```
/*
 * Time_NTP.pde
 * Example showing time sync to NTP time source
 *
 * This sketch uses the ESP8266WiFi library
 */

//SCL as D1/GPIO5
//SDA as D2/GPIO4
#include <Wire.h>
#include<LiquidCrystal_I2C.h>

//#include "DHT.h"

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

#define PUMPPIN 16   //GPIO16 (D0)
#define VALVE1PIN 14  //GPIO14 (D5)
#define VALVE2PIN 12  //GPIO12 (D6)
#define VALVE3PIN 13  //GPIO13 (D7)
#define VALVE4PIN 15  //GPIO15 (D8)

/*
#define DHTPIN 14     // what pin we're connected to
GPIO14 (D5)
#define DHTTYPE DHT22   // DHT 22

#define DEBUG

#define DEBUG_PRINTER Serial

#ifdef DEBUG
```

```
#define DEBUG_PRINT(...) { DEBUG_PRINTER.
print(__VA_ARGS__); }
#define DEBUG_PRINTLN(...) { DEBUG_PRINTER.
println(__VA_ARGS__); }
#else
#define DEBUG_PRINT(...) {}
#define DEBUG_PRINTLN(...) {}
#endif

DHT *dht;
*/

const char ssid[] = "dragons";   // your network
SSID (name)
const char pass[] = "dragonsoffice"; // your network
password

// Initial I2C-LCD
// Address is 0x27 (for PCF8574) or 0x3F (for
PCF8574A)
// Type 16 characters 2 lines
LiquidCrystal_I2C lcd(0x3F, 16, 2);
//LiquidCrystal_I2C lcd(0x27, 16, 2);

//void initDht(DHT **dht, uint8_t pin, uint8_t
dht_type);
//void readDht(DHT *dht, float *temp, float *humid,
float *hic);

// NTP Servers:
const char ntpServerName[] = "2.th.pool.ntp.org"; //
Time server
// IPAddress timeServer(132, 163, 4, 101); //
time-a.timefreq.bldrdoc.gov
```

```
// IPAddress timeServer(132, 163, 4, 102); //
time-b.timefreq.bldrdoc.gov
// IPAddress timeServer(132, 163, 4, 103); //
time-c.timefreq.bldrdoc.gov

const char tzName[] = "GMT+7 Bangkok / Thailand /
Indo China Time";
const int timeZone = 7;     // Indo China Time
//const int timeZone = 1;     // Central European
Time
//const int timeZone = -5;  // Eastern Standard Time
(USA)
//const int timeZone = -4;  // Eastern Daylight Time
(USA)
//const int timeZone = -8;  // Pacific Standard Time
(USA)
//const int timeZone = -7;  // Pacific Daylight Time
(USA)


WiFiUDP Udp;
//unsigned int localPort = 8888;  // local port to
listen for UDP packets
uint16_t localPort;  // local port to listen for UDP
packets

int PumpState = LOW;
int Valve1State = LOW;
int Valve2State = LOW;
int Valve3State = LOW;
int Valve4State = LOW;
int timer = 0;

void setup()
```

```
{
  lcd.init(); // Start
  lcd.backlight(); // Enable LED backlight

  //initDht(&dht, DHTPIN, DHTTYPE);
//Set mode of GPIO
 pinMode(PUMPPIN, OUTPUT);
 pinMode(VALVE1PIN, OUTPUT);
 pinMode(VALVE2PIN, OUTPUT);
 pinMode(VALVE3PIN, OUTPUT);
 pinMode(VALVE4PIN, OUTPUT);

//Clear all output to "OFF" stage
 digitalWrite(PUMPPIN, PumpState);
 digitalWrite(VALVE1PIN, Valve1State);
 digitalWrite(VALVE2PIN, Valve2State);
 digitalWrite(VALVE3PIN, Valve3State);
 digitalWrite(VALVE4PIN, Valve4State);

 Serial.begin(115200);
  //while (!Serial) ; // Needed for Leonardo only
  delay(250);
 Serial.println();
  Serial.println(String("Connecting to ") + ssid);
 lcd.setCursor(0,0);
  lcd.print(String("SSID:") + ssid);
  WiFi.begin(ssid, pass);

  int count=0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    lcd.setCursor(0,1);
    lcd.print(count);
    count++;
    if (count > 30) { // If try to connect over 20
times will be restart itself
      ESP.restart();
    }
  }
  lcd.clear();
 Serial.println();
  Serial.println("IP number assigned by DHCP is " +
WiFi.localIP());

  // Seed random with values unique to this device
  uint8_t macAddr[6];
 WiFi.macAddress(macAddr);
  uint32_t seed1 =
    (macAddr[5] << 24) | (macAddr[4] << 16) |
    (macAddr[3] << 8)  | macAddr[2];
  randomSeed(WiFi.localIP() + seed1 + micros());
  localPort = random(1024, 65535);

 Serial.println("Starting UDP");
 Udp.begin(localPort);
 Serial.println("Local port: " + Udp.localPort());
 Serial.println("waiting for sync");
 setSyncProvider(getNtpTime);
  setSyncInterval(60 * 60); // sync every 1 hr
}

time_t prevDisplay = 0; // when the digital clock
was displayed
/*
time_t prevDHT = 0; // when the DHT was read
static float t_dht;
static float h_dht;
static float hic_dht;
*/

void loop()
{
/*
  if ((now() - prevDHT) >= 10) {
    prevDHT = now();
   readDht(dht, &t_dht, &h_dht, &hic_dht);
  }
*/
  PumpState = LOW;
  Valve1State = LOW;
  Valve2State = LOW;
  Valve3State = LOW;
  Valve4State = LOW;
  timer = (now() % 72) / 3; //Simulate timer 0..23
  //timer = hour();

 switch(timer){
    case 8:
      PumpState = HIGH;
      Valve1State = HIGH;
      break;
    case 11:
      PumpState = HIGH;
      Valve2State = HIGH;
      break;
    case 14:
      PumpState = HIGH;
      Valve3State = HIGH;
      break;
    case 17:
      PumpState = HIGH;
      Valve4State = HIGH;
      break;
  }
/*
  if (timer == 8) {
    PumpState = HIGH;
    Valve1State = HIGH;
  }
  if (timer == 11) {
    PumpState = HIGH;
    Valve2State = HIGH;
  }
  if (timer == 14) {
    PumpState = HIGH;
    Valve3State = HIGH;
  }
  if (timer == 17) {
    PumpState = HIGH;
    Valve4State = HIGH;
  }
*/
  if (timeStatus() != timeNotSet) {
    if (now() != prevDisplay) { //update the display
only if time has changed
      prevDisplay = now();
     digitalClockDisplay();
    }
  }
}

void digitalClockDisplay(){
  // digital clock display of the time
Serial.print(dayStr(weekday()));
  Serial.print(" ");
```

```
 Serial.print(day());
  Serial.print(" ");
Serial.print(monthStr(month()));
  Serial.print(" ");
 Serial.print(year());
  Serial.print(" ");
 Serial.print(hour());
 printDigits(minute());
 printDigits(second());
 Serial.println();

 //lcd.clear();
  lcd.setCursor(0,0); // Set home cursor
lcd.print(dayShortStr(weekday()));
  lcd.print(" ");
lcd.print(monthShortStr(month()));
  lcd.print(" ");
 lcd.print(day());
  lcd.print(" ");
 lcd.print(hour());
  if ((second() % 2) == 0) {
    lcd.print(":");
  } else {
    lcd.print(" ");
  }
 printDigitsLCD(minute());
  lcd.print("  ");
/*
 lcd.setCursor(0,1);
 lcd.print(h_dht);
 lcd.setCursor(4,1);
  lcd.print("% ");
 lcd.print(t_dht);
 lcd.setCursor(10,1);
 lcd.print(">");
 lcd.print(hic_dht);
 lcd.setCursor(15,1);
 lcd.print("C");
*/
 //lcd.setCursor(14,1);
 //printDigitsLCD(second());

 lcd.setCursor(3,1);
 lcd.print(PumpState);
 lcd.print(Valve1State);
 lcd.print(Valve2State);
 lcd.print(Valve3State);
 lcd.print(Valve4State);
  lcd.print(" ");
 printDigitsLCD(timer);

 digitalWrite(PUMPPIN, PumpState);
 digitalWrite(VALVE1PIN, Valve1State);
 digitalWrite(VALVE2PIN, Valve2State);
 digitalWrite(VALVE3PIN, Valve3State);
 digitalWrite(VALVE4PIN, Valve4State);
}



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);
}
```

```
void printDigitsLCD(int digits){
  // utility for digital clock LCD display: prints
preceding colon and leading 0
 //lcd.print(":");
  if(digits < 10)
    lcd.print('0');
 lcd.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_tgetNtpTime()
{
  IPAddress timeServerIP; // NTP server address
  while (Udp.parsePacket() > 0) ; // discard any
previously received packets
 Serial.print("Transmit NTP Request");
  //get a random server from the pool
 WiFi.hostByName(ntpServerName, timeServerIP);
  Serial.print(" to ");
 Serial.print(timeServerIP);
  Serial.println(" .");
 sendNTPpacket(timeServerIP);
  uint32_t beginWait = millis();
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket();
    if (size >= NTP_PACKET_SIZE) {
     Serial.println("Received NTP Response.");
     Serial.print("TimeZone: ");
     Serial.println(tzName);
     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 :-(");
  lcd.clear();
 lcd.setCursor(0,0);
 lcd.print(timeServerIP);
 lcd.setCursor(0,1);
  lcd.print("No NTP Response!");
  delay(3000);
 ESP.restart();
  return 0; // return 0 if unable to get the time
}

// send an NTP request to the time server at the
given address
voidsendNTPpacket(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();
}

/*
void initDht(DHT **dht, uint8_t pin, uint8_t
dht_type) {
    // Connect pin 1 (on the left) of the sensor to
+5V
    // NOTE: If using a board with 3.3V logic like
an Arduino Due connect pin 1
    // to 3.3V instead of 5V!
    // Connect pin 2 of the sensor to whatever your
DHTPIN is
```

```
    // Connect pin 4 (on the right) of the sensor to
GROUND
    // Connect a 10K resistor from pin 2 (data) to
pin 1 (power) of the sensor

    // Initialize DHT sensor for normal 16mhz Arduino
    // NOTE: For working with a faster chip, like an
Arduino Due or Teensy, you
    // might need to increase the threshold for
cycle counts considered a 1 or 0.
    // You can do this by passing a 3rd parameter
for this threshold.  It's a bit
    // of fiddling to find the right value, but in
general the faster the CPU the
    // higher the value.  The default for a 16mhz
AVR is a value of 6.  For an
    // Arduino Due that runs at 84mhz a value of 30
works.
    // Example to initialize DHT sensor for Arduino
Due:
    //DHT dht(DHTPIN, DHTTYPE, 30);

    *dht = new DHT(pin, dht_type, 30);
    (*dht)->begin();
    DEBUG_PRINTLN(F("DHTxx test!"))  ;
}

void readDht(DHT *dht, float *temp, float *humid,
float *hic) {

    if (dht == NULL) {
        DEBUG_PRINTLN(F("[DHTxx] is not initialised.
please call initDht() first."));
        return;
    }

    // Reading temperature or humidity takes about
250 milliseconds!
    // Sensor readings may also be up to 2 seconds
'old' (its a very slow sensor)
    float h = dht->readHumidity();

    // Read temperature as Celsius
    float t = dht->readTemperature();
    // Read temperature as Fahrenheit
    float f = dht->readTemperature(true);

    // Check if any reads failed and exit early (to
try again).
    if (isnan(h) || isnan(t) || isnan(f)) {
        DEBUG_PRINTLN("Failed to read from DHT
sensor!");
        return;
    }

    // Compute heat index
    // Must send in temp in Fahrenheit!
    float hi = dht->computeHeatIndex(f, h);

    DEBUG_PRINT("[Humidity: ");
    DEBUG_PRINT(h);
    DEBUG_PRINT(" Percent]\t");
    DEBUG_PRINT("[Temperature: ");
    DEBUG_PRINT(t);
    DEBUG_PRINT(" Celsius, ");
    DEBUG_PRINT(f);
    DEBUG_PRINT(" Fahrenheit]\t");
    DEBUG_PRINT("[Heat index: ");
    DEBUG_PRINT(hi);
    DEBUG_PRINTLN(" Fahrenheit]");

    *temp = t;
    *humid = h;
    *hic = (hi-32)*5/9;
}

*/
```