Textul și imaginile din acest document sunt licențiate

Attribution-NonCommercial-NoDerivs
CC BY-NC-ND

Codul sursă din acest document este licențiat

Public-Domain

Ești liber să distribui acest document prin orice mijloace consideri (email, publicare pe website / blog, tipărire, sau orice alt mijloc), atât timp cât nu aduci nici un fel de modificări acestuia. Codul sursă din acest document poate fi utilizat în orice fel de scop, de natură comercială sau nu, fără nici un fel de limitări dar autorii nu își asumă nici o răspundere pentru pagubele pricinuite de implementările realizate de utilizatori. Schemele și codul sursă au un rol educativ și nu sunt gândite pentru a fi utilizate în mediu de producție (industrial, casnic sau comercial).

Ceas IoT cu programare OTA

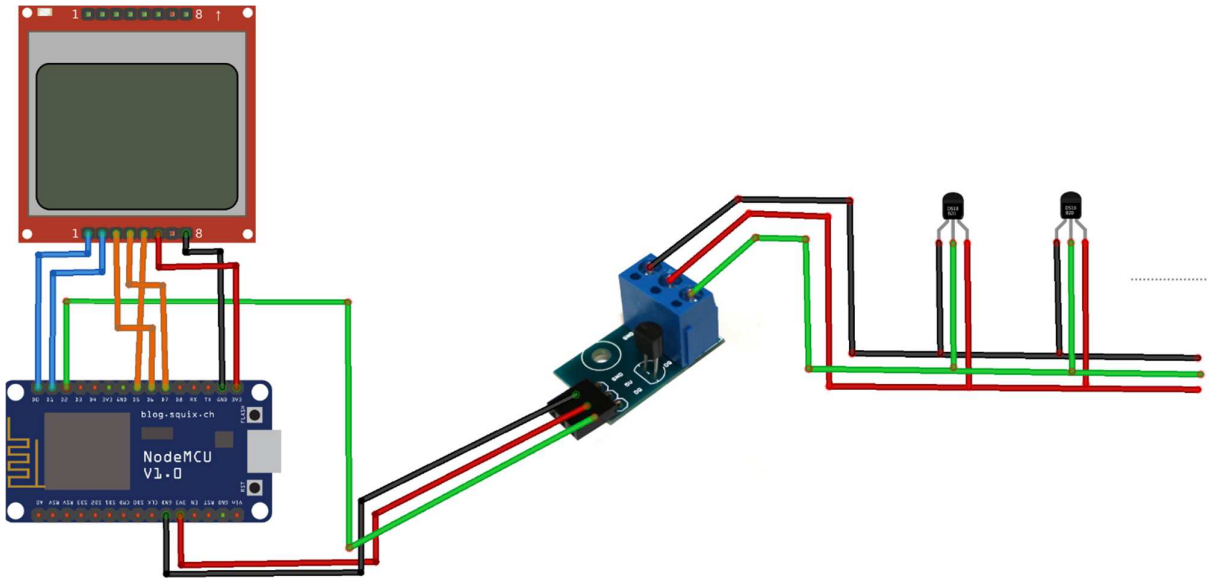
Realizarea unui ceas electronic este un proiect simplu dacă utilizăm o placă de dezvoltare programabilă dar asta nu înseamnă că nu există anumite provocări și în acest caz. Una din provocările majore ale implementării unui ceas electronic este acuratețea menținerii orei (asigurarea orei exacte). Acest lucru se realizează de obicei prin utilizarea unui modul RTC (1) dar în cadrul lecției de față vom utiliza o placă de dezvoltare NodeMCU (2) care prin conectivitatea WiFi de care dispune ne va permite să realizăm o sincronizare de timp în rețea de tip NTP (3). Astfel nu vom avea nevoie de nici o componentă suplimentară pentru a asigura ora exactă.

Pentru afișare vom utiliza un ecran LCD grafic (4) cu rezoluția de 84x48 pixeli de tip PCD8544 (LCD de Nokia 5110). Funcționalitatea specială propusă în această lecție este adăugarea sistemului unei funcționalități de tip IoT adică raportarea prin Internet a unui parametru de mediu către o platformă IoT. Sistemul va măsura (și afișa) temperatura și o va trimite pentru înregistrare către platforma Robofun IoT (5). Astfel sistemul nostru nu va avea simpla funcționalitate de ceas/termometru electronic ci și de senzor IoT. Pentru măsurarea temperaturii vom utiliza un senzor digital brick DS18B20 (6). Acest senzor are o acuratețe mare în măsurarea temperaturii, nu necesită decât 3 fire pentru conectarea la placa de dezvoltare și, cel mai interesant, permite de conectarea (pe aceleași trei fire) a mai multor senzori – sistemul poate fi extins foarte ușor prin adăugarea de noi senzori ce măsoară temperatura în zone diferite (interior / exterior, diverse camere etc.).

Ultima funcționalitate interesantă propusă pentru sistemul prezentat în această lecție este posibilitatea de programare la distanță – programare OTA (Over-The-Air) (7). Pentru a reprograma / reîncărca programul nu este nevoie să desfacem carcasa sistemului și să conectăm placa de

dezvoltare la calculator ci acest lucru poate fi făcut prin conexiunea WiFi a plăcii de dezvoltare. Singura limitare a acestei metode este că dimensiunea programului nu poate depăși jumătate din memoria program a plăcii de dezvoltare dar nu este cazul sistemului nostru.

Interconectarea componentelor (placă de dezvoltare, senzor de temperatură și ecran LCD) este prezentată în diagrama următoare:



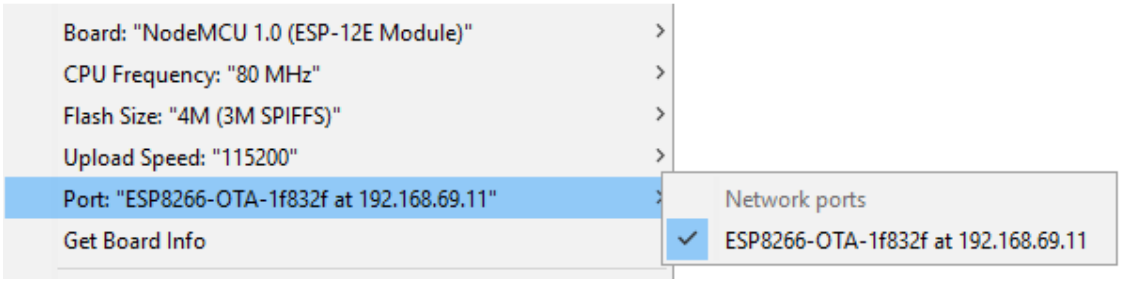
Senzorul de temperatură va avea pinul D0 conectat la pinul D2 al plăcii de dezvoltare iar pinii 5V și GND la pinii 3.3V și GND ai plăcii de dezvoltare (senzorul funcționează la tensiuni între 3V și 5.5V). Chiar dacă programul prezentat în cele ce urmează este gândit pentru un sistem cu un singur senzor, sistemul acceptă adăugarea pe aceleași 3 fire a mai multor senzori de temperatură pe distanțe de până la 200 de metri (8).

Ecranul LCD comunică cu placa de dezvoltare prin protocoul SPI și implică următoarele conexiuni:

- Pinul 1 (RST) al ecranului LCD se conectează la pinul D0 al plăcii de dezvoltare;
- Pinul 2 (CE) al ecranului la pinul D1 al plăcii de dezvoltare;
- Pinul 3 (DC) al ecranului la pinul D6 (HMISO) al plăcii de dezvoltare;
- Pinul 4 (DIN) al ecranului la pinul D7 (HMOSI) al plăcii de dezvoltare;
- Pinul 5 (CLK) al ecranului la pinul D5 (HSCLK) al plăcii de dezvoltare;
- Pinul 6 (VCC) al ecranului la un pin de 3.3V al plăcii de dezvoltare;

- Pinul 7 (LIGHT) al ecranului nu este conectat în schema noastră. Acest pin comandă aprinderea luminii de fundal a ecranului. Se poate conecta la un pin digital liber pentru comandă digitală sau PWM;
- Pinul 8 (GND) al ecranului la un pin GND al ecranului.

Programul a fost dezvoltat și testat utilizând Arduino IDE 1.8.3 cu extensia ESP8266 Community 2.3.0 instalată. Prima încărcare a programului necesită conectarea plăcii la calculator prin intermediul cablului USB dar ulterior placa va putea fi reprogramată prin OTA (calculatorul de pe care se face programarea trebuie să fie în aceiași rețea ca placa NodeMCU):




```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
```

Programul utilizează și următoarele biblioteci: Time 1.5.0 (9) - pentru a menține data și ora, sincronizarea NTP se face automat de către bibliotecă la intervale mai mari de timp; OneWire 2.3.3 și DallasTemperature 3.7.6 - pentru comunicația cu senzorul de temperatură; Adafruit GFX 1.2.2 și Adafruit PCD8544 versiune modificată pentru ESP8266 (10) – pentru partea de afișare pe ecranul LCD. În cadrul programului trebuie personalizate datele de conectarea WiFi (variabilele *ssid[]* și *pass[]*). Constanta *timeZone* indică fusul orar (2 sau 3 pentru România, oră de iarnă sau vară).

```
#include <TimeLib.h>
const int timeZone = 3;
WiFiUDP Udp;
const int NTP_PACKET_SIZE = 48;
```

```
byte packetBuffer[NTP_PACKET_SIZE];
unsigned int localPort = 2390;
char ssid[] = "…";
char pass[] = "…";
WiFiClient client;
#define HOSTNAME "ESP8266-OTA-"
#include <DallasTemperature.h>
#include <OneWire.h>
#define ONE_WIRE_BUS D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
const int8_t RST_PIN = D0;
const int8_t CE_PIN = D1;
const int8_t DC_PIN = D6;
Adafruit_PCD8544 display =
            Adafruit_PCD8544(DC_PIN, CE_PIN, RST_PIN);
```

Secțiunea *setup()* se va ocupa cu inițializarea conexiunii WiFi (în cazul unei erori pe ecranul LCD va apărea mesajul *!WiFi* și sistemul va intra într-o buclă de reinițializarea până la conectarea la rețea), inițializarea rețelei OneWire de senzori, inițializarea ecranului și a rutinei de programare OTA. În cazul unei inițializări normale pe ecranul LCD vor apărea informații legate de inițializarea sistemului (denumirea OTA, denumirea rețelei WiFi, adresa IP).

```
void setup() {
  sensors.begin();
  delay(100);
  display.begin();
  display.setContrast(60);
  display.setTextColor(BLACK);
```

```
display.setCursor(0,0);
display.setTextSize(1);
display.clearDisplay();
display.display();
String hostname(HOSTNAME);
hostname += String(ESP.getChipId(), HEX);
WiFi.hostname(hostname);
display.println("Hostname:" + hostname);
display.display();
WiFi.begin(ssid, pass);
delay(5000);
if (WiFi.status() != WL_CONNECTED) {
  display.clearDisplay();
  display.setCursor(17,16);
  display.setTextSize(2);
  display.print("!WiFi");
  display.display();
  delay(10000);
  ESP.restart();
}
display.print("Connected to ");
display.println(ssid);
display.print("IP:");
display.println(WiFi.localIP());
Udp.begin(localPort);
setSyncProvider(getNtpTime);
ArduinoOTA.setHostname((const char *)hostname.c_str());
ArduinoOTA.begin();
display.display();
delay(1000);
display.clearDisplay();
display.display(); }
```

Secțiunea *loop()* se va ocupa cu achiziția parametrului temperatură și cu partea de afișare. Apelarea rutinei de postare IoT se va face la un interval de o oră (constanta *postingInterval*). Pe ecranul LCD vor apărea alternativ (câte 30 de secunde fiecare) ora/data și temperatura. Ecranul de afișare al temperaturii va conține și mesajul OK/Fail indicând dacă ultima postare IoT s-a efectuat cu succes sau nu (astfel sistemul are și o funcționalitate de supraveghere a bunei funcționări a serviciului IoT).


```
unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 60L * 60L * 1000L;
boolean myiot_ok;

void loop() {
    ArduinoOTA.handle();
    sensors.requestTemperatures();
    delay(100);
    float temperature = sensors.getTempCByIndex(0);
    display.clearDisplay();
    if (second()<30) {
      display.drawRoundRect(2,2,80,44,3,BLACK);
      display.setTextColor(BLACK);
      display.setCursor(13,10);
      display.setTextSize(2);
      printDigits(hour(),false);
      printDigits(minute(),true);
      display.println();
      display.setTextSize(1);
```

```
  display.setCursor(13,26);
  display.print(day());
  display.print("/");
  display.print(month());
  display.print("/");
  display.print(year());
  display.println();
  display.display();
}
else {
  display.fillScreen(BLACK);
  display.drawRoundRect(2,2,80,44,3,WHITE);
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(22,10);
  display.print((int)temperature);
  display.print((char)247);
  display.println("C");
  display.setTextSize(2);
  if (myiot_ora!=-1) {
    if (myiot_ok) {
      display.setCursor(30,26);
      display.println("OK");    }
    else {
      display.setCursor(20,26);
      display.println("FAIL"); }
  }
  display.display();
}
if (millis() - lastConnectionTime > postingInterval) {
  if (IoTpublish(temperature)>0)
    { myiot_ok=true; }
```

```
      else
        { myiot_ok=false; }
    }
}
```

Funcția *IoTpublish* este responsabilă cu postarea IoT și returnează codul returnat de operația HTTP GET efectuată. În cadrul acesteia trebuie completată cheia de autentificare oferită de serviciul Robofun IoT (variabila *SENSOR_TOKEN*). Procedura *printDigits* este folosită pentru afișarea mai simplă a orei.

```
int IoTpublish(float temperature) {
  String SENSOR_TOKEN = "…";
  HTTPClient http;
  String data =
    String("http://iot.robofun.ro/api/v1/senzor/") +
    SENSOR_TOKEN + "/input?value=" +
    String(temperature, DEC);
  http.begin(data);
  int httpCode = http.GET();
  delay(100);
  if(httpCode > 0) {
      if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
      }
   }
  http.end();
  lastConnectionTime = millis();
  return httpCode;
}

void printDigits(int digits, boolean dots){
   if (dots) display.print(":");
  if(digits < 10)
```

```
    display.print('0');
  display.print(digits);
}
```

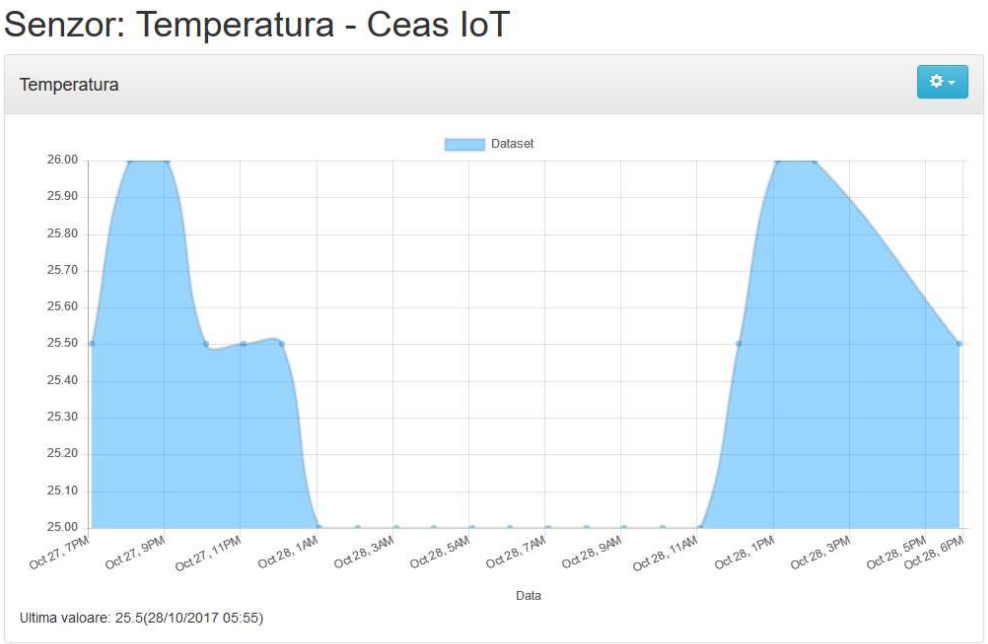
Funcțiile *getNtpTime* și *sendNTPpacket* sunt apelate automat de către biblioteca Time și asigură partea de comunicație NTP. Constanta *ntpServerName* indică serverul NTP care este utilizat pentru sincronizarea de timp.

```
IPAddress timeServerIP;
const char* ntpServerName = "time.nist.gov";
time_t lastsyncr;
time_t getNtpTime() {
  WiFi.hostByName(ntpServerName, timeServerIP);
  while (Udp.parsePacket() > 0) ;
  sendNTPpacket(timeServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 2500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
      Udp.read(packetBuffer, NTP_PACKET_SIZE);
      unsigned long secsSince1900;
      secsSince1900 =  (unsigned long)packetBuffer[40]
                              << 24;
      secsSince1900 |= (unsigned long)packetBuffer[41]
                              << 16;
      secsSince1900 |= (unsigned long)packetBuffer[42]
                              << 8;
      secsSince1900 |= (unsigned long)packetBuffer[43];
      lastsyncr = (time_t) (secsSince1900 - 2208988800UL +
                 timeZone * SECS_PER_HOUR);
      return lastsyncr;
    }
  }
```

```
  return 0;
}

void sendNTPpacket(IPAddress &address) {
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = 0b11100011;
  packetBuffer[1] = 0;
  packetBuffer[2] = 6;
  packetBuffer[3] = 0xEC;
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;
  Udp.beginPacket(address, 123);
  Udp.write(packetBuffer, NTP_PACKET_SIZE);
  Udp.endPacket(); }
```

După o oră după punerea în funcțiune a sistemului se pot consulta datele înregistrate în cadrul platformei Robofun IoT. Mai jos este o captură de ecran cu datele înregistrate de către sistemul de test în decursul unei zile.



https://www.robofun.ro/forum/

Referințe on-line

(1) Module RTC
https://www.robofun.ro/index.php?route=product/search&search=RTC

(2) NodeMCU v2
https://www.robofun.ro/nodemcu-v2-lua-based-esp8266-development-kit?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(3) Network Time Protocol
https://en.wikipedia.org/wiki/Network_Time_Protocol

(4) Graphic LCD 84x48 - Nokia 5110
https://www.robofun.ro/graphic-lcd-84x48-nokia-5110?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(5) Robofun IoT
http://iot.robofun.ro/

(6) Senzor Temperatura Inlantuibil Brick (DS18B20) - Motherboard
https://www.robofun.ro/senzor-temperatura-inlantuibil-brick-DS18B20-motherboard?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(7) OTA Update · ESP8266 Arduino Core
http://esp8266.github.io/Arduino/versions/2.0.0/doc/ota_updates/ota_updates.html

(8) Guidelines for Reliable Long Line 1-Wire Networks - Tutorial – Maxim
https://www.maximintegrated.com/en/app-notes/index.mvp/id/148

(9) Arduino Playground - Time
http://playground.arduino.cc/code/time

(10) GitHub - bbx10/Adafruit-PCD8544-Nokia-5110-LCD-library at esp8266
https://github.com/bbx10/Adafruit-PCD8544-Nokia-5110-LCD-library/tree/esp8266