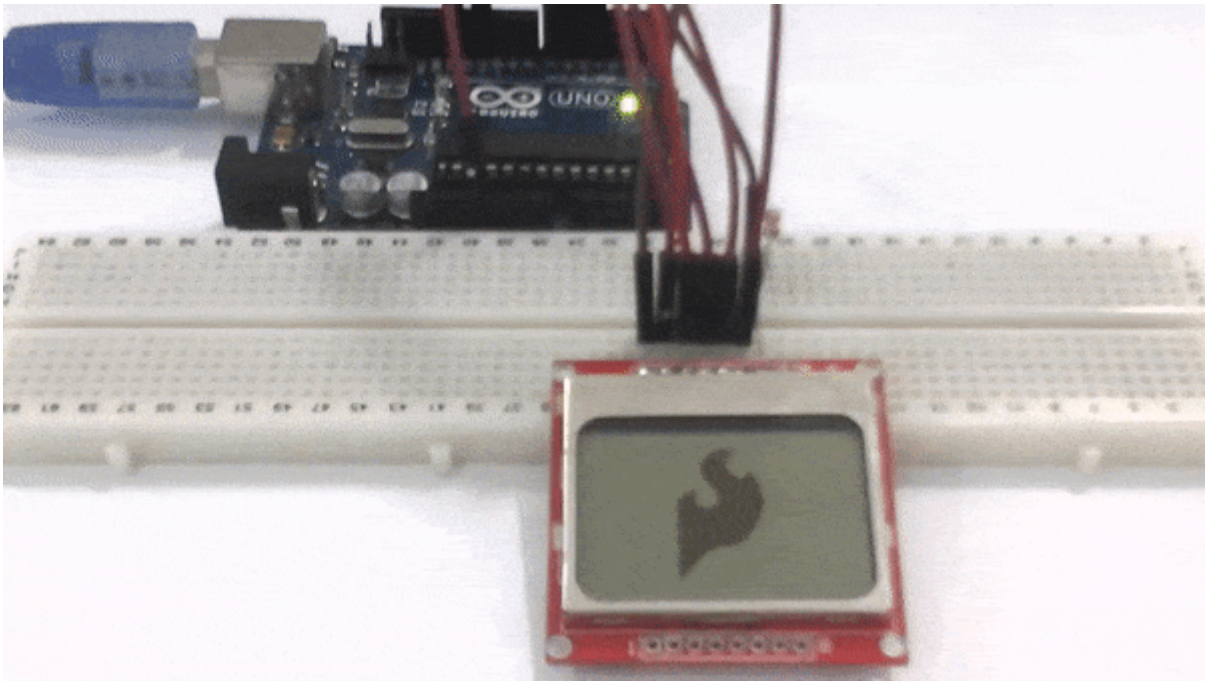
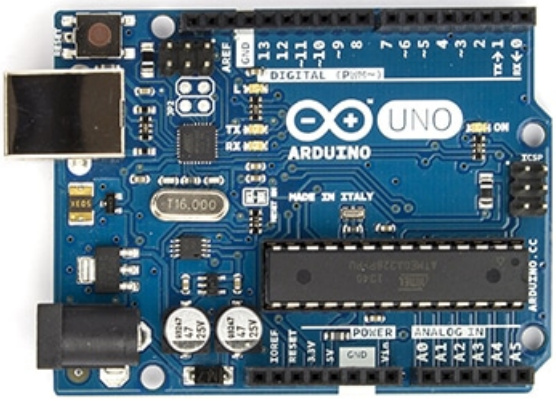
استخدام شاشة Nokia 5110 مع الاردوينو

في هذا المشروع سنتعلم كيفية استخدام شاشة Nokia 5110 مع الأردوينو لعرض صور ثابتة ومتحركة، وأيضا عرض الكلمات التي يتم ارسالها عن طريق الحاسوب.

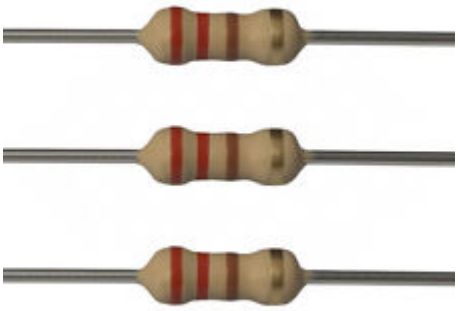


المكونات المطلوبة

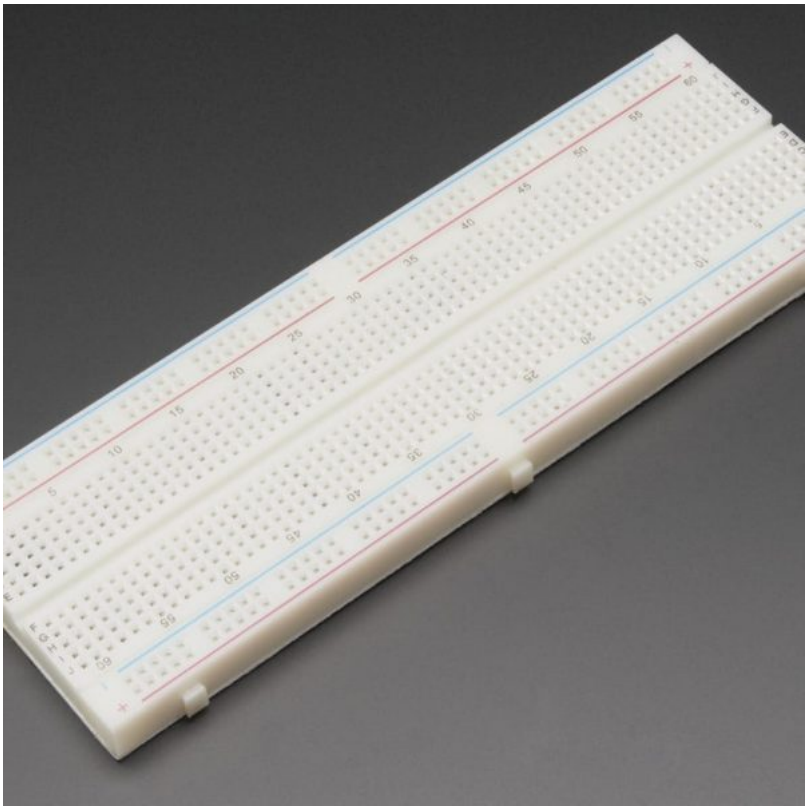


Arduino Uno

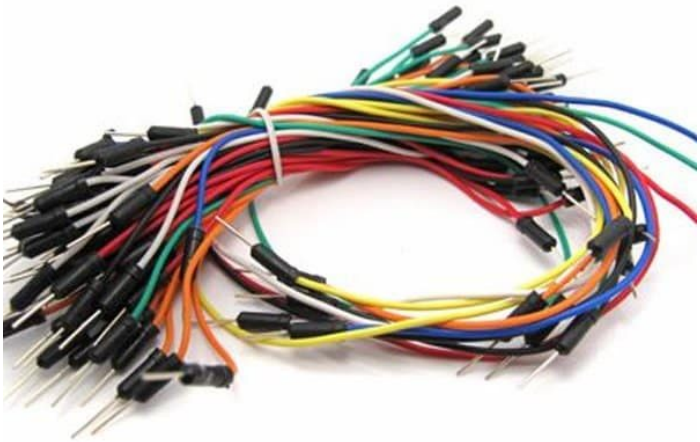
Nokia Screen 5110



Ohm Resistor 220



Breadboard



Wires

Nokia Screen 5110

تستخدم هذه الشاشة في العديد من التطبيقات، حيث كانت تستخدم في الهواتف النقالة . ويتم التحكم بها من خلال الإتصال عبر بروتوكول SPI، فهو وسيلة للربط بين المتحكمات والأجهزة الطرفية.

كما يمكننها التحكم بها على مستوى الـ Pixels، اي انها تعطي مرونة عالية للكتابة أو الرسم عليها.

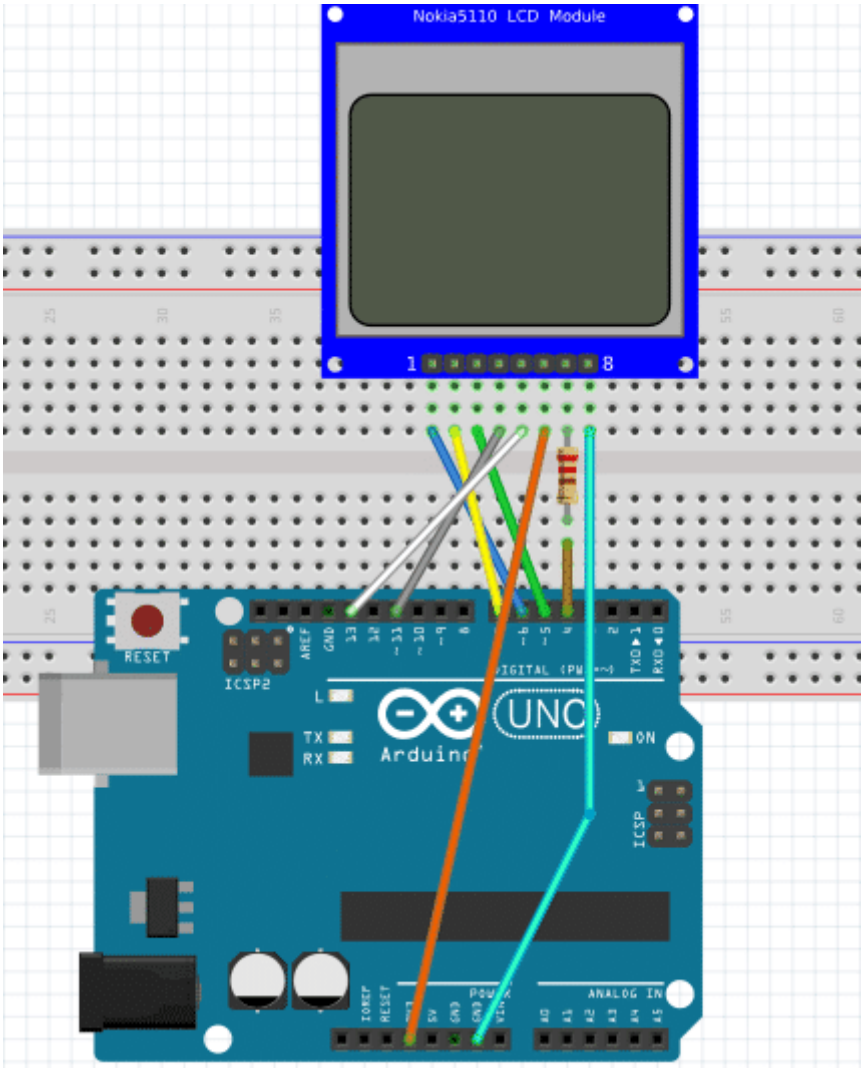
توصيلها مع الأردوينو :

هذة الشاشة تستخدم برتوكول SPI، ببساطة هو عبارة عن وسيلة لربط الاردوينو باجهزة اخرى مثل الشاشة في حالتنا. فيمكننا من ارسال البيانات اليها والتحكم فى تشغيلها من خلال هذا البروتوكول. لذلك لابد من توصيلها على منافذ الأردوينو المخصصة لذلك البروتوكول.

توصيل الدارة

نقوم بتوصيل الشاشة بالاردوينو باستخدام برتوكول الـ SPI ، ثم نقوم بإرسال الاوامر والبيانات اللازمة لعرض الصور بالشكل الذي نريده عن طريق الاردوينو.

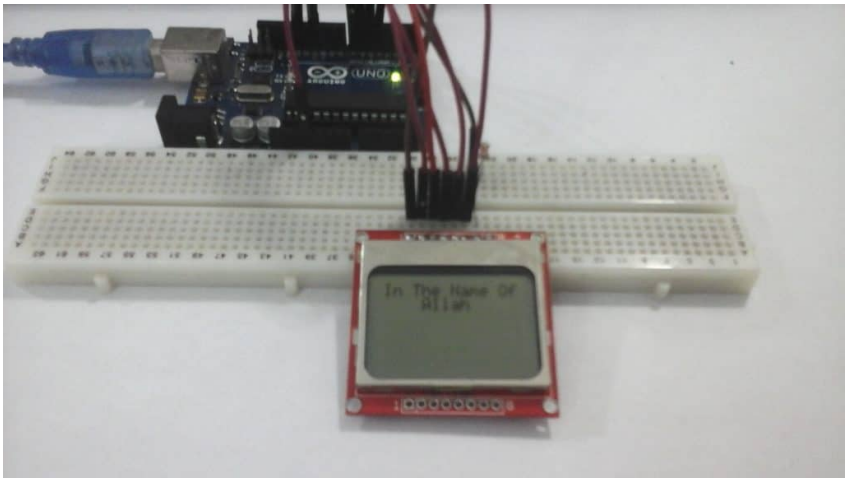
قم بتوصيل الدارة كما هو موضح بالصورة التالية :



يعمل هذا البروتوكول على ارجل محددة فى الاردوينو لذلك لا يمكننا تغيير المنافذ المستخدمة لذلك تم توصيل الشاشة مع الأردوينو كما هو موضح بالجدول :

الطرف (بداية من اليسار)	التوصيل
1	Arduino Pin 6
2	Arduino Pin 7
3	Arduino Pin 5
4	Arduino Pin 11
5	Arduino Pin 13
6	3.3v
7	Arduino Pin 4
8	GND

سنقوم بعمل برنامج يقوم في البداية بتشغيل الصور الثابتة والمتحركة، ثم يتوقف على صورة معينة منتظرا ادخال المستخدم لرسالة ما من خلال الشاشة التسلسلية (Serial Monitor)، ليقوم بعرضها على الشاشة.



البرمجة

في البداية، نقوم بإدراج المكتبة الخاصة بالـ SPI حتى نتمكن من استخدامه :

```
#include <SPI.h>
```

ثم نقوم بتسمية بعض الثوابت المستخدمة في إرسال الأوامر إلى الشاشة :

```
#define LCD_COMMAND  0
#define LCD_DATA     1
#define LCD_WIDTH   84
#define LCD_HEIGHT  48
#define WHITE       0
#define BLACK       1
```

بعد ذلك نقوم بتمسية منافذ الأردوينو المستخدمة في توصيل الشاشة :

```
const int scePin = 7;   // SCE - Chip select, pin 3 on LCD.
const int rstPin = 6;   // RST - Reset, pin 4 on LCD.
const int dcPin = 5;    // DC - Data/Command, pin 5 on LCD.
const int sdinPin = 11;  // DN(MOSI) - Serial data, pin 6 on LCD.
const int sclkPin = 13;  // SCLK - Serial clock, pin 7 on LCD.
const int blPin = 9;    // LED - Backlight LED, pin 8 on LCD.
```

بعد ذلك قمنا بعمل مصفوفة تحتوي على قيم الحروف والرموز والأرقام الممثلة بصيغة الـ ASCII ، وهي طريقة قياسية لتمثيل الحروف والأقام وبعض الرموز. فعند عرض الحرف a فإن القيمة المكافئة له تساوي 0x16 وهكذا.

```
static const byte ASCII[][5] = {
   {0x00, 0x00, 0x00, 0x00, 0x00} // 0x20
  ,{0x00, 0x00, 0x5f, 0x00, 0x00} // 0x21 !
  ,{0x00, 0x07, 0x00, 0x07, 0x00} // 0x22 "
  ,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 0x23 #
  ,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 0x24 $
  ,{0x23, 0x13, 0x08, 0x64, 0x62} // 0x25 %
  ,{0x36, 0x49, 0x55, 0x22, 0x50} // 0x26 &
  ,{0x00, 0x05, 0x03, 0x00, 0x00} // 0x27 '
```

```
  ,{0x00, 0x1c, 0x22, 0x41, 0x00} // 0x28 (
  ,{0x00, 0x41, 0x22, 0x1c, 0x00} // 0x29 )
  ,{0x14, 0x08, 0x3e, 0x08, 0x14} // 0x2a *
  ,{0x08, 0x08, 0x3e, 0x08, 0x08} // 0x2b +
  ,{0x00, 0x50, 0x30, 0x00, 0x00} // 0x2c ,
  ,{0x08, 0x08, 0x08, 0x08, 0x08} // 0x2d -
  ,{0x00, 0x60, 0x60, 0x00, 0x00} // 0x2e .
  ,{0x20, 0x10, 0x08, 0x04, 0x02} // 0x2f /
  ,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 0x30 0
  ,{0x00, 0x42, 0x7f, 0x40, 0x00} // 0x31 1
  ,{0x42, 0x61, 0x51, 0x49, 0x46} // 0x32 2
  ,{0x21, 0x41, 0x45, 0x4b, 0x31} // 0x33 3
  ,{0x18, 0x14, 0x12, 0x7f, 0x10} // 0x34 4
  ,{0x27, 0x45, 0x45, 0x45, 0x39} // 0x35 5
  ,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 0x36 6
  ,{0x01, 0x71, 0x09, 0x05, 0x03} // 0x37 7
  ,{0x36, 0x49, 0x49, 0x49, 0x36} // 0x38 8
  ,{0x06, 0x49, 0x49, 0x29, 0x1e} // 0x39 9
  ,{0x00, 0x36, 0x36, 0x00, 0x00} // 0x3a :
  ,{0x00, 0x56, 0x36, 0x00, 0x00} // 0x3b ;
  ,{0x08, 0x14, 0x22, 0x41, 0x00} // 0x3c <
  ,{0x02, 0x01, 0x51, 0x09, 0x06} // 0x3f ?
  ,{0x32, 0x49, 0x79, 0x41, 0x3e} // 0x40 @
  ,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 0x41 A
  ,{0x7f, 0x49, 0x49, 0x49, 0x36} // 0x42 B
  ,{0x3e, 0x41, 0x41, 0x41, 0x22} // 0x43 C
  ,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 0x44 D
  ,{0x7f, 0x49, 0x49, 0x49, 0x41} // 0x45 E
  ,{0x7f, 0x09, 0x09, 0x09, 0x01} // 0x46 F
  ,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 0x47 G
  ,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 0x48 H
  ,{0x00, 0x41, 0x7f, 0x41, 0x00} // 0x49 I
  ,{0x20, 0x40, 0x41, 0x3f, 0x01} // 0x4a J
  ,{0x7f, 0x08, 0x14, 0x22, 0x41} // 0x4b K
  ,{0x7f, 0x40, 0x40, 0x40, 0x40} // 0x4c L
  ,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 0x4d M
  ,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 0x4e N
  ,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 0x4f O
  ,{0x7f, 0x09, 0x09, 0x09, 0x06} // 0x50 P
  ,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 0x51 Q
  ,{0x7f, 0x09, 0x19, 0x29, 0x46} // 0x52 R
  ,{0x46, 0x49, 0x49, 0x49, 0x31} // 0x53 S
  ,{0x01, 0x01, 0x7f, 0x01, 0x01} // 0x54 T
  ,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 0x55 U
  ,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 0x56 V
  ,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 0x57 W
  ,{0x63, 0x14, 0x08, 0x14, 0x63} // 0x58 X
  ,{0x07, 0x08, 0x70, 0x08, 0x07} // 0x59 Y
  ,{0x61, 0x51, 0x49, 0x45, 0x43} // 0x5a Z
  ,{0x00, 0x7f, 0x41, 0x41, 0x00} // 0x5b [
  ,{0x02, 0x04, 0x08, 0x10, 0x20} // 0x5c \
  ,{0x00, 0x41, 0x41, 0x7f, 0x00} // 0x5d ]
  ,{0x00, 0x41, 0x41, 0x7f, 0x00} // 0x5d ]
```

```
  ,{0x04, 0x02, 0x01, 0x02, 0x04} // 0x5e ^
  ,{0x40, 0x40, 0x40, 0x40, 0x40} // 0x5f _
  ,{0x00, 0x01, 0x02, 0x04, 0x00} // 0x60 `
  ,{0x20, 0x54, 0x54, 0x54, 0x78} // 0x61 a
  ,{0x7f, 0x48, 0x44, 0x44, 0x38} // 0x62 b
  ,{0x38, 0x44, 0x44, 0x44, 0x20} // 0x63 c
  ,{0x38, 0x44, 0x44, 0x48, 0x7f} // 0x64 d
  ,{0x38, 0x54, 0x54, 0x54, 0x18} // 0x65 e
  ,{0x08, 0x7e, 0x09, 0x01, 0x02} // 0x66 f
  ,{0x0c, 0x52, 0x52, 0x52, 0x3e} // 0x67 g
  ,{0x7f, 0x08, 0x04, 0x04, 0x78} // 0x68 h
  ,{0x00, 0x44, 0x7d, 0x40, 0x00} // 0x69 i
  ,{0x20, 0x40, 0x44, 0x3d, 0x00} // 0x6a j
  ,{0x7f, 0x10, 0x28, 0x44, 0x00} // 0x6b k
  ,{0x00, 0x41, 0x7f, 0x40, 0x00} // 0x6c l
  ,{0x7c, 0x04, 0x18, 0x04, 0x78} // 0x6d m
  ,{0x7c, 0x08, 0x04, 0x04, 0x78} // 0x6e n
  ,{0x38, 0x44, 0x44, 0x44, 0x38} // 0x6f o
  ,{0x7c, 0x14, 0x14, 0x14, 0x08} // 0x70 p
  ,{0x08, 0x14, 0x14, 0x18, 0x7c} // 0x71 q
  ,{0x7c, 0x08, 0x04, 0x04, 0x08} // 0x72 r
  ,{0x48, 0x54, 0x54, 0x54, 0x20} // 0x73 s
  ,{0x04, 0x3f, 0x44, 0x40, 0x20} // 0x74 t
  ,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 0x75 u
  ,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 0x76 v
  ,{0x3c, 0x40, 0x30, 0x40, 0x3c} // 0x77 w
  ,{0x44, 0x28, 0x10, 0x28, 0x44} // 0x78 x
  ,{0x0c, 0x50, 0x50, 0x50, 0x3c} // 0x79 y
  ,{0x44, 0x64, 0x54, 0x4c, 0x44} // 0x7a z
  ,{0x00, 0x08, 0x36, 0x41, 0x00} // 0x7b {
  ,{0x00, 0x00, 0x7f, 0x00, 0x00} // 0x7c |
  ,{0x00, 0x41, 0x36, 0x08, 0x00} // 0x7d }
  ,{0x10, 0x08, 0x08, 0x10, 0x08} // 0x7e ~
  ,{0x78, 0x46, 0x41, 0x46, 0x78} // 0x7f DEL
};
```

بعد ذلك نقوم بالإعلان عن المصفوفة displayMap ، وهي تمثل الـ Pixels الموجودة في الشاشة. فعند وضع قيمة 0 في إحدى قيم المصفوفة، نقوم بذلك بإطفاء الـ Pixel المكافئة له على الشاشة. وعند وضع قيمة 255 نقوم بإضاءة الـ Pixel المكافئة :

```
byte displayMap[LCD_WIDTH * LCD_HEIGHT / 8] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0,
  0xF0, 0xF8, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFE, 0x1E, 0x0E, 0x02, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03,
  0x0F, 0x1F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFC, 0xF8,
  0xF8, 0xF0, 0xF8, 0xFE, 0xFE, 0xFC, 0xF8, 0xE0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0xF8, 0xFC, 0xFE, 0xFE, 0xFF, 0xFF, 0xF3, 0xE0, 0xE0, 0xC0,
  0xC0, 0xC0, 0xE0, 0xE0, 0xF1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x3E, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
  0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x1F, 0x07, 0x01, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x1F,
  0x0F, 0x0F, 0x0F, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x03, 0x03,
  0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

نقوم بإنشاء مصفوفة xkcdSandwich التي يقوم الأردوينو بإرسالها إلى الشاشة فتقوم برسم صورة معينة سنراها عندما نكمل كتابة الكود ويتم رفعه إلى الأردوينو :

```
char xkcdSandwich[504] = {
0xFF, 0x8D, 0x9F, 0x13, 0x13, 0xF3, 0x01, 0x01, 0xF9, 0xF9, 0x01, 0x81, 0xF9, 0xF9,
0x01, 0xF1,
0xF9, 0x09, 0x09, 0xFF, 0xFF, 0xF1, 0xF9, 0x09, 0x09, 0xF9, 0xF1, 0x01, 0x01, 0x01,
0x01, 0x01,
0xF9, 0xF9, 0x09, 0xF9, 0x09, 0xF9, 0xF1, 0x01, 0xC1, 0xE9, 0x29, 0x29, 0xF9, 0xF1,
0x01, 0xFF,
0xFF, 0x71, 0xD9, 0x01, 0x01, 0xF1, 0xF9, 0x29, 0x29, 0xB9, 0xB1, 0x01, 0x01, 0x01,
0xF1, 0xF1,
0x11, 0xF1, 0xF1, 0xF1, 0xE1, 0x01, 0xE1, 0xF1, 0x51, 0x51, 0x71, 0x61, 0x01, 0x01,
0xC1, 0xF1,
0x31, 0x31, 0xF1, 0xFF, 0xFF, 0x00, 0x01, 0x01, 0x01, 0x01, 0x60, 0xE0, 0xA0, 0x01,
0x01, 0x81,
0xE1, 0x61, 0x60, 0xC0, 0x01, 0xE1, 0xE1, 0x21, 0x21, 0xE0, 0xC1, 0x01, 0xC1, 0xE1,
0x20, 0x20,
0xFC, 0xFC, 0xE0, 0xE0, 0xC1, 0xE1, 0xE0, 0xC1, 0xE0, 0xE1, 0x01, 0xFC, 0xFC, 0x21,
0x21, 0xE1,
0xC1, 0xE5, 0xE4, 0x01, 0xC1, 0xE0, 0x20, 0x21, 0x20, 0x00, 0x01, 0xFD, 0xFD, 0x21,
0x20, 0xE0,
```

```
0x00, 0x00, 0x01, 0x01, 0xC0, 0x61, 0x31, 0x31, 0x21, 0x20, 0xC0, 0x81, 0x01, 0x01,
0x01, 0x00,
0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x01,
0x03, 0x02,
0x03, 0x01, 0x00, 0x01, 0x03, 0xF2, 0x1A, 0x0B, 0x08, 0x0B, 0x1B, 0x10, 0x60, 0xE3,
0x03, 0x00,
0x01, 0x03, 0x02, 0x02, 0x03, 0x03, 0x00, 0x03, 0x03, 0x00, 0x00, 0x03, 0x03, 0x00,
0x00, 0x03,
0x03, 0x00, 0x00, 0x03, 0x03, 0x03, 0x03, 0x00, 0x01, 0x03, 0x02, 0x02, 0x03, 0x01,
0x00, 0x03,
0x03, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x3E, 0x63, 0x80, 0x80, 0x80, 0x80, 0x60,
0x3F, 0x07,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00,
0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xFE, 0x01, 0x01, 0x01, 0x02, 0x03, 0x3E, 0xE8, 0xF8, 0xF0,
0xD0, 0x90,
0x18, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0,
0x38, 0xFF,
0x0C, 0x38, 0xE0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xF0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x33,
0x5F, 0x8F, 0x84, 0x05, 0x07, 0x06, 0x0C, 0x0E, 0x0E, 0x0C, 0x14, 0x34, 0x68, 0x88,
0xD8, 0x70,
0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x10, 0x10, 0x10, 0xF0, 0xE0, 0x00, 0xF0, 0xF0,
0x00, 0x80,
0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x80, 0x80, 0x00, 0x80, 0x00, 0x00,
0x20, 0x38,
0x0E, 0x01, 0xC0, 0x3F, 0xE0, 0x00, 0x00, 0x03, 0x0E, 0x08, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
0x00, 0x00, 0x00, 0xFF, 0xFF, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xB6, 0xED,
0xC0, 0xC0,
0xC0, 0xE0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA1, 0xA1, 0xA1, 0xA1, 0xA1, 0xA1, 0xA1, 0xE1,
0xE1, 0xC1,
0xEF, 0xBB, 0x83, 0x86, 0x88, 0xB0, 0x80, 0x80, 0x80, 0x8F, 0x90, 0x90, 0x90, 0x9F,
0x8F, 0x80,
0x9F, 0x9F, 0x87, 0x8D, 0x98, 0x80, 0x8C, 0x9E, 0x92, 0x92, 0x9F, 0xC0, 0xC7, 0xFF,
0xB8, 0x8F,
0x80, 0x90, 0x90, 0xC0, 0xF0, 0x8E, 0x81, 0x80, 0x81, 0x8F, 0xB8, 0xE0, 0x80, 0x80,
0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xFF,
};
```

في الدالة ()setup، نقوم بوضع الإعدادات اللازمة للمشروع مثل تشغيل الشاشة وضبط إعداداتها اللازمة مثل الوضوح وإظهار بعض الصور المتحركة.

وايضا نقوم بتشغيل الشاشة التسلسلية (Serial Monitor) التي سنستخدمها في إرسال الأحرف والكلمات لعرضها على الشاشة لاحقا.

حيث سنقوم بالككتابه على الشاشة بعد أن تنتهي من عرض الصور المتحركة.

```
void setup()
{
  Serial.begin(9600);

  lcdBegin(); // This will setup our pins, and initialize the LCD
  setContrast(55); // Pretty good value, play around with it

  updateDisplay(); // with displayMap untouched, SFE logo
  delay(2000);

  lcdFunTime(); // Runs a 30-second demo of graphics functions

  // Wait for serial to come in, then clear display and go to echo
  while (!Serial.available())
    ;
  clearDisplay(WHITE);
  updateDisplay();
}
```

في الدله ()loop، تقوم بإنتظار المستخدم إدخال الحروف أو الكلمات إلى الشاشة التسلسلية (Serial Monitor) ، ليقوم الأردوينو بإرسالها إلى الشاشة ليتم كتابتها. نقوم بعمل إختبار على ماتم ادخاله من قبل المستخدم. فمثلا إذا ادخل المستخدم الرمز ~ فإنه عبارة عن أمر لمسح الشاشة.

```
void loop()
{
  static int cursorX = 0;
  static int cursorY = 0;

  if (Serial.available())
  {
    char c = Serial.read();

    switch (c)
    {
    case '\n': // New line
      cursorY += 8;
      break;
    case '\r': // Return feed
      cursorX = 0;
      break;
    case '~': // Use ~ to clear the screen.
      clearDisplay(WHITE);
      updateDisplay();
      cursorX = 0; // reset the cursor
      cursorY = 0;
      break;
    default:
      setChar(c, cursorX, cursorY, BLACK);
      updateDisplay();
      cursorX += 6; // Increment cursor
      break;
    }
```

```
    // Manage cursor
    if (cursorX >= (LCD_WIDTH - 4))
    { // If the next char will be off screen...
      cursorX = 0; // ... reset x to 0...
      cursorY += 8; // ...and increment to next line.
      if (cursorY >= (LCD_HEIGHT - 7))
      { // If the next line takes us off screen...
        cursorY = 0; // ...go back to the top.
      }
    }
  }
}
```

باقي الدوال المستخدمة متقدمة بعض الشئ، يكفي فقط أن تعرف أنها تقوم بإرسال البيانات إلى الشاشة بشكل معين لتتمكن من عرض الصور أو الكلمات. وهي الدوال المستخدمة داخل الدالتين ()setup، و ()loop .

إذا كنت تريد تغيير ما يظهر على الشاشة، فلا حاجة إلى تغيير هذه الدوال، فقط قم بعمل التغير الذي تريده داخل الداليتين ()loop، و ()setup.

```
void lcdFunTime()
{
  clearDisplay(WHITE);
  randomSeed(analogRead(A0));
  const int pixelCount = 100;
  for (int i=0; i<pixelCount; i++)
  {
    setPixel(random(0, LCD_WIDTH), random(0, LCD_HEIGHT));
    updateDisplay();
    delay(10);
  }
  setStr("full of stars", 0, LCD_HEIGHT-8, BLACK);
  updateDisplay();
  delay(1000);
  for (int i=0; i<5; i++)
  {
    invertDisplay();
    delay(200);
    invertDisplay();
    delay(200);
  }
  delay(2000);
  clearDisplay(WHITE);
  int x0 = LCD_WIDTH/2;
  int y0 = LCD_HEIGHT/2;
  for (float i=0; i<2*PI; i+=PI/8)
  {
    const int lineLength = 24;
    int x1 = x0 + lineLength * sin(i);
    int y1 = y0 + lineLength * cos(i);
    setLine(x0, y0, x1, y1, BLACK);
    updateDisplay();
    delay(100);
```

```
  }
  for (int j=0; j<2; j++)
  {
    for (int i=255; i>=0; i-=5)
    {
      analogWrite(blPin, i);
      delay(20);
    }
    for (int i=0; i<256; i+=5)
    {
      analogWrite(blPin, i);
      delay(20);
    }
  }
  clearDisplay(WHITE);
  for (int x=0; x<LCD_WIDTH; x+=8)
  {
    setRect(0, 0, x, LCD_HEIGHT, 1, BLACK);
    updateDisplay();
    delay(10);
  }
  for (int x=0; x<LCD_WIDTH; x+=8)
  {
    setRect(0, 0, x, LCD_HEIGHT, 1, WHITE);
    updateDisplay();
    delay(10);
  }
  for (int x=0; x<12; x++)
  {
    setRect(0, 0, x, LCD_HEIGHT, 1, 1);
    setRect(11, 0, x+12, LCD_HEIGHT, 1, BLACK);
    setRect(23, 0, x+24, LCD_HEIGHT, 1, BLACK);
    setRect(35, 0, x+36, LCD_HEIGHT, 1, BLACK);
    setRect(47, 0, x+48, LCD_HEIGHT, 1, BLACK);
    setRect(59, 0, x+60, LCD_HEIGHT, 1, BLACK);
    setRect(71, 0, x+72, LCD_HEIGHT, 1, BLACK);
    updateDisplay();
    delay(10);
  }
  setRect(25, 10, 45, 30, 0, WHITE);
  setRect(35, 20, 55, 40, 0, WHITE);
  setLine(25, 10, 35, 20, WHITE);
  setLine(45, 30, 55, 40, WHITE);
  setLine(25, 30, 35, 40, WHITE);
  setLine(45, 10, 55, 20, WHITE);
  updateDisplay();
  delay(2000);
  clearDisplay(WHITE);
  for (int i=0; i<20; i++)
  {
    int x = random(0, LCD_WIDTH);
    int y = random(0, LCD_HEIGHT);
    setCircle(x, y, i, BLACK, 1);
```

```
    updateDisplay();
    delay(100);
  }
  delay(2000);
  setStr("Modern Art", 0, 10, WHITE);
  updateDisplay();
  delay(2000);
  setBitmap(xkcdSandwich);
  updateDisplay();
}

void setPixel(int x, int y)
{
  setPixel(x, y, BLACK);
}

void clearPixel(int x, int y)
{
  setPixel(x, y, WHITE);
}

void setPixel(int x, int y, boolean bw)
{
  if ((x >= 0) && (x < LCD_WIDTH) && (y >= 0) && (y < LCD_HEIGHT))
  {
    byte shift = y % 8;

    if (bw)
      displayMap[x + (y/8)*LCD_WIDTH] |= 1<<shift;
    else
      displayMap[x + (y/8)*LCD_WIDTH] &= ~(1<<shift);
  }
}

void setLine(int x0, int y0, int x1, int y1, boolean bw)
{
  int dy = y1 - y0; // Difference between y0 and y1
  int dx = x1 - x0; // Difference between x0 and x1
  int stepx, stepy;
  if (dy < 0)
  {
    dy = -dy;
    stepy = -1;
  }
  else
    stepy = 1;

  if (dx < 0)
  {
    dx = -dx;
    stepx = -1;
  }
  else
```

```
    stepx = 1;

  dy <<= 1;
  dx <<= 1;
  setPixel(x0, y0, bw);
  if (dx > dy)
  {
    int fraction = dy - (dx >> 1);
    while (x0 != x1)
    {
      if (fraction >= 0)
      {
        y0 += stepy;
        fraction -= dx;
      }
      x0 += stepx;
      fraction += dy;
      setPixel(x0, y0, bw);
    }
  }
  else
  {
    int fraction = dx - (dy >> 1);
    while (y0 != y1)
    {
      if (fraction >= 0)
      {
        x0 += stepx;
        fraction -= dy;
      }
      y0 += stepy;
      fraction += dx;
      setPixel(x0, y0, bw);
    }
  }
}

void setRect(int x0, int y0, int x1, int y1, boolean fill, boolean bw)
{
  if (fill == 1)
  {
    int xDiff;

    if(x0 > x1)
      xDiff = x0 - x1;
    else
      xDiff = x1 - x0;

    while(xDiff > 0)
    {
      setLine(x0, y0, x0, y1, bw);

      if(x0 > x1)
```

```
        x0--;
      else
        x0++;

      xDiff--;
    }
  }
  else
  {
    setLine(x0, y0, x1, y0, bw);
    setLine(x0, y1, x1, y1, bw);
    setLine(x0, y0, x0, y1, bw);
    setLine(x1, y0, x1, y1, bw);
  }
}

void setCircle (int x0, int y0, int radius, boolean bw, int lineThickness)
{
  for(int r = 0; r < lineThickness; r++)
  {
    int f = 1 - radius;
    int ddF_x = 0;
    int ddF_y = -2 * radius;
    int x = 0;
    int y = radius;

    setPixel(x0, y0 + radius, bw);
    setPixel(x0, y0 - radius, bw);
    setPixel(x0 + radius, y0, bw);
    setPixel(x0 - radius, y0, bw);

    while(x < y)
    {
      if(f >= 0)
      {
        y--;
        ddF_y += 2;
        f += ddF_y;
      }
      x++;
      ddF_x += 2;
      f += ddF_x + 1;

      setPixel(x0 + x, y0 + y, bw);
      setPixel(x0 - x, y0 + y, bw);
      setPixel(x0 + x, y0 - y, bw);
      setPixel(x0 - x, y0 - y, bw);
      setPixel(x0 + y, y0 + x, bw);
      setPixel(x0 - y, y0 + x, bw);
      setPixel(x0 + y, y0 - x, bw);
      setPixel(x0 - y, y0 - x, bw);
    }
    radius--;
```

```
  }
}

void setChar(char character, int x, int y, boolean bw)
{
  byte column;
  for (int i=0; i<5; i++)
  {
    column = ASCII[character - 0x20][i];
    for (int j=0; j<8; j++)
    {
      if (column & (0x01 << j))
        setPixel(x+i, y+j, bw);
      else
        setPixel(x+i, y+j, !bw);
    }
  }
}

void setStr(char * dString, int x, int y, boolean bw)
{
  while (*dString != 0x00)
  {
    setChar(*dString++, x, y, bw);
    x+=5;
    for (int i=y; i<y+8; i++)
    {
      setPixel(x, i, !bw);
    }
    x++;
    if (x > (LCD_WIDTH - 5))
    {
      x = 0;
      y += 8;
    }
  }
}

void setBitmap(char * bitArray)
{
  for (int i=0; i<(LCD_WIDTH * LCD_HEIGHT / 8); i++)
    displayMap[i] = bitArray[i];
}

void clearDisplay(boolean bw)
{
  for (int i=0; i<(LCD_WIDTH * LCD_HEIGHT / 8); i++)
  {
    if (bw)
      displayMap[i] = 0xFF;
    else
      displayMap[i] = 0;
  }
```

```
}

void gotoXY(int x, int y)
{
  LCDWrite(0, 0x80 | x);
  LCDWrite(0, 0x40 | y);
}

void updateDisplay()
{
  gotoXY(0, 0);
  for (int i=0; i < (LCD_WIDTH * LCD_HEIGHT / 8); i++)
  {
    LCDWrite(LCD_DATA, displayMap[i]);
  }
}

void setContrast(byte contrast)
{
  LCDWrite(LCD_COMMAND, 0x21);
  LCDWrite(LCD_COMMAND, 0x80 | contrast);
  LCDWrite(LCD_COMMAND, 0x20);
}

void invertDisplay()
{
  LCDWrite(LCD_COMMAND, 0x20);
  LCDWrite(LCD_COMMAND, 0x08 | 0x05);
  LCDWrite(LCD_COMMAND, 0x20);
  for (int i=0; i < (LCD_WIDTH * LCD_HEIGHT / 8); i++)
  {
    displayMap[i] = ~displayMap[i] & 0xFF;
  }
  updateDisplay();
}

void LCDWrite(byte data_or_command, byte data)
{
  digitalWrite(dcPin, data_or_command);
  digitalWrite(scePin, LOW);
  SPI.transfer(data); //shiftOut(sdinPin, sclkPin, MSBFIRST, data);
  digitalWrite(scePin, HIGH);
}

void lcdBegin(void)
{
  pinMode(scePin, OUTPUT);
  pinMode(rstPin, OUTPUT);
  pinMode(dcPin, OUTPUT);
  pinMode(sdinPin, OUTPUT);
  pinMode(sclkPin, OUTPUT);
  pinMode(blPin, OUTPUT);
  analogWrite(blPin, 255);
```

```
  SPI.begin();
  SPI.setDataMode(SPI_MODE0);
  SPI.setBitOrder(MSBFIRST);
  digitalWrite(rstPin, LOW);
  digitalWrite(rstPin, HIGH);

  LCDWrite(LCD_COMMAND, 0x21);
  LCDWrite(LCD_COMMAND, 0xB0);
  LCDWrite(LCD_COMMAND, 0x04);
  LCDWrite(LCD_COMMAND, 0x14);
  LCDWrite(LCD_COMMAND, 0x20);
  LCDWrite(LCD_COMMAND, 0x0C);
}
```