УДК 004.94
Бортник К.Я., Прокопюк М.І.
Луцький національний технічний університет

БЕЗПРОВІДНА СИСТЕМА ВИКЛИКУ НА ОСНОВІ ELMES RD448

Бортник К.Я., Прокопюк М.І. Безпровідна система виклику на основі Elmes RD448. У даній статті висвітлено актуальність розробки безпровідних систем виклику. Розглянуто спосіб вирішення даної проблеми за допомогою радіоприймача Elmes RD448 і архітектури додатків DataSnap.
Ключові слова: алгоритм, Elmes, DataSnap, Android.

Бортник К.Я, Прокопюк М.И. Беспроводная система вызова на основе Elmes RD448. В данной статье освещены актуальность разработки беспроводных систем вызова. Рассмотрен способ решения данной проблемы с помощью радиоприемника Elmes RD448 и архитектуры приложений DataSnap.
Ключевые слова: алгоритм, Elmes, DataSnap, Android.

Bortnyk K., Prokopyuk M. Wireless call system based on Elmes RD448. In this article highlights the relevance of the development of wireless call. The way to solve this problem by using radio Elmes RD448 and architectural applications DataSnap.
Keywords: algorithm, Elmes, DataSnap, Android.

В основі даної системи лежить приймач контролю доступу з PIN-кодовим захистом і світлодіодним дисплеєм Elmes RD-448. Приймач розроблений для дистанційного керування і систем контролю доступу, що працюють з великою кількістю брелків передавачів і вимагають високого рівня безпеки. Для реалізації зазначеного режиму роботи приймач має наступні технічні характеристики:

- Пам'ять на 448 (модель RD 1 448) або 1000 (модель RD1-1000) користувачів брелків-передавачів;
- Доступ до захисту програмування приймача редагованим користувачем PIN-кодом з чотирьох цифр;
- Можливість видалення одного брелка-передавача без необхідності повного очищення пам'яті;
- Система ідентифікації користувачів за допомогою високозащіщённого алгоритму кодування KEELOQ® з плаваючим динамічним кодом;
- Тризначна цифрова панель для відображення номера користувача, спрощення програмування, призначених для користувача налаштувань і видалення;
- Високочутливий супергетеродинний приймач, що забезпечує контроль рівня сигналу;
- Енергонезалежний журнал на 6144 останніх події (тільки модель RD 1-448), включаючи дату, номер користувача і кнопку брелка;
- Тампер захисту від злому;
- Широкий діапазон напруги живлення: 10 ... 35VDC або 24VAC.

Приймач здатний запам'ятовувати останні 6144 події. Кожен раз, коли від брелка-передавача надходить команда, в журналі подій реєструється його номер (0 ... 447), інформація про кнопку (0 ... 1) брелка-передавача і поточний час. Вміст журналу подій приймача можна прочитати / роздрукувати за допомогою зовнішнього персонального комп'ютера (ПК). Зв'язок приймача з ПК здійснюється за допомогою кабелю перетворення напруги RS232 (+ 12 / -12V) <-> TTL (0 / 5V), підключеного до послідовній шині RS232 ПК. Схематичне зображення кабелю представлено нижче в інструкції. Кабель можна замовити у постачальника приймача. Спеціальне програмне забезпечення англійською мовою "RD reader", що підтримує читання даних, можна безкоштовно скачати з сайту виробника www.elmes.pl. Програма дозволяє читати і зберігати хронологічний журнал подій на ПК в форматі файлу Microsoft Acces® (* .mdb). Подальша обробка і представлення змісту даних файлу * .mdb здійснюється за допомогою комерційних програм Microsoft Acces®, Lotus Approach® або Open Office®.

Як згадувалося раніше, показання журналу подій можна вважати за допомогою спеціальної програми ПК. Крім того, завдяки спеціальному режиму роботи мікропроцесора, що досягається замиканням його 5-го і 6 го контактів , RD приймач можна використовувати для побудови локальної системи контролю і обмеження доступу користувачів. У цьому випадку номер брелка-

передавача кожної прийнятої команди стає доступним на послідовному виході TX. Якщо протягом 100 мілісекунд логічний 0 ні поданий на послідовний вхід RX, релейний вихід приймача залишиться вимкненим. Ця особливість спрощує розробку системи контролю доступу, в якій можна тимчасово або постійно обмежити доступ на території, що охороняється одному або декільком користувачам брелків.

Для спрощення і зменшення величини програмного коду використовується база даних на основі вільної реляційної системи керування базами даних Firebird. Для під'єднання Android девайсів до бази даних використано так звану архітектуру додатків DataSnap. Багатоланкова архітектура додатків баз даних розроблена з необхідністю обробляти на стороні сервера запити від великого числа віддалених клієнтів. Здавалося, з цим завданням цілком можуть впоратися і клієнт/серверні додатки, проте в цьому випадку при великому числі клієнтів вся обчислювальна навантаження лягає на сервер БД, який володіє досить мізерним набором засобів для реалізації складної бізнес-логіки (збережені процедури, тригери, перегляди і т.д.). І розробники змушені істотно ускладнювати програмний код клієнтського ПЗ, а це вкрай небажано при наявності великого Числа віддалених клієнтських комп'ютерів.

Багатоланкова архітектура додатків БД покликана виправити перераховані недоліки.

В рамках даної архітектури "тонкі" клієнти представляють собою найпростіші додатки, що забезпечують лише передачу даних, їх локальне кешування, представлення засобами призначеного для користувача інтерфейсу, редагування і найпростішу обробку.

Клієнтські програми звертаються не до сервера БД безпосередньо, а до спеціалізованого ПЗ проміжного шару. Це може бути і одна ланка (найпростіша триланкова модель) і більш складна структура.

ПО проміжного шару називається сервером додатків, приймає запити клієнтів, обробляє їх відповідно до запрограмованих правил бізнес-логіки, при необхідності перетворює в форму, зручну для сервера БД і відправляє серверу.

Сервер додатків взаємодіє з сервером БД, використовуючи одну з технологій доступу до даних, реалізованих в Delphi, C++. Це технології ADO, BDE, InterBase Express і dbExpress. Дистанційні клієнтські програми створюються з використанням спеціального набору компонентів, об'єднаних загальною назвою DataSnap. Ці компоненти інкапсулюють стандартні транспорти (DCOM, HTTP, сокети) і забезпечують з'єднання віддаленого клієнтського додатку з сервером додатка. Також компоненти DataSnap забезпечують доступ клієнта до функцій сервера додатків за рахунок використання інтерфейсу AppServer.

Інтерфейс IAppServer є основною механізму віддаленого доступу клієнтських додатків до сервера додатка. Набір даних клієнта використовує його для спілкування з компонентом-провайдером на сервері додатка. Набори даних клієнта отримують примірник IAppServer від компонента сполуки в клієнтському додатку.

При створенні віддалених модулів даних кожному такому модулю ставиться у відповідність новостворюваний інтерфейс, предком якого є інтерфейс IAppServer.

Розробник може додати до нового інтерфейсу власні методи, які, завдяки можливостям механізму віддаленого доступу багатоланкових додатків, стають доступні додатку-клієнту.

Так як часто клієнтські комп'ютери - це досить слабкі машини, реалізація складної бізнес-логіки на сторону сервера дозволяє істотно підвищити швидкодію системи вцілому. І не тільки за рахунок більш потужної техніки, а й за рахунок оптимізації виконання однорідних запитів користувачів.

Наприклад, при надмірному завантаженні сервера, сервер додатків може самостійно обробляти запити користувачів (ставити їх в чергу або скасовувати) без додаткового завантаження сервера БД.

Наявність сервера додатків підвищує безпеку системи, оскільки це надає змогу організувати авторизацію користувачів, так і будь-які інші функції безпеки без прямого доступу до даних.

Клієнтська програма в триланковій моделі має володіти лише мінімально необхідним набором функцій, делегуючи більшість операцій з обробки даних сервера додатків.

В першу чергу віддалений клієнтський додаток повинен забезпечити з'єднання з сервером додатків. Для цього використовуються компоненти з'єднань DataSnap:

- TDCOMConnection - використовує DCOM;

- TSocketconnection - використовує сокети Windows;
- TWebConnection - використовує HTTP.

Компоненти з'єднання DataSnap надають інтерфейс IAppServer, використовуваний компонентами-провайдерами на стороні сервера і компонентами TClientDataSet на стороні клієнта для передачі пакетів даних.

Для роботи з наборами даних використовуються компоненти TClientDataSet, що працюють в режимі кешування даних.

За своєю структурою клієнтська програма виглядає, як звичайний додаток баз даних (рис.1).

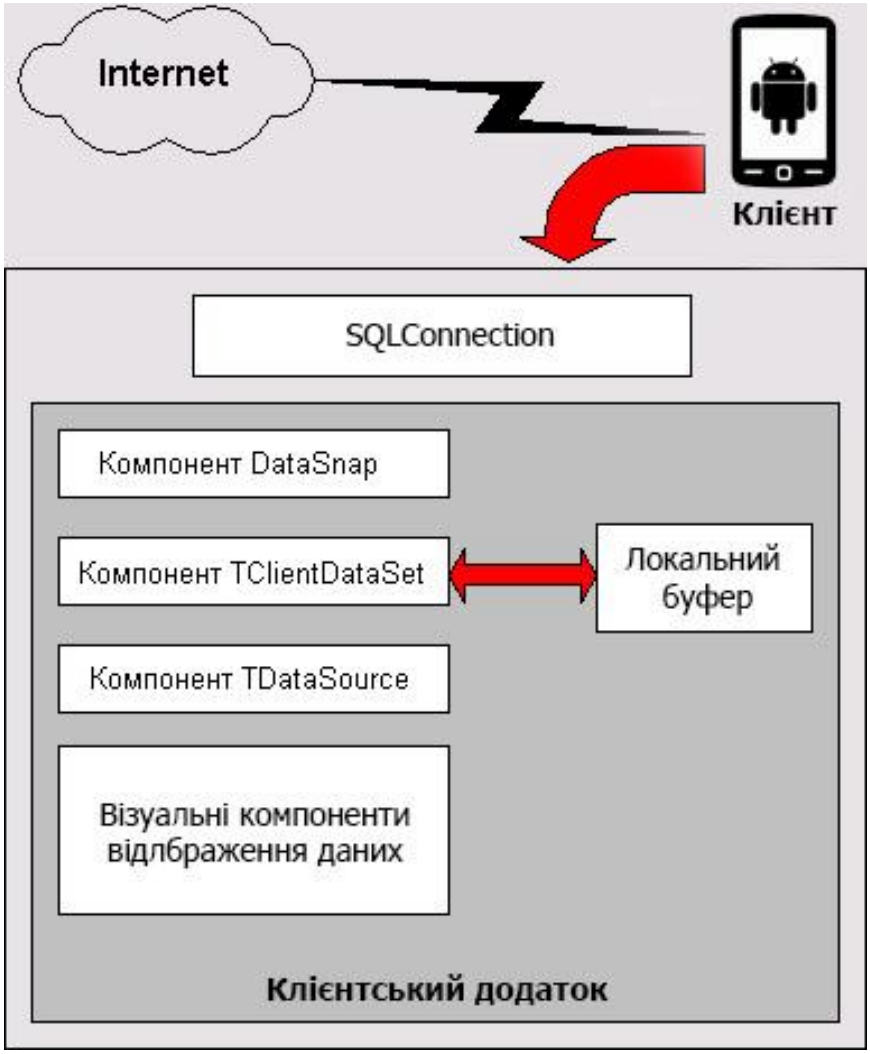

Рисунок 1. Структура клієнтської програми

З'єднання клієнта з сервером додатків здійснюється спеціалізованими компонентами DataSnap. Ці компоненти взаємодіють з віддаленим модулем даних, що входять до складу сервера, за допомогою методів інтерфейсу IAppServer.

Також в клієнтському додатку можуть використовуватися додаткові методи інтерфейсу віддаленого модуля даних, успадкованого від інтерфейсу IAppServer.

Як і звичайна програма БД, клієнт багатоланкового розподіленого додатка повинен містити компоненти, інкапсулюючий набір даних, які пов'язані з візуальними компонентами відображення даних за допомогою компонентів TDataSource.

Очевидно, що набір даних сервера повинен бути скопійований клієнтським додатком в якийсь локальний буфер. При цьому повинен використовуватися ефективний механізм завантаження даних порівняно невеликими порціями, що дозволяє значно розвантажити транспортний канал між клієнтом і сервером додатків.

Кешування і редагування даних в клієнтському додатку забезпечує спеціалізований компонент TclientDataSet, віддаленим предком якого є клас TDataSet (рис.2). Крім успадкованих від предків методів, клас TclientDataSet інкапсулює ряд додаткових функцій, що полегшують керування даними.

Подібно до звичайного додатком БД, в "тонкому" клієнті для розміщення невізуальних компонентів доступу до даних необхідно використовувати модулі даних.

Для отримання набору даних сервера компонент TclientDataSet взаємодіє з компонентом TDataSetProvider, використовуючи методи інтерфейсу IProviderSupport

По суті все унікальні функції клієнтського додатка зосереджені в компоненті. Клієнтську програму не відрізняється від звичайного застосування БД та при його розробці можуть застосовуватися стандартні методи.

У палітрі компонентів Delphi, C++ представлено кілька компонентів, інкапсулюючих клієнтський набір даних. У той же час при розробленні цих віддалених клієнтських додатків застосовується компонент TClientDataSet. Внесемо ясність у це питання. Отже, крім компонента TClientDataSet, розташованого на сторінці Data Access, існують ще два компоненти:

- TSimpleDataSet - розроблений для технології доступу до даних dbExpress і, по суті, є єдиним повноцінним засобом для роботи з набором даних в рамках цієї технології;
- TiBdientDataSet - використовується в технології доступу до даних сервера InterBase - InterBase Express.

Всі перераховані компоненти походять від спільного предка - класу TCustomClientDataSet. Вони забезпечують локальне кешування даних і взаємодію з серверним набором даних за допомогою інтерфейсу IProviderSupport.

Основна відмінність між компонентом TClientDataSet і іншими клієнтськими компонентами полягає в тому, що перший призначений для використання з зовнішнім компонентом-провайдером даних. А значить, він може взаємодіяти з віддаленим провайдером даних.

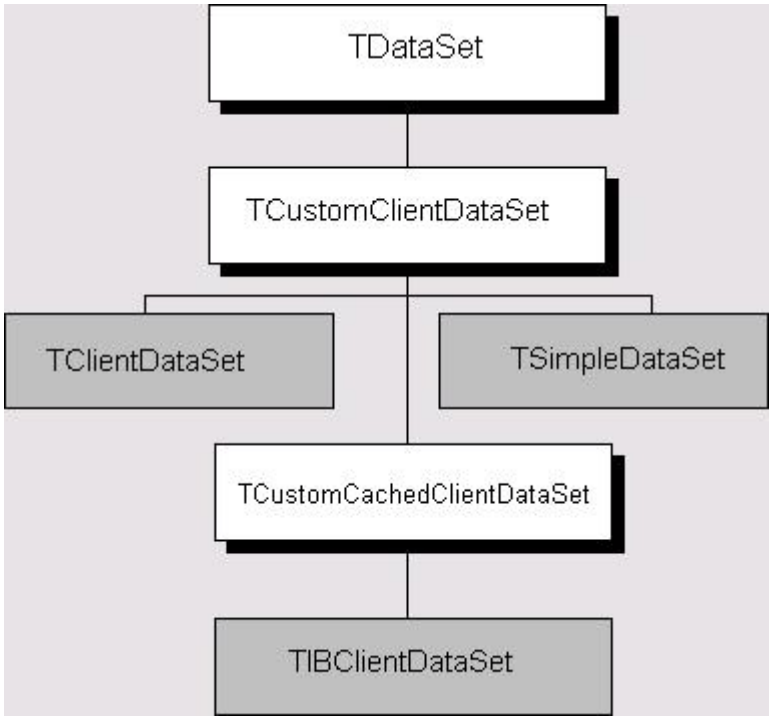

Рисунок 2. Ієрархія класів клінтських наборів даних

Решта перераховані компоненти інкапсулюють внутрішній провайдер даних, надаючи тим самим для використання в рамках відповідних технологій доступу до даних ефективний механізм локального кешування даних. Використання внутрішнього провайдера даних забезпечує загальний клас-предок TCustomCachedDataSet.

Для цього він має захищене властивість property Provider: TDataSetProvider;

З'єднання з джерелом даних здійснюється не властивістю RemoteServer задає віддалений сервер, а стандартними засобами відповідної технології доступу до даних.

Таким чином, для роботи з віддаленими даними (т. Е. Зовнішніми по відношенню до клієнта) придатний тільки компонент TclientDataSet, вміє працювати з зовнішнім провайдером даних.

Сервер додатків інкапсулює більшу частину бізнес-логіки розподіленого додатка і забезпечує доступ клієнтів до бази даних.

Основною частиною сервера додатків є віддалений модуль даних. По-перше, як звичайний модулю даних він є платформою для розміщення невізуальних компонентів доступу до даних і компонентів-провайдерів. Розміщені на ньому компоненти з'єднань, транзакцій і компоненти, інкапсулюючі набори даних, забезпечують триланковий додаток зв'язком з сервером БД.

По-друге, віддалений модуль даних реалізує основні функції сервера додатків на основі надання клієнтам інтерфейсу IAppServer або його нащадка. Для цього віддалений модуль даних повинен містити необхідну кількість компонентів-провайдерів TDataSetProvider. Ці компоненти передають пакети даних клієнтського додатку, а точніше компонентів TdientDataSet, а також забезпечують доступ до методів інтерфейсу.

Таким чином, за допомогою даних технологій було розроблено безпровідну систему виклику на основі Elmes RD448, яку можна використати у різних сферах людської діяльності таких як медицина, ресторанний бізнес і інші. Клієнтську частину даної системи було розроблено для Windows, Android систем.

1. 1 Л. Б. Кащеєв, С. В. Коваленко, С. М. Коваленко; Основи візуального програмування : навч. посібник. – Х.: Веста, 2011. – 192 с.
2. http://www.elmes.pl
3. http://docwiki.embarcadero.com/RADStudio/Seattle/en/Developing_DataSnap_Applications