Загуменнов Ф.А. Программный интерфейс виртуальной реальности для встраиваемых систем // Сборник научных трудов. 16-ая молодежная научно-техническая конференция "Наукоемкие технологии и интеллектуальные системы 2014". – М.: изд-во МГТУ им.Н.Э.Баумана, 23-24 апреля 2014 г. – с.351-355.

ПРОГРАММНЫЙ ИНТЕРФЕЙС ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ ДЛЯ ВСТРАИВАЕМЫХ СИСТЕМ

Загуменнов Ф.А.
Научный руководитель: Юдин А.В.
МГТУ им. Н.Э.Баумана, кафедра РЛ6, Москва, Россия

SOFTWARE VIRTUAL REALITY INTERFACE FOR EMBEDDED SYSTEMS

Zagumennov F.A.
Supervisor: Yudin A.V.
Bauman Moscow State Technical University, Moscow, Russia

Аннотация
В статье рассматриваются возможности связи и взаимодействия микроконтроллеров с программным движком виртуальной реальности. Подробно исследована связь между микроконтроллером STM32L152 и движком Run3 через протокол USB. Кратко представлено руководство по реализации данного способа на любом микроконтроллере и любом приложении. В заключении представлены рекомендации по применению технологии в других приложениях.

Annotation
The article is about opportunities of a connection between MCU and the game engine. The connection between the STM32L152 MCU and the Run3 game engine using USB is described in detail. Briefly is described how to realize this technique on any microcontroller and any computer application, including non-gaming applications. In conclusion, there are advices how to use this technology in other applications.

Введение

Трёхмерные программные приложения перестали быть чем-то уникальным. В ближайшем будущем границы между виртуальным и реальным миром будут стираться. Всё большее количество трёхмерных приложений не использует кнопочные органы управления, они реагируют на свойственные человеку движения. Одновременно увеличивается количество приложений, позволяющих синхронизировать трёхмерные виртуальные движения с движениями механизмов в реальном мире. Самым главным звеном для синхронизации и сопряжения работы отдельных элементов подобных систем служит взаимосвязь, интерфейс их аппаратной и программной частей . В статье затрагивается тема связи современных микроконтроллеров и программ виртуальной реальности.

В наши дни существует множество устройств ручного ввода и управления техникой, такие как клавиатура, мышь, джойстик и проч., многие из которых изобретены еще полвека назад. Для ввода уже обработанной информации данные могут передаваться через известные программные протоколы, присутствующие почти на всех вычислительных устройствах. Многие персональные компьютеры (ПК) в наше время выпускаются без портов LPT и COM, через которые сравнительно просто можно передавать уже обработанные данные. Новизна исследования заключается в том, что приведённая ниже технология позволяет организовать связь через порт USB с предварительной обработкой данных между движком виртуальной реальности и любым информационным объектом электроники.

Для демонстрации принципов предлагаемого подхода был выбран 32-битный микроконтроллер МК STM32L152 на плате STM32L-Discovery. Данный выбор обусловлен современностью микроконтроллера, доступностью и большим количеством аппаратных возможностей и портов [3]. Микроконтроллер (МК) связан с движком виртуальной реальности Run3, автором которого я являюсь, на котором реализовано взаимодействие с

виртуальным миром при помощи получения данных с аппаратной части, и обратная взаимосвязь – управление аппаратной частью при помощи виртуальных объектов и событий. Сопряжение осуществляется посредством виртуального порта RS232 через порт USB. Данные пересылаются в виде инструкций к исполнению, обрабатываются программным движком и исполняются в реальном времени. Обеспечивается обратная связь с микроконтроллером, куда пересылаются данные при происхождении в игровом процессе определённых действий (например, нажатие виртуального переключателя). Движок исполняет инструкции при помощи скриптовой подсистемы Lua, файлы которого исполняет движок, обрабатывая запросы микроконтроллера. Данные на микроконтроллере обрабатываются прошивкой. В зависимости от полученной информации производятся определённые действия, например, установка состояния определённого порта и т.д. Описанный метод представлен в виде структурной схемы (рис.1).

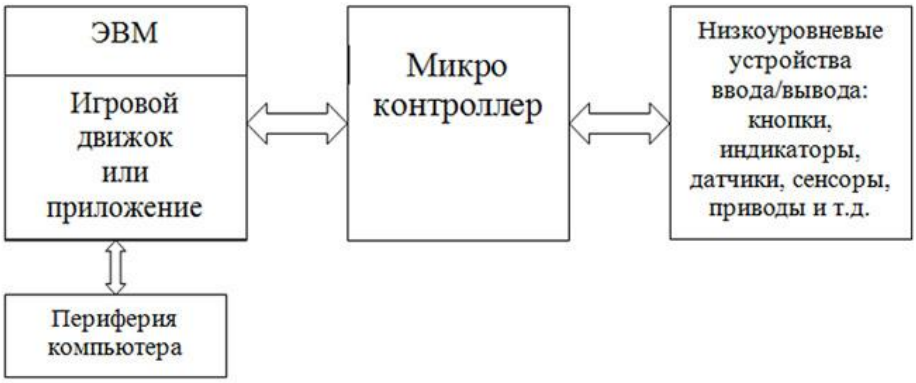

Рисунок 1 – Структурная схема системы

1. Получение данных с микроконтроллера

STM32L152 имеет аппаратную поддержку порта USB, которая реализована на ПК в виде виртуального порта RS232. При написании реализации описанного метода использовался готовый проект на STM32L152 с поддержкой USB[1]. В коде была оставлена лишь поддержка данного интерфейса. Для связи используется стандартная библиотека STM32_USB FS Device Driver от ST.

Стоит добавить, что для работы с аппаратным USB STM32 требуется установка внешнего кварцевого резонатора на 8 МГц. Данная реализация позволяет на любом компьютере иметь доступ к такому порту, принимать и пересылать на него данные. Для работы с COM-портом в состав Run3 был добавлен класс CSerial [2], реализующий простой обмен данными через порт. Таким образом, при правильном подключении STM32 через USB движок виртуальной реальности уже имеет возможность программно связываться с STM32.

2. Обработка данных

В этой части рассматриваются возможности обработки данных как на ПК, так и на микроконтроллере. В движок виртуальной реальности встроен дополнительный скриптовый язык программирования Lua, который применяется для обработки игровых событий. Целесообразно использовать его и для связи с функциями приёма и передачи данных. Функция передачи данных посимвольно отправляет строку через (виртуальный) последовательный порт. Примерный фрагмент кода с реализацией для использования совместно с Lua (код для приложения), данный код будет исполняться при регистрации функции в системе скриптов Lua и вызове этой функции из скрипта. Регистрация осуществляется следующим образом (табл. 2.1-2.2):

Таблица 2.1 - Код регистрации функции Lua

```
lua_register(pLuaState, "COMSendData",COMSendData)
```

Таблица 2.2 – Код функции

```
static int COMSendData(lua_State* pL){
  int n = lua_gettop(pL); //проверка количества входных параметров
if (n!=1){return 0;}
if (lua_isstring(pL, 1)){
        String tex = lua_tostring(pL, 1); //получение строки
if (global::getSingleton().port) //отправка строки на STM32 через COM
      global::getSingleton().port>SendData(tex.c_str(),tex.length());}
  return 1;
}
```

Приём данных на STM32 в данной реализации можно вести посимвольно. Для примера, при получении символа «a» микроконтроллер устанавливает лог. «1» на контакте, соединённом со светодиодом, что можно использовать для индикации (табл. 2.3):

Таблица 2.3 – Код приёма данных для STM32

```
      switch (ts){ //данные в ts через прерывание USART, подробнее см в [1]
case 'a':   {GPIO_TOGGLE(LD_GPIO_PORT, LD_GREEN_GPIO_PIN);    break;}
      //включим зелёный светодиод
case 'b':   {GPIO_ResetBits(LD_GPIO_PORT, LD_GREEN_GPIO_PIN);    break;}
      //выключим
default:break;}
```

Микроконтроллеры STM32 с ядром ARM являются современными и динамично развивающимися. Их возможности, по сравнению с многими микроконтроллерами другой архитектуры, выше, что в первую очередь заключается в наличии аппаратной поддержки многих интерфейсов и количестве отдельных функциональных портов на один МК.

Существует множество статей, посвящённых работе с устройствами, которые могут добавить новизны в рассматриваемое приложение интерфейса. Мы ограничимся описанием обработки данных, регулирующих порты общего назначения (т.н. GPIO). При нажатии и отпускании кнопки, замыкающей лог. «1» на контакт порта, порт отправляет строку на USB, например так (табл. 2.4-2.5):

Таблица 2.4 – Макрос отправки команды

```
  #define MACRO1(a1,b1,c1,d1,e1,f1,g1) if (GPIO_ReadInputDataBit(a1,b1)==1) \
                 { if (g1) { \
                              GPIO_SetBits(c1, d1); \ //Индикация
                              usb_out(e1); \  //Вывод данных
                              g1=0; \ //Переменная для установки флага
                         } \
                     } else  { \
                 if (!g1) { \
                 usb_out(f1); \
                 GPIO_ResetBits(c1, d1); \
                 g1=1; }}
```

Таблица 2.5 – Отправка данных на ПК

```
MACRO1(GPIOC,GPIO_Pin_0,GPIOB, GPIO_Pin_4,"arun3/lua/funcs/pin0.lua
","arun3/lua/funcs/pnn0.lua ",a);
```

Порт клавиш – C, пин – 0, порт с индикацией – B, пересылаемые значения - arun3/lua/funcs/pin0.lua при нажатии, и второе – при отпускании кнопочного переключателя.

Приведённый код для микроконтроллера позволит отправлять сообщения как, например, при нажатии, так и при отпускании кнопки, завязанной на порты ввода/вывода. Целесообразно отправлять данные в некотором формате, распознавая который, движок будет принимать определённые решения. Как одну из функций, следует встроить возможность запуска скрипта. При правильной реализации приведённый функционал обеспечит приложение всеми нужными возможностями для связи с МК.

В приложении виртуальной реальности нужно обрабатывать полученные данные каждый «фрейм», поэтому в программной реализации используется конструкция под названием frameListener, которая обрабатывает каждый кадр. Приведённый в табл. 2.6 код следует включить в подобную конструкцию.

Таблица 2.6 - frameListener

```
if (global::getSingleton().port){
global::getSingleton().port->ReadData(&a,1); //код команды
      if (a=='a'){ //команда «a» - исполнение скриптового файла
            char d[24]; // получение строки:
            global::getSingleton().port->ReadData(&d,23);
            String str = String(*&d,23);
            //проверка строки на корректность перед исполнением:
            if ((str[0]=='r')&&(str[22]=='a'))
            RunLuaScript(pLuaState,str.c_str());}}//исполнение скрипта
```

3. Реализация для других МК и приложений

Первым шагом к реализации связи с МК должно стать добавление в возможности движка класса работы с выбранным портом, который присутствует на микроконтроллере. Далее следует добавить функции, которые осуществляют приём и передачу данных через порт, причём функции должны быть доступны всем другим компонентам движка. Второй шаг заключается в переводе управления обвязкой микроконтроллера с ручной (если такая присутствует) на управление через выбранный порт, с учётом выбранного формата данных.

Благодаря введению в проект этих возможностей будет реализована система, полностью аналогичная приведённой выше. Код, показанный в предыдущих частях работы, можно считать алгоритмическим, то есть, описывающим лишь принцип действия, и это позволит реализовать данный способ на любом другом МК и приложении.

4. Практический результат

Реализация данной идеи представлена на рис.2.

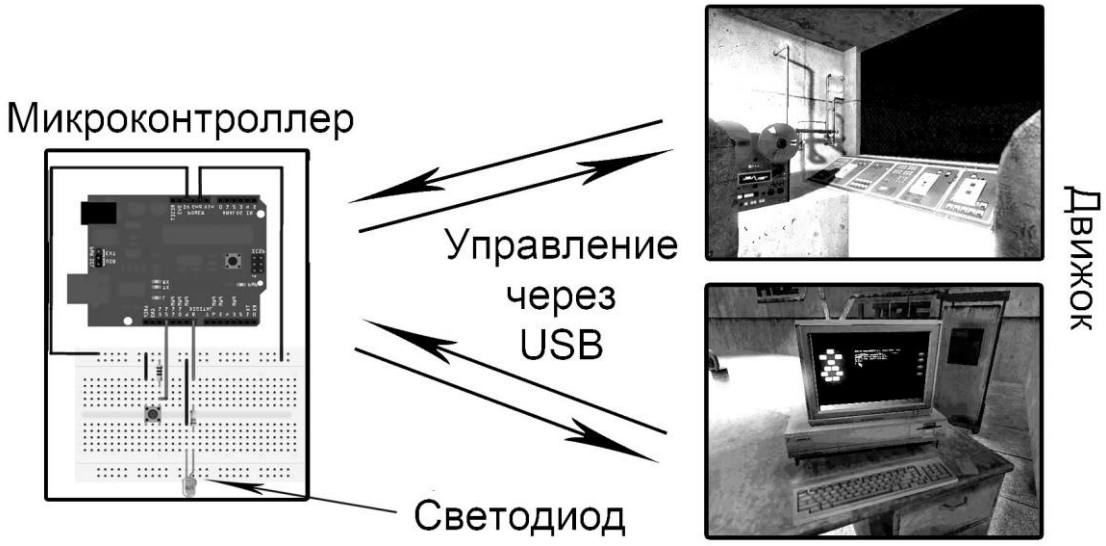

Рисунок 2 – Фотография и снимки экрана реализованной технологии.

Заключение

Сфера применения подобной технологии не ограничивается созданием игровых приложений с применением новых технологий управления и взаимосвязи.

При использовании других приложений и МК следует обращать внимание на то, что связь должна быть двусторонней. Внимание следует также уделять выбору интерфейса связи. Протокол USB является наилучшим вариантом, по причине его скорости, простоты подключения и простоты работы с ним.

Таким образом, мы рассмотрели, как можно совершать обмен данными между приложением виртуальной реальности и микроконтроллером.

Литература

1. Пример реализации связи по USB для STM32L152 – Электронный ресурс. Режим доступа: http://we.easyelectronics.ru/STM32/usb-virtual-com-na-stm32l---keil-project.html – Проверено 31.01.2014.
2. Класс CSerial для программной связи через COM-порт. – Электронный ресурс. Режим доступа:http://www.codeguru.com/cpp/in/network/serialcommunications/article.php/c2503/CSerial--A-C-Class-for-Serial-Communications.htm – Проверено 31.01.2014
3. Data-sheet(описание) на микроконтроллер STM32L152. – Электронный ресурс. Режим доступа:http://www.st.com/web/catalog/mmc/FM141/SC1544/SS1374/LN1041/PF248824 – Проверено 31.01.2014
4. LaMothe A., Tricks of the Windows Game Programming , 2E. — Sams, 2002.
5. Junker G., Pro OGRE 3D Programming. — Apress, 2006.