```
/*
  A simple string is read from https://larsenhenneberg.dk/elpristabel.php
  that holds the time of day, date and the electricity price indexes.
  The data part of the string looks like this:
  Time: 20:29 2023-11-25,6,4,3,2,1,5,7,10,15,14,12,11,9,8,13,16,18,22,23,24,21,19,20,17,
                         10,8,5,3,1,2,4,7,11,16,15,13,12,9,14,19,20,23,24,22,21,18,17,6
  It holds the prioritized price indexes for 2 days, 2x24 hours of price indexes.
  Each day has 24 indexes with 1 indicating the lowest price index and 24 the highest price index.
  The test hardware is an ESP8266 S01 module, a relay board and a suitable USB programmer.
*/
#define TZ_DENMARK "CET-1CEST,M3.5.0,M10.5.0/3"
#define LED_pin 2  // GPIO2 as output , active low for LED light
#define Relay_pin 0  // GPIO0 as output , active low for relay on
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ESP8266HTTPClient.h>

// local wireless network setup, add your wifi settings
const char* ssid = "...";
const char* password = ".....";
HTTPClient https;
WiFiClientSecure client;
const char *TZstr = TZ_DENMARK;
struct tm timeinfo;
time_t now;
String TimerOn_str = "   ";
String Hour_str = "   ";
String Minute_str = "   ";
int Hours[2][24];  // 2 days elec. price table on hour base
String https_response;
int strIdx;
int Hour;       // Hours 0..23
int Minute;     // Minutes 0..59
int Day;        // Day becomes 0 on a new price table read
int idx;

// Root certificate for https://larsenhenneberg.dk
const char IRG_Root_X1 [] PROGMEM = R"CERT(
-----BEGIN CERTIFICATE-----
MIIGJTCCBQ2gAwIBAgIQM93couXJ2ikRcL65I3p1fTANBgkqhkiG9w0BAQsFADBy
MQswCQYDVQQGEwJVUzELMAkGA1UECBMCVFgxEDAOBgNVBAcTB0hvdXN0b24xFTA
T
BgNVBAoTDGNQYW5lbCwgSW5jLjEtMCsGA1UEAxMkY1BhbmVsLCBJbmMuIENlcnRp
ZmljYXRpb24gQXV0aG9yaXR5MB4XDTIzMTEwMTAwMDAwMFoXDTI0MDEzMDIzNTk1
OVowHTEbMBkGA1UEAxMSbGFyc2VuaGVubmViZXJnLmRrMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAsGu3MSXMhFCt2ruCmbeks3le1zxhRHOgbllShsjd
4yGcnIdhkOXAH5+6IlPezeHKncOdFWZ7M1R7vW8QQ3EI3uf3qqu2EHIdqLSEus/O
asoBlwq9i2fQl89mR+Y8AoRx2aBcSa7P19OYlsqFBnrG7PWX1i58mSguxDUaaK0V
G/xYOm8vZyHQiMLXRaeteqsM2UtdHpEq/cQH4LL11AxUZuoLybFT+8daGsv2bIKG
ZJvs5FB9Q1AAhn4bUtIbZgKZ6AGniK3AN4tklOX/i3nlcMcco5OaZSRZKwvS4FjD
2vnZdUdzPUDte4Po2t3DcNcwoQ+7lAzwmvDVMlhFQcx1xwIDAQABo4IDCjCCAwYw
HwYDVR0jBBgwFoAUfgNaZUFrp34K4bidCOodjh1qx2UwHQYDVR0OBBYEFAsaUWJb
```

```
lNQ7l4YxL3yQbbut8dYRMA4GA1UdDwEB/wQEAwIFoDAMBgNVHRMBAf8EAjAAMB0G
A1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjBJBgNVHSAEQjBAMDQGCysGAQ
QB
sjEBAgI0MCUwIwYIKwYBBQUHAgEWF2h0dHBzOi8vc2VjdGlnby5jb20vQ1BTMAgG
BmeBDAECATBMBgNVHR8ERTBDMEGgP6A9hjtodHRwOi8vY3JsLmNvbW9kb2NhLmNv
bS9jUGFuZWxJbmNDZXJ0aWZpY2F0aW9uQXV0aG9yaXR5LmNybDB9BggrBgEFBQcB
AQRxMG8wRwYIKwYBBQUHMAKGO2h0dHA6Ly9jcnQuY29tb2RvY2EuY29tL2NQYW5l
bEluY0NlcnRpZmljYXRpb25BdXRob3JpdHkuY3J0MCQGCCsGAQUFBzABhhhodHRw
Oi8vb2NzcC5jb21vZG9jYS5jb20wggEEBgorBgEEAdZ5AgQCBIH1BIHyAPAAdgB2
/4g/Crb7lVHCYcz1h7o0tKTNuyncaEIKn+ZnTFo6dAAAAYuIghY/AAAEAwBHMEUC
IQCujY/CbtZJ5fiz87aVoTNyZ+LqJXfJ2NaAkWBJe4ivBwIgQKg2ixTwsbx7ZxmT
DKcwqT/VQ1+wsbNflruvm4d2Gy0AdgDatr9rP7W2Ip+bwrtca+hwkXFsu1GEhTS9
pD0wSNf7qwAAAYuIghaXAAAEAwBHMEUCIQCZOWHgi++tH6uGTMx2q3dIQpOSH57m
JK9U53D7BflV9QIgFLEsc3++901aO96np/GsikR6ALzZa4WA3brzoRGUEuUwZwYD
VR0RBGAwXoISbGFyc2VuaGVubmViZXJnLmRrghdpcHY2LmxhcnNlbmhlbm5lYmVy
Zy5ka4IXbWFpbC5sYXJzZW5oZW5uZWJlcmcuZGuCFnd3dy5sYXJzZW5oZW5uZWJl
cmcuZGswDQYJKoZIhvcNAQELBQADggEBAHsOYlBrrFoTSjT4CF6iM0WopGDT3R9c
rnI6EJPpGWimVTgSvdWAvcV7+w49PlnQqq+/W5pEqkZnaWH3C7xhISZgxBzugrLk
1fWrVGPsVORM6WQ9N006voFSOarVCyTkUWzMEytSzWMkqiWLtW3JBf28dAAqkW+b
JBnhwRl2H1oLrpsuh4gATF38JvNTcbfOoRZRK2gQMHS2oYqGyEfzjaZdR3K6IUef
t6OmxBFLNPz9XMnmFuFpq1vGLmVDEiLF++VWcwGT9yfaco7x3bb9bVznA46hxwa9
wNmWewjTMEN3O0eOqGp1E8cQ3krvwsTkNUTSLOgmBtoGaQPVE8fTfHc=
-----END CERTIFICATE-----
)CERT";

// Create a list of certificates with the server certificate
X509List cert(IRG_Root_X1);

void setup() {
  pinMode(LED_pin, OUTPUT); // GPIO2 ( LED pin ) as output
  pinMode(Relay_pin, OUTPUT); // GPIO2 ( LED pin ) as output
  digitalWrite(LED_pin, 1); // set LED inactive
  digitalWrite(Relay_pin, 1); // set relay inactive
  Serial.begin(115200);
  //Serial.setDebugOutput(true);
  Serial.println();
  WiFi.mode(WIFI_STA);      //Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    digitalWrite(LED_pin, 0);  // short 500 msec LED blink to
    delay(500);                // indicate missing
    digitalWrite(LED_pin, 1);  // wifi link
    delay(500);
  }
  // Set time via NTP, as required for x.509 validation
  //configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
  configTime (TZstr, "pool.ntp.org", "time.nist.gov");

  now = time(nullptr);
  while (now < 8 * 3600 * 2) {
```

```
    delay(500);
    now = time(nullptr);
  }
  gmtime_r(&now, &timeinfo);
  // empty prices tables , output not active for a start
  for (int x=0; x<24; x++) Hours[0][x] = 0;  // reset ON/OFF state table
  for (int x=0; x<24; x++) Hours[1][x] = 0;  // reset ON/OFF state table

  Minute = 0;    // force first server request in loop()
}

void loop() {
  // if wifi available then request server each 10 minutes
  if ((WiFi.status() == WL_CONNECTED) && ((Minute % 15) == 0)) {
    now = time(nullptr);
    while (now < 8 * 3600 * 2) {
      delay(500);
      now = time(nullptr);
    }
    gmtime_r(&now, &timeinfo);
    client.setTrustAnchors(&cert);
    if (https.begin(client, "https://www.larsenhenneberg.dk/elpristabel.php"))
    {  // issue HTTPS request
     int httpCode = https.GET();      // httpCode will be negative on error
     if (httpCode > 0)
     {       // check file found at server
      if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
      {
       https_response = https.getString();
       idx = https_response.indexOf("Time: ");
       https_response.remove(0, idx+6);  // strip to hour:minute field
       Hour_str[0] = https_response[0];
       Hour_str[1] = https_response[1];
       Hour = Hour_str.toInt();         // save hour as integer
       Minute_str[0] = https_response[3];
       Minute_str[1] = https_response[4];
       Minute = Minute_str.toInt();     // save minute as integer
       idx = https_response.indexOf(",");
       https_response.remove(0, idx+1);  // strip to comma separated values
       // empty prices tables , output not active for a start
       for (int x=0; x<24; x++) Hours[0][x] = 0;  // reset ON/OFF state table
       for (int x=0; x<24; x++) Hours[1][x] = 0;  // reset ON/OFF state table
       Day = 0; // start with day 0 for Hours[0] pricetable
       strIdx = 0;
       idx = https_response.indexOf(",");
       while (idx != -1)
       {
        if (strIdx <= 23) {
          Hours[0][strIdx++] = https_response.substring(0, idx).toInt();
        }
        else {
          Hours[1][-24+strIdx++] = https_response.substring(0, idx).toInt();
```

```
      }
      https_response.remove(0, idx+1);
      idx = https_response.indexOf(",");
     }
      if (strIdx <= 23) {
        Hours[0][strIdx++] = https_response.substring(0, idx).toInt();
      }
      else {
        Hours[1][-24+strIdx++] = https_response.substring(0, idx).toInt();
      }
 /*    Serial.println();
     for (idx = 0; idx < strIdx; idx++)
     {
      if (idx <= 23) {
       Serial.print(Hours[0][idx]);
      }
      else {
      if (idx == 24) Serial.println();
       Serial.print(Hours[1][-24+idx]);
      }
      Serial.print(",");
     } */
     Serial.println();
     Serial.print(Hour);
     Serial.print(":");
     Serial.print(Minute);
    }
    } else {
      Serial.printf("[HTTPS] GET failed, error: %s\n", https.errorToString(httpCode).c_str());
    }
    https.end();
  } else {
    Serial.printf("[HTTPS] Unable to connect\n");
  }
}
// the next  line can be replaced by a prioritized ON scheme
if ((Hours[Day][Hour] > 0) && (Hours[Day][Hour] <= 4))
{
  digitalWrite(LED_pin, 0); // set LED output active
  digitalWrite(Relay_pin, 0); // set relay output active
}
else
{
  digitalWrite(LED_pin, 1);   // turn off LED
  digitalWrite(Relay_pin, 1); // disable relay output
}
delay(59000);
Minute++;
if (Minute == 60)
{
  Minute = 0;
  Hour++;
```

```
    if (Hour == 24) {
      Hour = 0;
      Day = 1;  // continue with Hours[1] price table
    }           //
  }
}
```