การทดลองที่ 5 การเทียบเวลามาตรฐาน NTP

การดึงค่าเวลามาตรฐานสากลจากอินเตอร์เน็ต จะดึงจาก Server ที่ให้บริการโพรโทคอล NTP โดย NTP Server มีอยู่ด้วยกันหลายที่

คำว่า .NTP" จาก https://www.thaicert.or.th/papers/technical/2014/pa2014te002.html ได้อธิบายความหมายของ ว่า "NTP หรือที่ย่อมาจากคำว่า Network Time Protocol เป็นโพรโทคอลสำหรับการเทียบเวลามาตรฐานและตั้งค่าเวลาบนเครื่องคอมพิวเตอร์นั้นๆผ่านระบบเครือข่าย ส่วนใหญ่ถูกนำมาใช้ในองค์กรเพื่อช่วยทำให้เวลาในเครื่องคอมพิวเตอร์หรือเครื่องแม่ข่ายนั้นมีค่าตรงกัน โดยรูปแบบการทำงานนั้นแบ่งเป็น 2 ส่วนหลักๆคืออุปกรณ์ที่ให้บริการเทียบเวลา (NTP Server) กับเครื่องคอมพิวเตอร์หรืออุปกรณ์ที่ต้องการเทียบเวลา (NTP Client) โดยโพรโทคอล NTP นั้นเชื่อมต่อผ่านทางระบบเครือข่ายด้วยโพรโทคอล UDP ผ่านพอร์ต 123 (โพรโทคอล UDP มีความรวดเร็วในการรับส่งข้อมูลแต่ก็มีข้อเสียตรงที่โพรโทคอล UDP ไม่มีการตรวจสอบความถูกต้องของข้อมูลที่รับส่งกันอยู่เหมือน TCP)"

แบบที่ 1 อ่านค่าดิบเป็นวินาที

ค่าที่อ่านได้จะเป็นจำนวนวินาที ตั้งแต่ปี คศ.1900 และเมื่อลบด้วยจำนวนวินาทีของ 7 ปีแล้วนำมาคำนวณจึงจะได้เป็น วัน-เดือน-ปี ชม- นาที และ วินาที

```
//----------------------------------------------------------------------------------------------------------
//  - www.geekstips.com
//  - Arduino Time Sync from NTP Server using ESP8266 WiFi module
//  - Arduino code example
//----------------------------------------------------------------------------------------------------------

#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

const char* ssid = "------------";                  //  your network SSID (name)
const char* password = "------------";              // your network password


unsigned int localPort = 2390;      // local port to listen for UDP packets

/* Don't hardwire the IP address or we won't get the benefits of the pool.
 *  Lookup the IP address for the host name instead */
//IPAddress timeServer(129, 6, 15, 28); // time.nist.gov NTP server
IPAddress timeServerIP; // time.nist.gov NTP server address
const char* ntpServerName = "time.nist.gov";

const int NTP_PACKET_SIZE = 48; // NTP time stamp is in the first 48 bytes of the message

byte packetBuffer[ NTP_PACKET_SIZE]; //buffer to hold incoming and outgoing packets

// A UDP instance to let us send and receive packets over UDP
WiFiUDP udp;
```

```
unsigned char uh,um,us;
unsigned char bh;
//--------------------------------------------------------------------------------------------------------------
void setup()
{
   Serial.begin(115200);
   Serial.println();
   Serial.println();

   // We start by connecting to a WiFi network
   Serial.print("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);

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

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  Serial.println("Starting UDP");
  udp.begin(localPort);
  Serial.print("Local port: ");
  Serial.println(udp.localPort());
}
//--------------------------------------------------------------------------------------------------------------
void loop()
{
    //get a random server from the pool
    WiFi.hostByName(ntpServerName, timeServerIP);

    sendNTPpacket(timeServerIP); // send an NTP packet to a time server
    // wait to see if a reply is available
    delay(1000);

    int cb = udp.parsePacket();
    if (!cb) {
       Serial.println("no packet yet");
    }
    else
   {
      Serial.print("packet received, length=");
      Serial.println(cb);
      // We've received a packet, read the data from it
      udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer

      //the timestamp starts at byte 40 of the received packet and is four bytes,
      // or two words, long. First, esxtract the two words:

      unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
      unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
```

```
      // combine the four bytes (two words) into a long integer
      // this is NTP time (seconds since Jan 1 1900):
      unsigned long secsSince1900 = highWord << 16 | lowWord;
      Serial.print("Seconds since Jan 1 1900 = " );
      Serial.println(secsSince1900);

      // now convert NTP time into everyday time:
      Serial.print("Unix time = ");
      // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
      const unsigned long seventyYears = 2208988800UL;
      unsigned long epoch = secsSince1900 - seventyYears;        // subtract seventy years:
      Serial.println(epoch);          // print Unix time:

      // print the hour, minute and second:
      Serial.print("The UTC time is ");       // UTC is the time at Greenwich Meridian (GMT)
      uh = (epoch  % 86400L) / 3600;
      if ( uh < 10 ) { // In the first 10 minutes of each hour, we'll want a leading '0'
         Serial.print('0');
      }
      Serial.print(uh);               // print the hour (86400 equals secs per day)
      Serial.print(':');
      um = ((epoch % 3600) / 60);
      if ( um < 10 ) {                // In the first 10 minutes of each hour, we'll want a leading '0'
         Serial.print('0');
      }
      Serial.print(um);               // print the minute (3600 equals secs per minute)
      Serial.print(':');

      us = (epoch % 60) < 10;
      if (us < 10) {                  // In the first 10 seconds of each minute, we'll want a leading '0'
      Serial.print('0');
      }
      Serial.println(us); // print the second
  }
  Serial.println("--------------------");
  delay(10000);             // wait ten seconds before asking for the time again
}
//------------------------------------------------------------------------------------------------------------------
// send an NTP request to the time server at the given address
unsigned long sendNTPpacket(IPAddress& address)
{
   Serial.println("sending NTP packet...");
   // 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();
}
```

แบบที่ 2 ใช้ไลบรารี Time.h และใช้ ฟังก์ชั่นมาตรฐาน localtime() or gmtime() and the struct tm. สามารถดึงค่าเป็น วัน-เดือน-ปี ชม- นาที และ วินาที ได้เลย

```
#include <ESP8266WiFi.h>
#include <Time.h>

const char* ssid = ".....";                 //  your network SSID (name)
const char* password = ".....";             // your network password

int timezone = 7 * 3600;                    //ตั้งค่า TimeZone ตามเวลาประเทศไทย
int dst = 0;                                //กำหนดค่า Date Swing Time

void setup()
{
  Serial.begin(115200);
  Serial.setDebugOutput(true);

  WiFi.mode(WIFI_STA);                          //เชื่อมต่อ Wifi
  WiFi.begin(ssid, password);
  Serial.println("\nConnecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(",");
    delay(1000);
  }
  configTime(timezone, dst, "pool.ntp.org", "time.nist.gov");     //ดึงเวลาจาก Server
  Serial.println("\nWaiting for time");
  while (!time(nullptr)) {
   Serial.print(".");
   delay(1000);
  }
  Serial.println("");
}

void loop()
{
  configTime(timezone, dst, "pool.ntp.org", "time.nist.gov");    //ดึงเวลาปัจจุบันจาก Server
  time_t now = time(nullptr);
  struct tm* p_tm = localtime(&now);
  Serial.print(p_tm->tm_hour);
  Serial.print(":");
  Serial.print(p_tm->tm_min);
  Serial.print(":");
  Serial.print(p_tm->tm_sec);
  Serial.println("");

  delay(1000);
}
```

ความหมาย

1. configTime จาก

   https://github.com/esp8266/Arduino/blob/9913e5210779d2f3c4197760d6813270dbba6232/cores/esp8266/time.c#L58 มีรายละเอียดดังนี้

```
void configTime(int timezone, int daylightOffset_sec, const char* server1, const char* server2,
const char* server3)
{
  sntp_stop();

  setServer(0, server1);
  setServer(1, server2);
  setServer(2, server3);

  sntp_set_timezone(timezone/3600);
  sntp_set_daylight(daylightOffset_sec);
  sntp_init();
}
```

   (1) timezone ของกรุงเทพเป็น GMT+7 จึงเท่ากับ 7 * 3600

   int timezone = 7 * 3600;

   ถ้าเป็น San Francisco TimeZone เป็น GMT-7

   int timezone = -7 * 3600;

   (2) daylightOffset_sec ค่า Date Swing Time เป็นวินาที

   (3) server สามารถกำหนด Server ได้ตั้งแต่ 1 ถึง 3 ที่ เช่นถ้าต้องการใช้ NTP Server ของไทยที่มีอยู่หลายแห่ง

Name: ntp.ku.ac.th Address: 158.108.212.149	Name: itoml.live.rmutt.ac.th Address: 203.158.111.32	Name: time1.nimt.or.th Address: 203.185.69.60
Name: fw.eng.ku.ac.th Address: 158.108.32.17	Name: delta.cpe.ku.ac.th Address: 158.108.32.3	Name: time2.nimt.or.th Address: 203.185.69.59
Name: ilm.live.rmutt.ac.th Address: 203.158.118.3	Name: time.navy.mi.th Address: 118.175.67.83	Name: time3.nimt.or.th Address: 203.185.69.56
Name: time.uni.net.th Address: 202.28.18.72	Name: clock.nectec.or.th Address: 202.44.204.114	

configTime(timezone, dst, "time.navy.mi.th");

2. โครงสร้างของตัวแปร tm มีดังนี้

```
struct tm {
  int tm_sec;     /* วินาที,  range 0 to 59        */
  int tm_min;     /* นาที, range 0 to 59           */
  int tm_hour;    /* ชั่วโมง, range 0 to 23        */
  int tm_mday;    /* วันที่, range 1 to 31  */
  int tm_mon;     /* เดือน, range 0 to 11          */
  int tm_year;    /* ปีคริสศักราช ตั้งแต่ 1900   */
  int tm_wday;    /* วัน, range 0 to 6    */
  int tm_yday;    /* วันใน 1 ปี, range 0 to 365  */
  int tm_isdst;   /* daylight saving time          */
};
```

การเรียกใช้ก็ตามตัวอย่างในโปรแกรม

งานมอบหมาย

ให้เขียนโปรแกรมแสดงค่า

1. วันที่ ชื่อเดือน ("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep", "Oct","Nov","Dec") ปี (เป็นปี พศ.)
2. เวลาเป็น ชม.นาที ตามเวลาในประเทศไทย