МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Базовая кафедра №234 — Управляющих ЭВМ

Отчет по практической работе

Выполнил:
Студент группы ИКМО-05-23 Миронов Д.С.

Проверил: Плужнова Т.С.

Москва, 2024

Ссылка на архив: https://disk.yandex.ru/d/ea1iSB_iFe1XSg

В варианте 20 задана конфигурация сервера OPC UA. Реализовать заданную конфигурацию сервера OPC UA. Динамически эмулировать сигналы. Организовать запись и сохранение необходимых сигналов.

Имена сигналов могут быть любыми. Тип сигналов указан в варианте.

Условные обозначения:

- R – чтение;
- W – запись;
- H – архивирование;

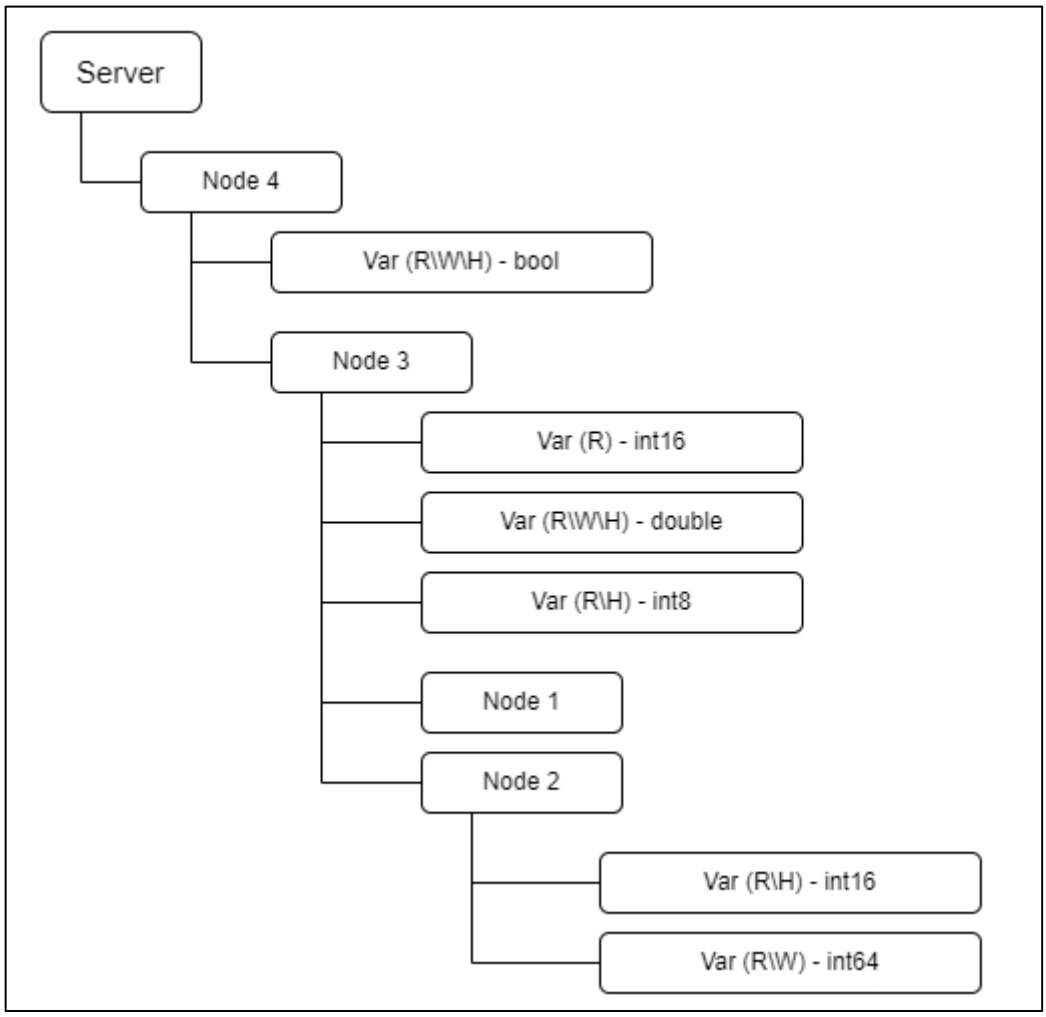

Рисунок 1 – 20 вариант работы

1. Инициализация

Необходимо инициализировать OPC UA сервер по адресу opc.tcp://0.0.0.0:4840/freeopcua/server/

Листинг 1

```
server = Server()
server.set_endpoint(URL)
server.set_server_name("OPC UA Server Example")
```

2. Структура

Создаем структуру узлов

2.1. Node 4

Листинг 2

```
node4 = root_node.add_object("ns=2;i=1001", "Node4")
var_rw_bool=node4.add_variable("ns=2;i=2001","Var_RW_H", False, ua.VariantType.Boolean)
var_rw_bool.set_writable()
```

- Узел Node4 добавляется в корневую структуру (root_node) с уникальным NodeId.
- Переменная Var_RW_H типа Boolean добавляется как дочерний узел.
- Метод set_writable() делает переменную доступной для записи.

2.2. Node 3

Листинг 3

```
node3 = root_node.add_object("ns=2;i=1002", "Node3")
var_r_int16 = node3.add_variable("ns=2;i=2002", "Var_R", 0, ua.VariantType.Int16)
var_rwh_double=node3.add_variable("ns=2;i=2003","Var_RW_H",0.0, ua.VariantType.Double)
var_rh_int8 = node3.add_variable("ns=2;i=2004", "Var_RH", 0, ua.VariantType.Byte)
var_r_int16.set_read_only()
var_rwh_double.set_writable()
```

- Узел Node3 содержит три переменные:
  - **Var_R**: Только для чтения (Int16).
  - **Var_RW_H**: Доступна для записи и архивирования (Double).
  - **Var_RH**: Только для чтения и архивирования (Byte).

2.3. Node 1 -> Node 2

Листинг 4

```
node1 = root_node.add_object("ns=2;i=1003", "Node1")
node2 = node1.add_object("ns=2;i=1004", "Node2")
var_rh_int16 = node2.add_variable("ns=2;i=2005", "Var_RH", 0, ua.VariantType.Int16)
var_rw_int64 = node2.add_variable("ns=2;i=2006", "Var_RW", 0, ua.VariantType.Int64)
var_rh_int16.set_read_only()
var_rw_int64.set_writable()
```

- Узел Node1 содержит вложенный узел Node2.
- В Node2 добавлены:
    - **Var_RH**: Только для чтения и архивирования (Int16).
    - **Var_RW**: Доступна для записи (Int64).

3. Обновление переменных

Функция обновления переменных, в которой происходит генерация случайных значений для переменных, при помощи библиотеки random.

Значения переменных, указанных для архивирования (Var_RW_H, Var_RW_H_double, Var_RH_int8, Var_RH_int16), сохраняются в таблицу базы данных вместе с текущим временем.

Листинг 5

```
# Функция обновления переменных
def update_signals():
    # Генерация случайных значений для переменных
    var_rw_bool.set_value(random.choice([True, False]))
    var_r_int16.set_value(random.randint(-32768, 32767))
    var_rwh_double.set_value(random.uniform(0, 100))
    var_rh_int8.set_value(random.randint(0, 255))  # Byte имеет диапазон от 0 до 255
    var_rh_int16.set_value(random.randint(-32768, 32767))
    var_rw_int64.set_value(random.randint(-9223372036854775808, 9223372036854775807))

    # Сохранение в базу данных (архивируемые переменные)
    timestamp = datetime.now()

    DB.query("INSERT INTO signals (timestamp, signal_name, value) VALUES (?, ?, ?)",
             (timestamp, "Var_RW_H", var_rw_bool.get_value()))

    DB.query("INSERT INTO signals (timestamp, signal_name, value) VALUES (?, ?, ?)",
             (timestamp, "Var_RW_H_double", var_rwh_double.get_value()))

    DB.query("INSERT INTO signals (timestamp, signal_name, value) VALUES (?, ?, ?)",
             (timestamp, "Var_RH_int8", var_rh_int8.get_value()))

    DB.query("INSERT INTO signals (timestamp, signal_name, value) VALUES (?, ?, ?)",
             (timestamp, "Var_RH_int16", var_rh_int16.get_value()))
```

4. Запуск сервера

Листинг 6

```
server.start()
print("Server started at {}".format(server.endpoint))
```

Сервер запускается. Клиенты могут подключиться по указанному адресу.

5. Основной цикл

Листинг 7

```
try:
    while True:
        update_signals()
        time.sleep(1)
except KeyboardInterrupt:
    print("Stopping server...")
finally:
    server.stop()
    conn.close()
```

В бесконечном цикле обновляются значения переменных каждую секунду.

При завершении программы сервер корректно останавливается, а соединение с базой данных закрывается.