МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 6**

**по дисциплине «Операционные системы»**

**Тема: «Сетевые компоненты на примере создания приложения для синхронизации точного времени»**

Выполнил: Сидоров Д. С., группа ИТС-123

Проверил: к.т.н., доц. Семёнов А. А.

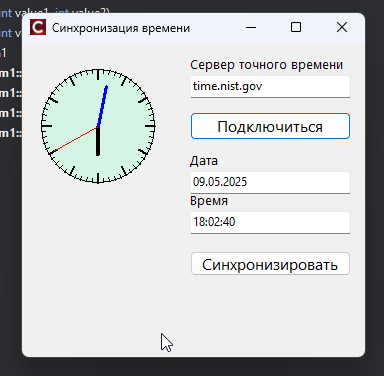
Москва, 2025г

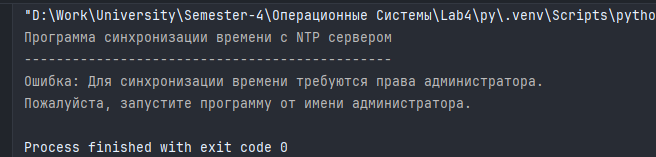
**Сетевые компоненты на примере создания приложения для синхронизации точного времени**

1. Создать приложение, позволяющие подключиться к серверу точного времени, получить точное время и синхронизировать его с часами компьютера. Графический интерфейс приложения должен выглядеть следующим образом.
2. Оснастить приложение аналоговыми часами, которые появляются на форме и начинают ходить после нажатия кнопки Синхронизировать.
3. Дополнительно необходимо:

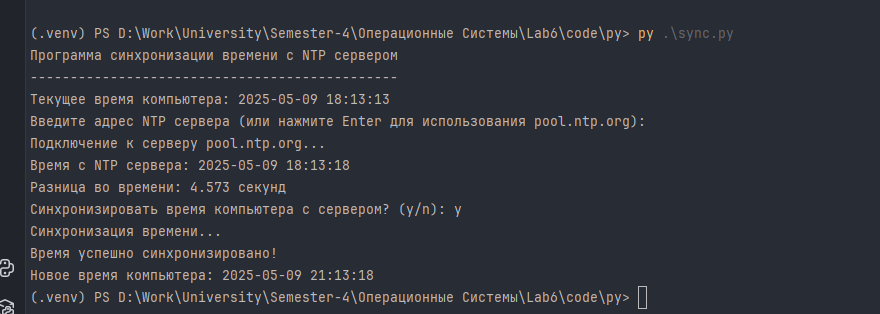
1) реализовать консольную программу (можно на Python) или скрипт (PS или bash), в которой будет подключение к серверу точного времени и синхронизация его с часами на компьютере;

2) в программе с ГИП сделать фон для часов и исправить ошибку в позиционировании часовой стрелки.

  
*Рисунок 1. Работающее GUI приложение*



*Рисунок 2. Пример ошибки в приложении на питоне*



*Рисунок 3. Работающее приложение из п. 2.*

**Листинг кода**

**Unit1.cpp**//---------------------------------------------------------------------------

#include <vcl.h>

#include <IdTime.hpp>

#include <SysUtils.hpp>

#include <DateUtils.hpp>

#include <windows.h>

#include <math.h>

#pragma hdrstop

#include "Unit1.h"

#ifdef \_\_cplusplus

int min (int value1, int value2);

int min(int value1, int value2)

{

return ( (value1 < value2) ? value1 : value2);

}

#endif

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::btnConnectClick(TObject \*Sender)

{

try {

IdTime1->Host = editNTP->Text;

TDateTime currentTime = IdTime1->DateTime;

editDate->Text = DateToStr(currentTime);

editTime->Text = TimeToStr(currentTime);

IdTime1->Disconnect();

} catch (Exception &e) {

ShowMessage(L"Ошибка поключения: " + e.Message);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::btnSynchronizeClick(TObject \*Sender)

{

try {

SYSTEMTIME st;

TDateTime localTime = StrToDateTime(editDate->Text + " " + editTime->Text);

TDateTime utcTime = TTimeZone::Local->ToUniversalTime(localTime);

unsigned short year, month, day;

utcTime.DecodeDate(&year, &month, &day);

unsigned short hour, minute, second, msec;

utcTime.DecodeTime(&hour, &minute, &second, &msec);

st.wYear = year;

st.wMonth = month;

st.wDay = day;

st.wHour = hour;

st.wMinute = minute;

st.wSecond = second;

st.wMilliseconds = 0;

if (SetSystemTime(&st)) {

ShowMessage(L"Точные дата и время синхронизированы.");

} else {

DWORD err = GetLastError();

ShowMessage(L"Ошибка при синхронизации времени." + String(err));

}

} catch (Exception &e) {

ShowMessage(L"Ошибка синхронизации: " + e.Message);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender)

{

TCanvas \*Canvas = Image1->Canvas;

int w = Image1->Width;

int h = Image1->Height;

int centerX = w / 2;

int centerY = h / 2;

int radius = min(w, h) / 2 - 10;

TDateTime currentTime = Now();

Word hour, minute, second, millisecond;

DecodeTime(currentTime, hour, minute, second, millisecond);

int hours = hour % 12;

int minutes = minute;

int seconds = second;

Canvas->Brush->Color = Form1->Color;

Canvas->FillRect(Rect(0, 0, w, h));

Canvas->Pen->Color = clBlack;

Canvas->Brush->Color = (TColor)RGB(213, 245, 228);

Canvas->Ellipse(centerX - radius, centerY - radius, centerX + radius, centerY + radius);

//Засечки

for (int i = 0; i < 60; ++i) {

double angle = i \* 6.0 \* M\_PI / 180.0;

int outerX = centerX + radius \* cos(angle - M\_PI\_2);

int outerY = centerY + radius \* sin(angle - M\_PI\_2);

int innerRadius = (i % 5 == 0) ? radius - 10 : radius - 5;

int innerX = centerX + innerRadius \* cos(angle - M\_PI\_2);

int innerY = centerY + innerRadius \* sin(angle - M\_PI\_2);

Canvas->Pen->Width = (i % 5 == 0) ? 2 : 1;

Canvas->Pen->Color = clBlack;

Canvas->MoveTo(outerX, outerY);

Canvas->LineTo(innerX, innerY);

}

// Углы для стрелок

double angleHour = (hours + minutes / 60.0) \* 30.0 \* M\_PI / 180.0;

double angleMinute = minutes \* 6.0 \* M\_PI / 180.0;

double angleSecond = seconds \* 6.0 \* M\_PI / 180.0;

// Стрелка часов

Canvas->Pen->Color = clBlack;

Canvas->Pen->Width = 4;

Canvas->MoveTo(centerX, centerY);

Canvas->LineTo(centerX + radius \* 0.5 \* cos(angleHour - M\_PI\_2),

centerY + radius \* 0.5 \* sin(angleHour - M\_PI\_2));

// Стрелка минут

Canvas->Pen->Color = clBlue;

Canvas->Pen->Width = 3;

Canvas->MoveTo(centerX, centerY);

Canvas->LineTo(centerX + radius \* 0.7 \* cos(angleMinute - M\_PI\_2),

centerY + radius \* 0.7 \* sin(angleMinute - M\_PI\_2));

// Стрелка секунд

Canvas->Pen->Color = clRed;

Canvas->Pen->Width = 1;

Canvas->MoveTo(centerX, centerY);

Canvas->LineTo(centerX + radius \* 0.9 \* cos(angleSecond - M\_PI\_2),

centerY + radius \* 0.9 \* sin(angleSecond - M\_PI\_2));

}

//---------------------------------------------------------------------------

**Unit1.h**//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <IdBaseComponent.hpp>

#include <IdComponent.hpp>

#include <IdTCPClient.hpp>

#include <IdTCPConnection.hpp>

#include <IdTime.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Mask.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TButton \*btnConnect;

TButton \*btnSynchronize;

TIdTime \*IdTime1;

TImage \*Image1;

TTimer \*Timer1;

TLabeledEdit \*editNTP;

TLabeledEdit \*editDate;

TLabeledEdit \*editTime;

void \_\_fastcall btnConnectClick(TObject \*Sender);

void \_\_fastcall btnSynchronizeClick(TObject \*Sender);

void \_\_fastcall Timer1Timer(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**2.py**//---------------------------------------------------------------------------

import socket

import struct

import time

import sys

import ctypes

import datetime

import win32api

def get\_ntp\_time(server="pool.ntp.org", port=123):

"""

Получение точного времени с NTP сервера

"""

# NTP запрос происходит в виде 48-байтного пакета

client = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

client.settimeout(5.0)

# Формируем NTP пакет

# Первый байт включает в себя версию и режим

# Версия 3, режим клиента = 3

packet = b'\x1b' + 47 \* b'\0'

try:

# Отправляем запрос

client.sendto(packet, (server, port))

# Получаем ответ

packet, \_ = client.recvfrom(1024)

except socket.timeout:

print(f"Ошибка: Время ожидания ответа от {server} истекло.")

return None

except socket.gaierror:

print(f"Ошибка: Не удалось разрешить имя сервера {server}.")

return None

except Exception as e:

print(f"Ошибка при получении времени: {e}")

return None

finally:

client.close()

if not packet:

return None

# Распаковываем ответ: нас интересуют байты с 40-го по 43-й,

# которые представляют собой timestamp передачи пакета сервером

seconds = struct.unpack('!12I', packet)[10]

# NTP использует эпоху, которая начинается с 1900-01-01

# Unix время начинается с 1970-01-01, поэтому мы вычитаем разницу

seconds -= 2208988800

return seconds

def set\_system\_time(seconds):

"""

Установка системного времени с помощью pywin32

"""

try:

# Конвертация Unix timestamp в локальное время

local\_time = time.localtime(seconds)

# Создаем структуру SYSTEMTIME для win32api

system\_time = win32api.GetSystemTime()

# Обновляем время

new\_time = (

local\_time.tm\_year,

local\_time.tm\_mon,

0, # Не используется в SYSTEMTIME

local\_time.tm\_mday,

local\_time.tm\_hour,

local\_time.tm\_min,

local\_time.tm\_sec,

0 # Миллисекунды

)

# Устанавливаем системное время

win32api.SetSystemTime(\*new\_time)

return True

except Exception as e:

print(f"Ошибка при установке системного времени: {e}")

return False

def check\_admin\_privileges():

"""

Проверка прав администратора

"""

try:

return ctypes.windll.shell32.IsUserAnAdmin() != 0

except:

return False

def main():

"""

Основная функция программы

"""

print("Программа синхронизации времени с NTP сервером")

print("----------------------------------------------")

# Проверка прав администратора

if not check\_admin\_privileges():

print("Ошибка: Для синхронизации времени требуются права администратора.")

print("Пожалуйста, запустите программу от имени администратора.")

return

# Получение текущего времени компьютера

current\_time = time.time()

local\_time = time.localtime(current\_time)

print(f"Текущее время компьютера: {time.strftime('%Y-%m-%d %H:%M:%S', local\_time)}")

# Выбор сервера NTP

server = input("Введите адрес NTP сервера (или нажмите Enter для использования pool.ntp.org): ")

if not server:

server = "pool.ntp.org"

print(f"Подключение к серверу {server}...")

# Получение времени с NTP сервера

ntp\_seconds = get\_ntp\_time(server)

if ntp\_seconds is None:

print("Не удалось получить время с NTP сервера.")

return

ntp\_time = time.localtime(ntp\_seconds)

print(f"Время с NTP сервера: {time.strftime('%Y-%m-%d %H:%M:%S', ntp\_time)}")

# Вычисление разницы во времени

time\_diff = abs(current\_time - ntp\_seconds)

print(f"Разница во времени: {time\_diff:.3f} секунд")

# Синхронизация времени

prompt = input("Синхронизировать время компьютера с сервером? (y/n): ")

if prompt.lower() == 'y':

print("Синхронизация времени...")

if set\_system\_time(ntp\_seconds):

print("Время успешно синхронизировано!")

# Показываем обновленное время

updated\_time = time.localtime(time.time())

print(f"Новое время компьютера: {time.strftime('%Y-%m-%d %H:%M:%S', updated\_time)}")

else:

print("Не удалось синхронизировать время.")

else:

print("Синхронизация отменена.")

main()

**Вывод:** В ходе выполнения лабораторной работы было создано приложение для синхронизации системного времени компьютера с серверами точного времени с использованием сетевых компонентов. Основное приложение разработано с графическим интерфейсом, который позволяет пользователю выбрать сервер NTP, получить и отобразить точное время, а также выполнить синхронизацию системных часов. Дополнительно реализованы аналоговые часы, которые начинают работать после синхронизации, включая корректное позиционирование часовой стрелки и улучшенный фоновый дизайн. Параллельно создана консольная программа на Python с использованием библиотеки pywin32, предоставляющая аналогичный функционал для синхронизации времени в текстовом режиме. Работа наглядно демонстрирует применение сетевых компонентов для создания практически полезных приложений с разными интерфейсами.