```
#include <TimeLib.h>        // for Date/Time operations

#include <ESP8266WiFi.h>    // for WiFi

#include <WiFiUdp.h>        // for UDP NTP

#include <Adafruit_GFX.h>   // Core graphics library

#include <Adafruit_ST7735.h> // Hardware-specific library

#include <SPI.h>            // Serial Peripheral Interface


#define TFT_CS     15

#define TFT_RST    0

#define TFT_DC     5


// Option 1 (recommended): must use the hardware SPI pins

// (for UNO thats sclk = 13 and sid = 11) and pin 10 must be

// an output. This is much faster - also required if you want

// to use the microSD card (see the image drawing example)

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS,  TFT_DC, TFT_RST);


// Option 2: use any pins but a little slower!

#define TFT_SCLK 13   // set these to be whatever pins you like!

#define TFT_MOSI 11   // set these to be whatever pins you like!

//Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK, TFT_RST);


const int softoffPin =  4;

int softoffState = HIGH;

const int maxLoopsBeforeSoftoff = 10;


char ssid[] = "************";      //  your network SSID (name)

char pass[] = "************";      // your network password


// NTP Servers:
```

```
IPAddress timeServer(195, 186, 4, 101); // 195.186.4.101 (bwntp2.bluewin.ch)
const char* ntpServerName = "ch.pool.ntp.org";
const int timeZone = 2;     // Central European Time (summer time)

WiFiUDP Udp;
unsigned int localPort = 8888;  // local port to listen for UDP packets

void setup(void) {

  pinMode(softoffPin, OUTPUT);

  Serial.begin(115200);

  Serial.print("Connecting to wifi ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.print("IP number assigned by DHCP is ");
  Serial.println(WiFi.localIP());
  Serial.println("Starting UDP");
  Udp.begin(localPort);
  Serial.print("Local port: ");
  Serial.println(Udp.localPort());

  Serial.print("Using NTP Server ");
  Serial.println(ntpServerName);
  //get a random server from the pool
  WiFi.hostByName(ntpServerName, timeServer);
  Serial.print("NTP Server IP ");
  Serial.println(timeServer);

  Serial.println("waiting for sync");
  setSyncProvider(getNtpTime);
  setSyncInterval(60);

  /*
    time_t t = getNtpTime();
    setTime(t);
  */

  digitalClockDisplay();

  tft.initR(INITR_144GREENTAB);   // initialize a ST7735S chip, black tab
  Serial.println("TFT Initialized");

  digitalClockDisplayToTft();
  Serial.println("Output to TFT done");
}

int counter = 0;

void loop() {

  if (counter == maxLoopsBeforeSoftoff) {
    Serial.println("System Shutdown");
    digitalWrite(softoffPin, softoffState);
  }
  digitalClockDisplayToTft();


  //time = millis() - time;
  //Serial.println(time, DEC);


  counter ++;
  delay(850);
}



/*-------- TFT code ----------*/


void digitalClockDisplayToTft() {
  // digital clock display of the time at the TFT
  tft.setTextWrap(false);
  tft.fillScreen(ST7735_BLACK);
  tft.setCursor(0, 20);
  tft.setTextColor(ST7735_RED);
  tft.setTextSize(1);
  tft.println(" NTP Time CH");
  tft.setTextColor(ST7735_BLUE);
  tft.println("");
  tft.println("");
  tft.setTextSize(2);
  tft.print(" ");
  tft.print(hour());
  printDigitsToTft(minute());
  printDigitsToTft(second());
  tft.println("");
  tft.println("");
  tft.setTextColor(ST7735_GREEN);
  tft.setTextSize(2);
  tft.print(" ");
  tft.print(day());
  tft.print(".");
  tft.print(month());
  tft.print(".");
  tft.print(year());
  tft.println();
}

void printDigitsToTft(int digits) {
  // utility for digital clock display: prints preceding colon and leading 0
  tft.print(":");
  if (digits < 10)
    tft.print('0');
  tft.print(digits);
}


/*-------- Serial Debug Code ----------*/

void digitalClockDisplay() {
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(".");
  Serial.print(month());
  Serial.print(".");
  Serial.print(year());
  Serial.println("");
}

void printDigits(int digits) {
  // utility for digital clock display: prints preceding colon and leading 0
  Serial.print(":");
  if (digits < 10)
    Serial.print('0');
  Serial.print(digits);
}



/*-------- NTP code ----------*/

const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets

time_t getNtpTime()
{
  while (Udp.parsePacket() > 0) ; // discard any previously received packets
  Serial.println("Transmit NTP Request");
  sendNTPpacket(timeServer);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Serial.println("Receive NTP Response");
      Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
      unsigned long secsSince1900;
      // convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
    }
  }
  Serial.println("No NTP Response :-(");
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the given address
void sendNTPpacket(IPAddress &address)
{
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;
  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket();
}
```