

УДК 004.4

Интеграция RFID считывателя с веб-приложением с использованием службы Windows

*Ожегов Г. А., бакалавр
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»*

*Научный руководитель: Гапанюк Ю. Е., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
gapyu@bmstu.ru*

В современном мире большую роль в нашей жизни занимают компьютерные технологии. С ними мы ежедневно сталкиваемся на улице, в метро, у себя дома. Они помогают нам быстро совершать платежи, мгновенно обмениваться информацией, работать. Текущая реальность не мыслима без интернета и компьютеров.

Последнее время самой динамично развивающейся областью в мире компьютерных технологий являются веб-технологии. На мой взгляд этому способствует несколько причин:

1. Высокая скорость доставки потребителям. В отличие от десктопных программ, для использования достаточно открыть браузер и получить необходимую программу
2. Большое сообщество в сети интернет, которое оперативно помогает в решении сложных вопросов. Полученные знания накапливаются и остаются в открытом доступе, что увеличивает скорость образования
3. Большое количество открытого программного обеспечения, которым делятся участники глобального сообщества. Благодаря этому разработчики не создают программы «с нуля», а используют наработки человечества
4. Мультиплатформенность. Программы, созданные с помощью веб-технологий, легко переносимы и запускаются на большинстве программно-аппаратных платформ

Однако программы, созданные для веба имеют ряд ограничений:

1. Недостаточная производительность для узкоспециальных задач. Выполнять сложное моделирование, использовать профессиональные средства мультимедиа в браузере невозможно из-за недостаточной производительности. Это обуславливается спецификой языков программирования, мультиплатформенностью, хотя за последние несколько лет были достигнуты большие успехи.
2. Низкая степень интеграции с аппаратными средствами. Взаимодействие с внешними устройствами организует операционная система через специальные программы взаимодействия — драйвера. Доступ программиста к этим инструментам очень ограничен, поэтому степень проникновения веб-технологий в аппаратно-программные комплексы не велика.

Рассмотрим проблему интеграции с аппаратными средствами более подробно.

Существует несколько способов интеграции аппаратных средств со средой выполнения веб-приложений (с браузером):

1. Технология ActiveX. Устаревшая технология, сложная в реализации, не кроссплатформенная: поддерживается браузером Internet Explorer и Firefox при установке дополнительного расширения. Целесообразность ее использования в наше время сомнительно
2. Java-апплет. Приложение, написанное на языке Java с использованием специального интерфейса. Достаточно широко применяется для использования аппаратных средств, требует наличия установленной JRE (Java Runtime Environment). Метод достаточно кроссплатформенный.
3. Специальный веб-сервер, имеющий доступ к операционной системе и интерфейс для взаимодействия с браузером. Этот способ является воплощением клиент-серверной архитектуры. Он позволяет нескольким клиентам использовать один и тот же аппаратный ресурс, достаточно кроссплатформенный.

В данной статье описывается создание специального веб-сервера, упомянутого выше.

К серверу можно предъявить ряд требований:

1. Скорость. При появлении данных на устройстве, они должны быть прочитаны с минимальной задержкой. Также при отправки информации из браузера, данные должны быть переданы в устройство максимально быстро
2. Надежность. Данные при передаче не должны теряться и искажаться

3. Доступность. Большое количество клиентов на одно устройство не должно приводить к недоступности устройства, если данное количество предусмотрено в спецификации
4. Кроссплатформенность. Веб-сервер должен быть переносим между платформами при наличии драйверов

Принципиальная схема взаимодействия представлена на рисунке 1.

Веб-браузер подключается к веб-серверу, который имеет доступ к устройству через API операционной системы. Когда на устройстве происходит какое-либо событие, сервер отвечает браузеру, и тот в свою очередь обрабатывает данные.

Важно понимать то, как браузер производит общение с сервером, как и когда устанавливает соединения, когда их разрывает. От этих факторов зависит скорость работы системы и ее надежность.

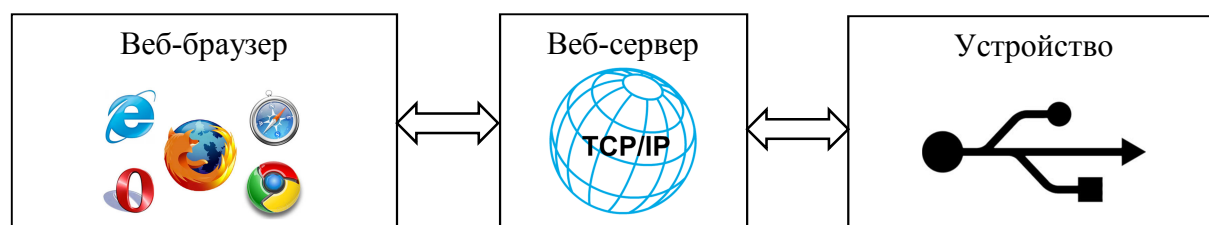


Рис. 1. Принципиальная схема взаимодействия

Существует несколько подходов работы с сетью в браузере.

Обычный запрос – ответ

Браузер время от времени запрашивает очередную порцию данных, сервер ее отдает.

Преимущества

1. Самый простой способ. Не требует глубокого понимания работы сети

Недостатки

1. Большое количество накладных расходов на передачу данных. Помимо HTTP заголовков будут тратиться ресурсы либо на установку подключения при каждой передаче данных, либо на поддержание соединения (если включена поддержка Keep-Alive)
2. На сервере необходимо учитывать с какого момента времени нужно отдать данные. Соответственно браузер должен передавать время последнего обновления

3. Маленькая скорость. Появление данных будет происходить не в режиме реального времени, что будет влиять на скорость передачи данных

Метод long-pooling

Браузер подключается к серверу, сервер устанавливает соединение, но данные не пересылает до тех пор, пока они не появятся. После появления данных производится их передача, и соединение закрывается. Если соединение закрылось по таймауту, то браузер вновь переподключается к серверу

Преимущества

1. Данные пересылаются только тогда, когда это необходимо. Это позволяет снизить нагрузку на сеть

Недостатки

1. Более сложная логика взаимодействия, которую нужно тщательно запрограммировать
2. Расходы на переподключения
3. Необходимо переподключаться даже если данные не передавались

Протокол WebSocket

WebSocket — протокол полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Преимущества:

1. Очень быстрый обмен данными
2. Экономичность передачи
3. Надежность

Недостатки:

1. Необходима поддержка со стороны сервера
2. Старые браузеры не поддерживают эту технологию
3. Более высокая сложность разработки

В основу сервера лег протокол WebSocket. Он лучше всех остальных вариантов позволяет достичь ранее заявленных требований по скорости, надежности и доступности.

Устройством, которое необходимо интегрировать с браузером, является RFID считыватель IronLogic Z-2, который изображен на рисунке 2. Производителем предоставляются драйвера и комплекты разработчика для языков C#, VC++, Delphi.



Рис. 2. Внешний вид RFID-считывателя Z-2 с интерфейсом USB

Для реализации сервера был выбран язык C#. Язык C# является языком высокого уровня, программы, написанные на нем, просто интегрируются как службы Windows для работы в фоновом режиме. На рисунке 3 показана диаграмма классов сервера

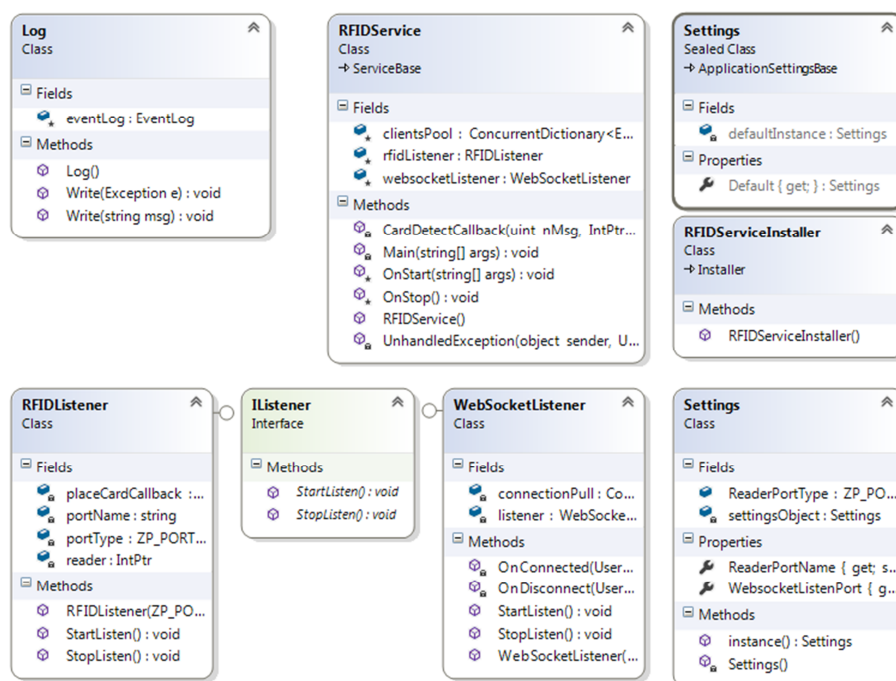


Рис. 3. Диаграмма классов сервера

В качестве реализации WebSocket протокола была использована библиотека Alchemy Websockets.

Основу функционирования сервера составляет класс RFIDService. Он запускает обработку данных с RFID считывателя за счет RFIDListener и начинает прослушивать tcp порт, к которому подключаются WebSocket клиенты. За обработку клиентов отвечает WebSocketListener. При возникновении данных на RFID считывателе RFIDListener вызывает функцию, которая итерируется по всем установленным соединениям и пересылает данные со считывателя.

Для обслуживания соединений в фоновом режиме в Windows предусмотрен механизм служб. Службы – это программы, которые не имеют пользовательского интерфейса, стандартный поток ввода отключен, а стандартный поток вывода отключен или перенаправлен в файл.

Для реализации службы класс RFIDService наследуется от системного класса ServiceBase. После сборки и установки программы в систему необходимо добавить службу. Для этого выполняется следующая команда:

```
sc create <имя службы> binPath= "<путь до исполняемого файла>"
```

Затем в разделе служб меню администрирования служба активируется и запускается.

Запущенная служба должна определить с устройством на каком порту необходимо работать. Существует несколько вариантов решения этого вопроса:

1. Передавать в качестве аргумента при запуске название порта, к которому нужно подключиться. В операционной системе Windows это реализуется с помощью редактирования строки запуска службы, которая находится в реестре. Это не удобный способ настройки, поэтому понадобится создать отдельную утилиту для конфигурации.
2. Опрашивать все порты и работать с теми, к которым подключен считыватель. В этом случае ручной конфигурации не

Следует уделить внимание логированию. Для быстрого выявления проблем необходимо выводить служебную и отладочную информацию с помощью специального класса *System.Diagnostics.EventLog*. Данные будут выводиться в Просмотре событий меню администрирования.

Служба работы с RFID считывателем является конкретной реализацией задачи интеграции аппаратных средств и web-технологий с использованием клиент-серверной архитектуры. Аналогичным способом можно интегрировать множества различных

устройств, создавать кластера из различных датчиков и интегрировать их с web-приложениями.

Список литературы

1. Microsoft Developer Network. Режим доступа: <https://msdn.microsoft.com/ru-RU/> (дата обращения 15.04.2014).
2. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб.: Питер, 2009. 432 с.
3. Самохвалов Э.Н., Ревунков Г.И., Гапанюк Ю.Е. Генерация исходного кода программного обеспечения на основе многоуровневого набора правил // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2014. № 5. С. 77–88.
4. Финкенцеллер К. Справочник по RFID. Теоретические основы и практическое применение индуктивных радиоустройств, транспондеров и бесконтактных чип-карт. М.: Додэка-XXI, 2008. 496 с.
5. Троелсен Э. Язык программирования С# 2010 и платформа .NET 4. М.: ООО «И.Д. Вильямс», 2011. 1392 с.