Timer using Nokia 3310/5110 LCD Display

https://github.com/moozzyk/ArduinoDigitalClockhttp://www.thaieasyelec.com/
http://blog.3d-logic.com/
http://www.youtube.com/watch?v=-60g9EO3W8o

CONNECTION: From LCD to Arduino Pins

pin 7	Serial clock out (SCLK)
pin 6	Serial data out (DIN)
pin 5	Data/Command select (D/C)
pin 4	LCD chip select (CS) or (CE)
pin 3	LCD reset (RST)
LCD_CMD	13 (LIGHT) Digital Pin 13 else control LCD backlight with potentiometer using PWM pins
VCC	3.3V (DO NOT PLUG IT TO 5V)
GRD	GROUND

HARDWARE REQUIRED

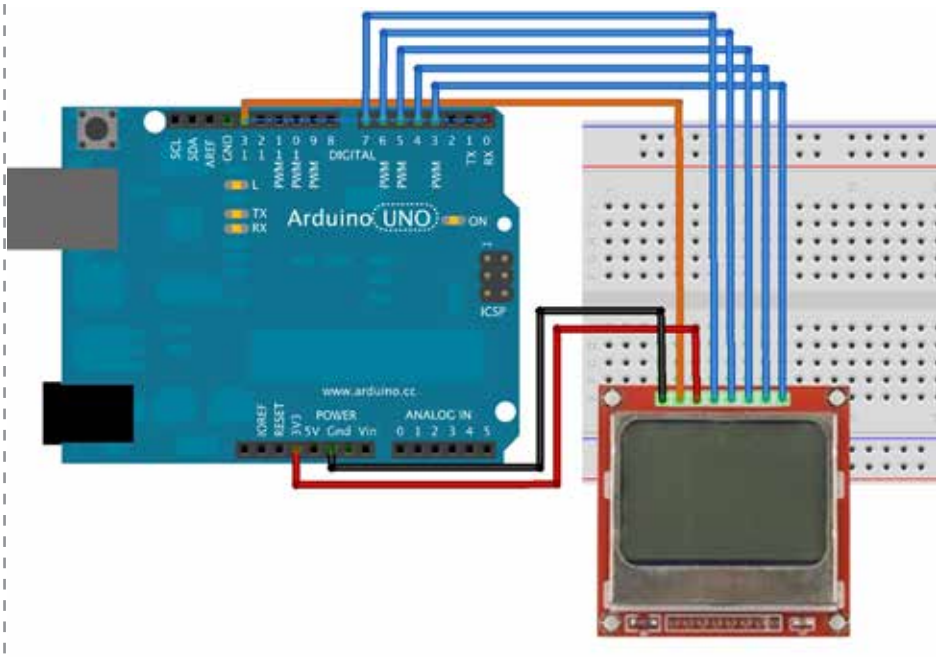
Arduino Board
LCD Display NOKIA type 3310 or 5110

CIRCUIT

SOME PROJECTS REQUIRE LIBRARIES TO BE INSTALLED
These displays are small, only about 1.5» diameter, but very readable due and comes with a backlight. This display is made of 84x48 individual pixels, so you can use it for graphics, text or bitmaps. These displays are inexpensive, easy to use, require only a few digital I/O pins and are fairly low power as well.

To drive the display, you will need 3 to 5 digital output pins (depending on whether you want to manually control the chip select and reset lines). Another pin can be used to control (via on/off or PWM) the backlight. To make things easy for you, we've written a nice graphics library that can print text, pixels, rectangles, circles and lines! The library is written for the Arduino but can easily be ported to your favorite microcontroller.

IMAGE



CODE

```
#define PIN_SCLK  7
#define PIN_SDIN  6
#define PIN_DC    5
#define PIN_SCE   4 // or CE
#define PIN_RESET 3
#define LCD_CMD   0

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48

static const byte Digits[][4][18] =
{
 {
   { 0xE0, 0xF0, 0xF8, 0xF4, 0xEE, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0x1F, 0xEE, 0xF4, 0xF8, 0xF0, 0xE0 },
   { 0x1F, 0x3F, 0x7F, 0x3F, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x1F, 0x3F, 0x7F, 0x3F, 0x1F },
   { 0xFC, 0xFE, 0xFF, 0xFE, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xFC, 0xFE, 0xFF, 0xFE, 0xFC },
   { 0x03, 0x07, 0x0F, 0x17, 0x3B, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x7C, 0x3B, 0x17, 0x0F, 0x07, 0x03 },
  },
  {
   { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xE0, 0xF0, 0xF8, 0xF0, 0xE0 },
   { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x1F, 0x3F, 0x7F, 0x3F, 0x1F },
   { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xFC, 0xFE, 0xFF, 0xFE, 0xFC },
   { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x03, 0x07, 0x0F, 0x07, 0x03 },
  },
  {
   { 0x00, 0x00, 0x00, 0x04, 0x0E, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0x1F, 0xEE, 0xF4, 0xF8, 0xF0, 0xE0 },
   { 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xDF, 0xBF, 0x7F, 0x3F, 0x1F },
   { 0xFC, 0xFE, 0xFF, 0xFE, 0xFD, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00 },
   { 0x03, 0x07, 0x0F, 0x17, 0x3B, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x7C, 0x38, 0x10, 0x00, 0x00, 0x00 },
  },
  {
   { 0x00, 0x00, 0x00, 0x04, 0x0E, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0x1F, 0xEE, 0xF4, 0xF8, 0xF0, 0xE0 },
   { 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xDF, 0xBF, 0x7F, 0x3F, 0x1F },
   { 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC },
   { 0x00, 0x00, 0x00, 0x10, 0x38, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x7C, 0x3B, 0x17, 0x0F, 0x07, 0x03 },
  },
  {
   { 0xE0, 0xF0, 0xF8, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xE0, 0xF0, 0xF8, 0xF0, 0xE0 },
   { 0x1F, 0x3F, 0x7F, 0xBF, 0xDF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xDF, 0xBF, 0x7F, 0x3F, 0x1F },
   { 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC },
   { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x03, 0x07, 0x0F, 0x07, 0x03 },
  },
  {
   { 0xE0, 0xF0, 0xF8, 0xF4, 0xEE, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0x1F, 0x0E, 0x04, 0x00, 0x00, 0x00 },
   { 0x1F, 0x3F, 0x7F, 0xBF, 0xDF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00 },
```

```
/***************************
*****************************
**************
This is an example sketch for our
Monochrome Nokia 5110 LCD
Displays

  Pick one up today in the adafruit
shop!
  ------> http://www.adafruit.com/
products/338

These displays use SPI to commu-
nicate, 4 or 5 pins are required to
interface

Adafruit invests time and re-
sources providing this open source
code,
please support Adafruit and open-
source hardware by purchasing
products from Adafruit!

Written by Limor Fried/Ladyada
for Adafruit Industries.
BSD license, check license.txt for
more information
All text above, and the splash
screen must be included in any
redistribution
*****************************
*****************************
*************/
```

```
    { 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC },
    { 0x00, 0x00, 0x00, 0x10, 0x38, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x3B, 0x17, 0x0F, 0x07, 0x03 },
  },
  {
    { 0xE0, 0xF0, 0xF8, 0xF4, 0xEE, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0x0E, 0x04, 0x00, 0x00, 0x00 },
    { 0x1F, 0x3F, 0x7F, 0xBF, 0xDF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00 },
    { 0xFC, 0xFE, 0xFF, 0xFE, 0xFD, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC },
    { 0x03, 0x07, 0x0F, 0x17, 0x3B, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x3B, 0x17, 0x0F, 0x07, 0x03 },
  },
  {
    { 0x00, 0x00, 0x00, 0x04, 0x0E, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0xEE, 0xF4, 0xF8, 0xF0, 0xE0 },
    { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x1F, 0x3F, 0x7F, 0x3F, 0x1F },
    { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xFC, 0xFE, 0xFF, 0xFE, 0xFC },
    { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x03, 0x07, 0x0F, 0x07, 0x03 },
  },
  {
    { 0xE0, 0xF0, 0xF8, 0xF4, 0xEE, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0xEE, 0xF4, 0xF8, 0xF0, 0xE0 },
    { 0x1F, 0x3F, 0x7F, 0xBF, 0xDF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xDF, 0xBF, 0x7F, 0x3F, 0x1F },
    { 0xFC, 0xFE, 0xFF, 0xFE, 0xFD, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC },
    { 0x03, 0x07, 0x0F, 0x17, 0x3B, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x3B, 0x17, 0x0F, 0x07, 0x03 },
  },
  {
    { 0xE0, 0xF0, 0xF8, 0xF4, 0xEE, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F,
0x1F, 0xEE, 0xF4, 0xF8, 0xF0, 0xE0 },
    { 0x1F, 0x3F, 0x7F, 0xBF, 0xDF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xDF, 0xBF, 0x7F, 0x3F, 0x1F },
    { 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC },
    { 0x00, 0x00, 0x00, 0x10, 0x38, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C, 0x7C,
0x7C, 0x7C, 0x3B, 0x17, 0x0F, 0x07, 0x03 },
  }
};

static const byte SecondIndicator[4] =
{
  0x00, 0x07, 0x70, 0x00
};

void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);

  LcdWrite( LCD_CMD, 0x21 ); // LCD Extended Commands.
  LcdWrite( LCD_CMD, 0xC8 ); // Set LCD Vop (Contrast)
  LcdWrite( LCD_CMD, 0x06 ); // Set Temp coefficent
  LcdWrite( LCD_CMD, 0x14 ); // LCD bias mode 1:48

  LcdWrite( LCD_CMD, 0x20 ); // LCD Standard Commands.
  LcdWrite( LCD_CMD, 0x0C ); // LCD in normal mode. 0x0d for inverse
}
```

```
void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

void LcdClear(void)
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}

void Spacer()
{
  LcdWrite(LCD_D, 0x00);
  LcdWrite(LCD_D, 0x00);
}

void DisplayTime(byte hour, byte minutes, byte seconds)
{
  byte components[4] =
  {
    (byte)(hour / 10),
    (byte)(hour % 10),
    (byte)(minutes / 10),
    (byte)(minutes % 10)
  };

  for(byte row = 0; row < 4; row++)
  {
    LcdWrite(LCD_C, 0x80 | 0);
    LcdWrite(LCD_C, 0x40 | row);

    for(byte digit = 0; digit < 4; digit++)
    {
      for(byte col = 0; col < 18; col++)
      {
        LcdWrite(LCD_D, Digits[components[digit]][row][col]);
      }

      Spacer();

      if(digit == 1) // Display second indicator after the second digit
      {
        DisplaySecondIndicator(row, seconds & 0x01);
      }
    }
  }

  DrawSecondsBar(seconds);
}

void DisplaySecondIndicator(byte row, boolean show)
{
  for(int secondIndicatorSegment = 0; secondIndicatorSegment < 3; secondIndicator-
Segment++)
  {
    if(show)
    {
      LcdWrite(LCD_D, SecondIndicator[row]);
    }
    else     {
      LcdWrite(LCD_D, 0x00); // clear
    }
  }
```

```
  Spacer();
}

void DrawSecondsBar(byte seconds)
{
  // Position the pointer
  LcdWrite(LCD_C, 0x80 | 0x0b);
  LcdWrite(LCD_C, 0x44);

  // Draw the left side of the progress bar box
  LcdWrite(LCD_D, 0xF0);

  for(byte i = 0; i < 59; i++)
  {
    if(i < seconds)
    {
      LcdWrite(LCD_D, 0xF0);
    }
    else
    {
      LcdWrite(LCD_D, 0x90);
    }
  }

  // Draw the right side of the progress bar box
  LcdWrite(LCD_D, 0xF0);
}

byte tcnt2;
unsigned long time = 0; // 86390000;

void setup(void)
{
  SetupInterrupt();
  InitializeDisplay();
}

// Credits for the interrupt setup routine:
// http://popdevelop.com/2010/04/mastering-timer-interrupts-on-the-arduino/
void SetupInterrupt()
{
  /* First disable the timer overflow interrupt while we're configuring */
  TIMSK2 &= ~(1<<TOIE2);

  /* Configure timer2 in normal mode (pure counting, no PWM etc.) */
  TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
  TCCR2B &= ~(1<<WGM22);

  /* Select clock source: internal I/O clock */
  ASSR &= ~(1<<AS2);

  /* Disable Compare Match A interrupt enable (only want overflow) */
  TIMSK2 &= ~(1<<OCIE2A);

  /* Now configure the prescaler to CPU clock divided by 128 */
  TCCR2B |= (1<<CS22)  | (1<<CS20); // Set bits
  TCCR2B &= ~(1<<CS21);             // Clear bit

  /* We need to calculate a proper value to load the timer counter.
   * The following loads the value 131 into the Timer 2 counter register
   * The math behind this is:
   * (CPU frequency) / (prescaler value) = 125000 Hz = 8us.
   * (desired period) / 8us = 125.
   * MAX(uint8) + 1 - 125 = 131;
   */


  /* Save value globally for later reload in ISR */
  tcnt2 = 131;
```

```
  /* Finally load end enable the timer */
  TCNT2 = tcnt2;
  TIMSK2 |= (1<<TOIE2);
}

void InitializeDisplay()
{
  LcdInitialise();
  LcdClear();
}

/*
 * Install the Interrupt Service Routine (ISR) for Timer2 overflow.
 * This is normally done by writing the address of the ISR in the
 * interrupt vector table but conveniently done by using ISR()  */

ISR(TIMER2_OVF_vect) {
  /* Reload the timer */
  TCNT2 = tcnt2;

  time++;
  time = time % 86400000;
}

void loop(void)
{
  unsigned long t = (unsigned long)(time/1000);

  DisplayTime((byte)(t / 3600), (byte)((t / 60) % 60), (byte)(t % 60));
}
```