD0
 */
// include the SPI library:
#include <SPI.h>
#include <stdio.h>
#include  <stdlib.h>
#include  <string.h>

byte rows = 0x08;                       // Display mode: 1/3 lines or 2/4 lines; default
```

```
(0x08)
int up = 13, down = 12, left = 11, right = 10, ok = 6;

//

void command(byte c)                    // SUBROUTINE: PREPARES THE TRANSMISSION OF A COM
{
   SPI.transfer(0x1F);

  send_byte(c);                        // Transmits the byte
}



//

void data(byte d)
{
   SPI.transfer(0x5F);

   send_byte(d);
}

//

void send_byte(byte tx_b)
{
  //Split the bytes into two and pad the last four bits with 0s
  byte tx_b1 = tx_b & 0x0F;
  byte tx_b2 = (tx_b >> 4) & 0x0F;

  //Or together the bytes
  int tx_int = (tx_b2<<8)|tx_b1;

  //transfer it
   SPI.transfer16(tx_int);
}

//

const byte line[4] = {0x80, 0xA0, 0xC0, 0xE0};

void LCD_print(String ip, byte line)
{
   clearLCDLine(line);
   command(line);
  int i = 0;
  while (ip[i] != '\0')
    {
        data(ip[i]);
        i++;
    }

   command(line);
}

void clearLCDLine(byte line)
{
   command(line);
  char empty[] = {"                    "};
  int i = 0;
  while (empty[i] != '\0')
    {
        data(empty[i]);
        i++;
    }
   command(line);
}

void setup() {
  //OLED SETUP
   SPI.setBitOrder(LSBFIRST);
    SPI.setClockDivider(SPI_CLOCK_DIV2);
   SPI.setDataMode(SPI_MODE3);
  delayMicroseconds(200);               // Waits 200 us for stabilization purpose

  command(0x22 | rows); // Function set: extended command set (RE=1), lines #
  command(0x71);          // Function selection A:
  data(0x5C);              //  enable internal Vdd regulator at 5V I/O mode (def. value) (
for disable, 2.8V I/O)
  command(0x20 | rows); // Function set: fundamental command set (RE=0) (exit from exte
command set), lines #
  command(0x08);          // Display ON/OFF control: display off, cursor off, blink off
(default values)
  command(0x22 | rows); // Function set: extended command set (RE=1), lines #
  command(0x79);          // OLED characterization: OLED command set enabled (SD=1)
  command(0xD5);          // Set display clock divide ratio/oscillator frequency:
  command(0x70);          //  divide ratio=1, frequency=7 (default values)
  command(0x78);          // OLED characterization: OLED command set disabled (SD=0) (exi
from OLED command set)

  command(0x06);          // Entry Mode set - COM/SEG direction: COM0->COM31, SEG99->SEG0
(BDC=1, BDS=0)
  command(0x72);          // Function selection B:
  data(0x0A);              //  ROM/CGRAM selection: ROM C, CGROM=250, CGRAM=6 (ROM=10, OPR
  command(0x79);          // OLED characterization: OLED command set enabled (SD=1)
  command(0xDA);          // Set SEG pins hardware configuration:
  command(0x10);          //  alternative odd/even SEG pin, disable SEG left/right remap
(default values)
  command(0xDC);          // Function selection C:
  command(0x00);          //  internal VSL, GPIO input disable
  command(0x81);          // Set contrast control:
  command(0x7F);          //  contrast=127 (default value)
  command(0xD9);          // Set phase length:
  command(0xF1);          //  phase2=15, phase1=1 (default: 0x78)
  command(0xDB);          // Set VCOMH deselect level:
  command(0x40);          //  VCOMH deselect level=1 x Vcc (default: 0x20=0,77 x Vcc)
  command(0x78);          // OLED characterization: OLED command set disabled (SD=0) (exi
from OLED command set)
  command(0x20 | rows); // Function set: fundamental command set (RE=0) (exit from exte
command set), lines #
  command(0x01);          // Clear display
  delay(2);               // After a clear display, a minimum pause of 1-2 ms is required
  command(0x80);          // Set DDRAM address 0x00 in address counter (cursor home) (def
value)
  command(0x0C);          // Display ON/OFF control: display ON, cursor OFF, blink OFF
  delay(250);             // Waits 250 ms for stabilization purpose after display on

  //Serial Setup
   Serial.begin(9600);

    //PUSH-BUTTONs  SETUP-----------------------------------
   pinMode(up, INPUT_PULLDOWN); //UP pin 13
   pinMode(down, INPUT_PULLDOWN); //DOWN pin 12
   pinMode(left, INPUT_PULLDOWN); //LEFT pin 11
   pinMode(right, INPUT_PULLDOWN); //RIGHT pin 10
   pinMode(ok, INPUT_PULLDOWN); //OK pin 6
   attachInterrupt (digitalPinToInterrupt (ok), check_ok, CHANGE);
}
/*********************************************************************************************
*******************************/
int count = 0, ok_flag = 0, obj = 1;

String mode, host, rcv_ip, rcv_subnet, rcv_gateway, rcv_ok;

String tx_ip = "192.168.100.100", tx_subnet = "255.255.000.000", tx_gateway = "000.000.
000";

void check_ok(void) //Interrupt Routine
{
  if(digitalRead(ok) == HIGH)
  {
     while(digitalRead(ok) == HIGH);
     ok_flag = !ok_flag;
  }
}

void  loop()
{
    /*----------------------Configure  Mode-------------------------*/
  if (ok_flag == 1)
  {
     command(0x01);
     delay(2);
     command(0x0F);
     LCD_print("SET MODE", line[0]);
     LCD_print("1. DHCP", line[1]);
     LCD_print("2. Static", line[2]);
     command(line[1]);

     while (ok_flag == 1)
       {
         mode_updown();

         if (ok_flag == 0 && obj == 1)
           {
             setDHCP();
             goto displaymode;
           }

         if (ok_flag == 0 && obj == 2)
           {
             command(0x01);
             delay(2);
              LCD_print("SET IPv4 Address", line[0]);
              LCD_print(tx_ip, line[1]);
              LCD_print("Def: 192.168.100.100", line[3]);
             command(line[1]);
             count = 0;
           }

         while (ok_flag == 0 && obj == 2)
           {
             command(0x0E);
             left_n_right();
             up_n_down(tx_ip);

             if (ok_flag == 1)
               {
                 command(0x01);
                 delay(2);
                  LCD_print("SET IPv4 Netmask", line[0]);
                  LCD_print(tx_subnet, line[1]);
```

```
                   LCD_print("Def: 255.255.0.0", line[3]);
                  command(line[1]);
                 count = 0;
               }

             while (ok_flag == 1)
               {
                 command(0x0E);
                  left_n_right();
                  up_n_down(tx_subnet);

                 if (ok_flag == 0)
                   {
                     setStatic();
                       tx_ip = "192.168.100.100", tx_subnet = "255.255.000.000";
                     goto displaymode;
                   }
               }
          }
      }
  }
/*-------------------------------Display      Mode-----------------------------------------
   displaymode:
  if (ok_flag == 0)
  {
     command(0x01);
    delay(2);
     command(0x0C);
    while(ok_flag == 0)
    {
       getMode();
       getHost();
       getIP();
       getGate();
       command(0x0C);
    }
  }
}

/******************************************Display Mode
Functions*********************************/
void  getMode()
{
  mode = "";

   while  (!Serial.available()){
     Serial.println("Mode");
  }
  while (Serial.available() > 0)
  {
     char c = Serial.read();  //gets one character from serial buffer
     mode += c; //stores the character in a string variable
  }
   //clearLCDLine(line[0]);
  LCD_print(mode, line[0]);
}

void getHost()
{
  host = "";

  while (!Serial.available()){
    Serial.println("Host");
  }
  while (Serial.available() > 0)
  {
     char c = Serial.read();  //gets one byte from serial buffer
     host += c; //stores the character in a string variable
  }
   //clearLCDLine(line[1]);
  LCD_print(host, line[1]);
}

void getIP()
{
  rcv_ip = "";

  while (!Serial.available()){
    Serial.println("IP");
  }
  while (Serial.available() > 0)
  {
     char c = Serial.read();  //gets one byte from serial buffer
     rcv_ip += c; //stores the character in a string variable
  }

  //SUBNET Mask
  String subnet = getSub();
  char sub_char[16] = {0};
  subnet.toCharArray(sub_char, 16);

  rcv_ip += '/';
  rcv_ip += toCidr(sub_char); //Append subnet mask in CIDR notation onto the IP address
   //clearLCDLine(line[2]);
  LCD_print(rcv_ip, line[2]);
}
```

```
String  getSub()
{
  rcv_subnet = "";

   while  (!Serial.available()){
      Serial.println("Sub");
  }
   while  (Serial.available() > 0)
  {
      char c = Serial.read();  //gets one byte from serial buffer
      rcv_subnet += c; //makes the string readString
  }
   return  rcv_subnet;
}

void  getGate()
{
  rcv_gateway = "";

   while  (!Serial.available()){
      Serial.println("Gate");
  }
   while  (Serial.available() > 0)
  {
      char c = Serial.read();  //gets one byte from serial buffer
      rcv_gateway += c; //makes the string readString
  }
   //clearLCDLine(line[3]);
   LCD_print(rcv_gateway,  line[3]);
}

/*******************************************Configure Mode
Functions*********************************/
void  setDHCP()
{
   command(0x01);
   delay(2);
   command(0x0C);
   LCD_print("Setting  DHCP...",  line[0]);
  rcv_ok = "";
   Serial.println("DHCP");
   while  (!Serial.available()){
  }
   while  (Serial.available() > 0)
  {
      char c = Serial.read();  //gets one byte from serial buffer
      rcv_ok += c; //makes the string readString
  }
  if (rcv_ok == "OK")
  {
    rcv_ok = "";
     Serial.println(tx_ip);
     while (!Serial.available());
     while (Serial.available() > 0)
    {
       char c = Serial.read();  //gets one byte from serial buffer
       rcv_ok += c; //makes the string readString
    }

    if (rcv_ok == "OK")
    {
       delay(2000);
        LCD_print("Success", line[3]);
      rcv_ok = "";
       delay(1000);
    }
  }
}

void  setStatic()
{
   command(0x01);
  delay(2);
   command(0x0C);
   LCD_print("Setting Static IP...", line[0]);
  rcv_ok = "";
   Serial.println("Static");
   while (!Serial.available()){
  }
  while (Serial.available() > 0)
  {
      char c = Serial.read();  //gets one byte from serial buffer
      rcv_ok += c; //makes the string readString
  }

  if (rcv_ok == "OK")
  {
    rcv_ok = "";
     Serial.println(tx_ip);
     while (!Serial.available());
     while (Serial.available() > 0)
    {
       char c = Serial.read();  //gets one byte from serial buffer
       rcv_ok += c; //makes the string readString
    }
    if (rcv_ok == "OK")
    {
      rcv_ok = "";
       Serial.println(tx_subnet);
       while (!Serial.available());
       while (Serial.available() > 0)
      {
         char c = Serial.read();  //gets one byte from serial buffer
         rcv_ok += c; //makes the string readString
      }
      if (rcv_ok == "OK")
      {
        delay(2000);
         LCD_print("Success", line[3]);
        delay(1000);
      }
    }
  }
}

/*-----------------------------Setting Network Mode Cursor Function--------------------
void mode_updown(void)
{
  //UP
   command(line[obj]);
  if(digitalRead(up) == HIGH)
  {
     while(digitalRead(up) == HIGH){

    }

    if (obj < 2)
    {
      obj++;
       command(line[obj]);
    }
    else if (obj >= 2)
    {
      obj = 1;
       command(line[obj]);
    }

  }

  //DOWN
  if(digitalRead(down) == HIGH)
  {
     while(digitalRead(down) == HIGH){

    }
    if (obj > 1)
    {
       obj--;
        command(line[obj]);
    }
    else if (obj <= 1)
    {
       obj = 2;
        command(line[obj]);
    }
  }

}

/*-----------------------------Setting  Static  IP  Cursor  Functions---------------------*/
void  left_n_right(void)
{
   if(digitalRead(right)  ==  HIGH)
  {
     while(digitalRead(right)  ==  HIGH)
      {
         //Do nothing
      }

     if(count < 14)
      {
         command(0x14);
         count++;
      }
     else if (count >= 14)
      {
          command(line[1]);
         count = 0;
      }
  }


   if(digitalRead(left)  ==  HIGH)
  {
     while(digitalRead(left)  ==  HIGH)
    {
       //Do nothing
    }

     if(count > 0)
    {
       command(0x10);
       count--;
    }
```

```
     else if (count <= 0)
    {
       count = 14;
        command(line[1]+count);
    }
  }
}



void up_n_down(String &ip)
{
      //UP-----------------------------------------------------
   if(digitalRead(up) == HIGH)
  {
     while(digitalRead(up) == HIGH){
    }
       if (count == 0 || count == 1 || count == 2 || count == 4 || count == 5 || count ==
||
       count == 8 || count == 9 || count == 10 || count == 12 || count == 13 || count ==
      {
         //Add one (ASCII addition)
         ip[count]++;
         if(ip[count] > '9')
             ip[count] = '0';

         //Print to LCD and Move cursor to home
         LCD_print(ip, line[1]);

          //Return cursor to its original position
         for(int i=0; i < count; i++)
             command(0x14);
      }
  }
      //DOWN-----------------------------------------------------
   if(digitalRead(down) == HIGH)
  {
     while(digitalRead(down) == HIGH){
    }
       if (count == 0 || count == 1 || count == 2 || count == 4 || count == 5 || count ==
||
       count == 8 || count == 9 || count == 10 || count == 12 || count == 13 || count ==
      {
          //Subtract one (ASCII subtraction)
         ip[count]--;
         if(ip[count] < '0')
             ip[count] = '9';

         //Print to LCD and Move cursor to home
         LCD_print(ip, line[1]);
```

```
          //Return cursor to its original position
         for(int i=0; i < count; i++)
              command(0x14);
      }
  }
}

static  unsigned  short  toCidr(char*  ipAddress)
{
     unsigned short netmask_cidr;
     int ipbytes[4];

     netmask_cidr=0;
      sscanf(ipAddress,  "%d.%d.%d.%d",  &ipbytes[0],  &ipbytes[1],  &ipbytes[2],  &ipbytes[3]

     for (int i=0; i<4; i++)
    {
         switch(ipbytes[i])
        {
             case 0x80:
                  netmask_cidr+=1;
                 break;

             case 0xC0:
                  netmask_cidr+=2;
                 break;

             case 0xE0:
                  netmask_cidr+=3;
                 break;

             case 0xF0:
                  netmask_cidr+=4;
                 break;

             case 0xF8:
                  netmask_cidr+=5;
                 break;

             case 0xFC:
                  netmask_cidr+=6;
                 break;

             case 0xFE:
                  netmask_cidr+=7;
                 break;

             case 0xFF:
                  netmask_cidr+=8;
                 break;

             default:
                  return netmask_cidr;
                 break;
         }
    }

     return netmask_cidr;
}
```