# ВВЕДЕНИЕ

……..

# Обзор методов и средств дистанционного управления робототехническими объектами. Постановка задачи

## Применение дистанционного управления в робототехнике

Использование роботов связано с необходимость решения задач, когда ее выполнение сопряжено с чрезмерным риском для здоровья человека, или находится вне его возможностей, а также в трудоемких задачах для решения которых требуется профессионально подготовленный персонал при его недостатке. К таким задачам относятся военные цели, разведывательные функции, цели физического уничтожения объектов, проведение проверок подозрительных безнадзорных предметов и бомб, автоматизация производственного процесса. Для решения этих задач и многих других используются роботы и робототехнические комплексы.

Классифицировать роботов можно по следующим признакам:

* область применения(промышленное, военное, исследовательское)
* среда эксплуатации (наземные, подводные, воздушные, космические)
* степени подвижности (стационарные, мобильные)
* по типу системы управления (программное, адаптивное, интеллектуальное)
* по функциональному назначению (манипуляционные, транспортные, информационные, комбинированные)
* по уровню универсальности (специальные, универсальные)
* по способу управления (автоматические, дистанционно управляемые, ручные и др.)

Для работы в различных условиях все большее значение получили мобильные роботы, отличительной чертой которых является способность перемещаться в пространстве, данный робот может быть представлен в виде совокупности 3 систем: транспортной, специальной и управления.[1]

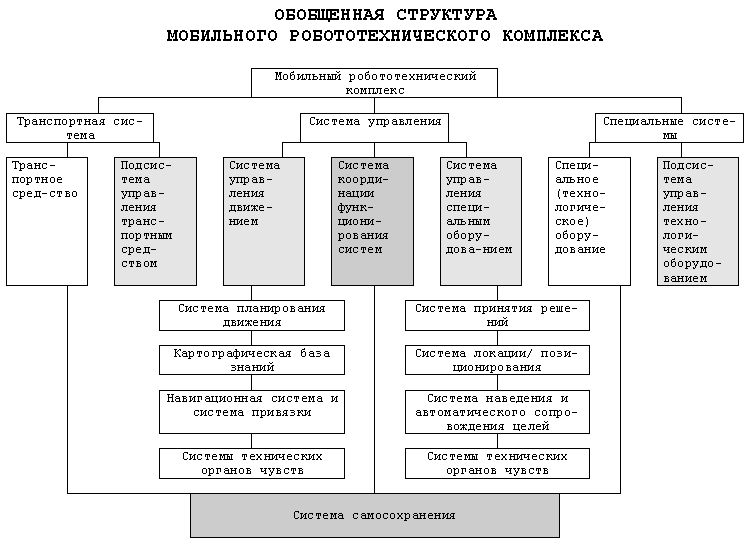


Рис.1.1 Обобщенная структура мобильного робототехнического комплекса

Количество систем и устройств, требующих беспроводного метода управления быстро растет. Здесь можно привести такие примеры как управление и сбор данных со спутников в космосе поддержание связи с ними. Управление мобильными роботами, например робот-сапер, которым часто пользуются службы МЧС, который позволяет осуществлять видеонаблюдение опасных участков и работать на расстоянии 500 метров от оператора, пехотные роботы используются военными США серия роботов Packbot [[http://www.irobot.com](http://www.irobot.com/)], нельзя не отметить дистанционно управляемые беспилотные летательные аппараты, например БПЛА Predator B (США), Skylark-I LE – разработка израильской компании Silver Arrow и другие. Имеются также и российские разработки серия Иркут, Эникс, Дозор, Tу-143 «Рейс» и другие. [<http://www.vector.ru/pdf/bpla-demo.pdf>]

Также к таким целям можно отнести территориально распределенные производственные комплексы, где использование кабеля для связи объектов не целесообразно с экономической или других точек зрения. Также беспроводное дистанционное управление используется в системах «умный дом», а также в различных охранных системах, системах контроля и диагностики, в которых датчики и пульт управления территориально находятся на некотором расстоянии, и прокладка кабеля невозможна или нецелесообразна, например экономической причине.

Над построением таких систем в настоящее время работает большое количество коллективов и компаний, многие компоненты можно без труда найти в открытой продаже и реализовать собственные проекты. В коммерческих проектах, которые позволяют осуществлять управление дистанционным и беспроводным образом зачастую в качестве пульта оператора используется собственные специализированные разработки или компактные ПК. Примерами таких пультов управления являются пульт управления саперным роботом МЧС (Рис. 1.2).



Рис.1.2 Пульт оператора мобильного робота-сапера

Специализированный пульт управления (Рис. 1.3) комплексом системы «Дозор-4» - данный комплекс «предназначен для сбора и обработки информации о заданных объектах (районах) с применением выбранных средств целевой нагрузки БЛА в целях обеспечения государственных министерств и ведомств данными, необходимыми для эффективного решения задач в ходе их служебной и хозяйственной деятельности.» [2]



Рис.1.3 Пульт управления комплексом «Дозор-4»

Для примера пульта управления оператора с помощью портативных ПК можно привести комплекс по управлению БПЛА Инспектор-201 (Рис.1.4) здесь используется портативный ПК (ноутбук) и несколько вспомогательных экранов. Время развертывания такого комплекса по заявлению разработчиков составляет 10 минут, а радиус действия не менее 5 км.



Рис.1.4 Пульт управления БПЛА Инспектор-101 с портативным ПК

Таким образом областей требующих дистанционного достаточное множество, в некоторых невозможно обойтись без беспроводных технологий. Уровень их развития позволяет выбрать по таким критериям как радиус действия, скорость передачи данных, потребляемая мощность подходящую технологию для решения поставленной задачи.

## Классификация технологий беспроводной передачи данных

Беспроводные сети подразделяют, в частности [3], по следующим признакам:

* по географической протяжённости: персональные (WPAN - Wireless Private Area Network), локальные (WLAN – Wireless Local Area Network), глобальные (WWAN – Wireless Wide Area Network);
* по локализации узлов: с подвижными узлами, со стационарными узлами;
* по топологии: точка-точка, многоточка;
* по ширине полосы передачи: узкополосные, широкополосные, сверхширокополосные;
* по способу первичной обработки информации: цифровые, аналоговые;
* по используемой технологии: спутниковые, атмосферные, оптические;
* по виду передаваемой информации: передача речи, видеоинформации, данных.

Дополнительно принято разделение на коммерческие и экспериментальные решения. «Под коммерческими подразумеваются разработки, которые производятся и тиражируются как готовый товар, под экспериментальными – разработки, на которых апробируются новые методы и технологии.»[4]

В данной работе используются системы радиосвязи, перечислим основные виды.[4]

* **WPAN (Wireless Personal Area Network – персональные сети)** – короткодействующие, радиусом около 10 м сети, которые связывают ПК и другие устройства – КПК, мобильные телефоны, принтеры и т.п. С помощью таких сетей реализуется простая синхронизация данных, устраняются проблемы с обилием кабельных сетей, реализуется простой обмен информацией в небольших рабочих группах. Две наиболее употребимые технологии – Bluetooth и ZigBee.
* **WLAN (Wireless Local Area Network – беспроводные локальные сети)** - радиус действия составляет сотни метров. С их помощью реализуется беспроводной доступ к групповым ресурсам в здании, университетском кампусе и т.п. Обычно такие сети используются для продолжения проводных корпоративных локальных сетей. В небольших организациях WLAN могут полностью заменить проводные соединения. Две наиболее употребимые технологии – Wi-Fi и WiMAX.
* **WWAN (Wireless Wide Area Network – беспроводные сети широкого действия)** – беспроводная связь, которая обеспечивает мобильным пользователям доступ к их корпоративным сетям и Интернету. Глобальные беспроводные сети WWAN отличаются от локальных беспроводных сетей [WLAN](http://ru.wikipedia.org/wiki/Wireless_LAN) тем, что для передачи данных в них используются беспроводные технологии [сотовой связи](http://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B2%D1%8F%D0%B7%D1%8C), такие как [UMTS](http://ru.wikipedia.org/wiki/UMTS), [GPRS](http://ru.wikipedia.org/wiki/GPRS), [CDMA2000](http://ru.wikipedia.org/wiki/CDMA2000), [GSM](http://ru.wikipedia.org/wiki/GSM), [CDPD](http://ru.wikipedia.org/w/index.php?title=CDPD&action=edit&redlink=1), [Mobitex](http://ru.wikipedia.org/w/index.php?title=Mobitex&action=edit&redlink=1), [HSDPA](http://ru.wikipedia.org/wiki/HSDPA), [3G](http://ru.wikipedia.org/wiki/3G).

В таблице 1.2.1 приведены основные характеристики технологий беспроводной передачи данных.

Таблица 1.2.1 Характеристики технологий радиосвязи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Технология/ Параметр | Bluetooth | ZigBee | Wi-Fi | WiMAX | Сотовые сети |
| Дальность действия | Десятки метров | Десятки метров | Сотни метров | Единицы километров | Определяется провайдером[[1]](#footnote-1) |
| Энергопотребление оборудования[[2]](#footnote-2) | Не более 0.1 Вт | Не более 0.1 Вт | Единицы Вт | Единицы Вт | Единицы Вт |
| Макс. скорость передачи | 3 Мбит/с | 250 Кб/с | 54 Мб/с и более[[3]](#footnote-3) | 75 Мб/с | 384 Кб/с |
| Макс. число пользователей | 255 | 7 | 32 | Определяется качеством связи[[4]](#footnote-4) | Определяется провайдером[[5]](#footnote-5) |

Помимо неоспоримых преимуществ беспроводных технологий, имеются также и недостатки. Помимо физических ограничений различных стандартов на количество подключенных устройств и усложнение топологии сети при алгоритмизации задачи увеличения количества абонентов, разработчики сталкиваются с проблемами связанными с временными задержками между отправкой команды и выдачей управляющего воздействия. При увеличении расстояния между пультом оператора и устройством падает качество связи, что сказывается на скорости передачи управляющей информации с пульта, и как следствие увеличение временных задержек. По этой причине часто становится невозможным управление системой в реальном времени, что влечет за собой увеличение затрат на реализацию вспомогательных алгоритмов для решения поставленных задач.

Вследствие этого возникают проблемы синхронизации работы устройств, внутренних часов. Что может привести к нежелательным последствиям, искажению измеряемых величин, например при определении местоположения с помощью системы спутниковой навигации.

Одной из проблем при беспроводной передаче является обеспечение безопасности. Ведь перехватить данные из эфира с помощью специальных устройств не составляет труда. Технология WiFi и различные стандарты предоставляют средства для создания безопасного обмена. В частности, это технологии шифрования ключа для доступа к сети с различными алгоритмами, WPA, WPA2, WEP, а также фильтрация по MAC-адресу. При использовании беспроводной передачи, в задачах где требуется максимальная защита передаваемых данных от перехвата, необходимо с особой ответственностью подходить к вопросу выбора алгоритма шифрования на уровне устройства поддерживающего сеть (модуль wifi) а так же реализовывать свои алгоритмы шифрования и протоколы при осуществлении обмена.

Другой проблемой является возможность заглушить сигнал с помощью специальных средств, на рынке присутствует великое множество «глушилок» работы беспроводных устройств. Некоторые характеристики такого устройства - Барьер-Д1 (Цена 18 тыс. руб):

* Блокируемые стандарты: DECT, Wi-Fi, Bluetooth, ZigBee, 4G.
* Дальность подавления: до 30 метров в радиусе от места установки (в зависимости от условий эксплуатации и местонахождения базовой станции).
* Антенная система: встроенная.
* Питание: 220В, 50Гц.
* Габариты: 260х180х65 мм.



Рис.1.5 Устройство блокиратора сотовой связи, wifi, Bluetooth, ZigBee

Существуют и более доступные с экономической точки зрения «глушилки» сигнала. На рынке существует большое разнообразие устройств от «Noname» прозводителя, в основном произведенные в Китае.



Рис. 1.6 Блокиратор радиосигнала gps, wifi

Заявленные производителем характеристики таких устройств:

* Эффективный радиус: 0,5-15 метров
* Питание: встроенная батарея 3.7V/1500mAh

Преимущества данных устройств это их компактность, но достаточно сильно снижается радиус действия.

Таким образом при проектировании системы, использующей в качестве средства передачи данных между блоками беспроводной сигнал, необходимо учитывать возможные нарушения нормальной работы такого рода «глушилками».

Также необходимо отметить, что на скорость работы непосредственно влияют и погодные условия если это территориально распределенное производство с отдельно отстоящими зданиями.

Выбор технологии зависит от технического задания на разработку системы, и основан на требованиях к дальности действия, условиям эксплуатации, виду передаваемой информации, скорости передачи и др. Таким образом поиск решений сводится к пошаговому сужению круга возможных технологий. Для увеличения скорости разработки таких систем, и прежде всего гибкости и взаимозаменяемости, целесообразно иметь библиотеку решений, которая позволяла бы строить систему из различных блоков, каждый из которых легко замещаем. Такой подход увеличит гибкость и надежность, сократит время решения подобных задач возникающих перед коллективом разработчиков.

## 1.3 Постановка задачи.

Таким образом, разработка специализированных портативных пультов управления робототехническими объектами является актуальной технической проблемой.

В данной работе решается задача создания средств беспроводного управления мобильным гусеничным роботом по протоколу Wi-Fi с применением портативного операторского пульта.

В качестве объекта управления выступает мобильный гусеничный робот, имеющий в своем составе модули WiFi и Bluetooth. На рис. 1.7 приведен внешний вид данного робота; его структурная схема представлена в приложении 1. На рис.1.8 изображена схема системы «Пульт оператора – Wi-Fi – Мобильный робот».



Рис.1.7 Внешний вид мобильного робота

Портативный

пульт

оператора

Объект управления

(мобильный робот)

Рис.1.8 Схема системы с иcпользованием Wi-Fi

В задачу разработки входит создание программных средств для портативного пульта оператора , обеспечивающих задание траекторий и управление мобильным роботом, а именно:

* считывание карты местности
* движение к указанной цели
* движение по заданной траектории

Также робот предоставляет сервисные функции, с помощью которых происходит задание координат целей, а также задание программы движения при командном управлении:

Передача этой информации должна базироваться на имеющемся множестве сервисных функций встроенного ПО робота:

* чтение во внутреннюю память данных
* запись во внутреннюю память данных
* чтение во внешнюю память данных
* запись во внешнюю память данных
* прием команды запуска

Протокол обмена подробно приведен в приложении 1.

В качестве портативного пульта оператора используется игровая приставка Sony PSP 3008 с модифицированной прошивкой HEN 6.20 внешний вид приведен на рисунке 1.9.



Рис.1.9 Внешний вид игровой приставки Sony PSP 3008

Отметим, что подобные разработки ведутся рядом зарубежных компаний, в частности, фирма WIFIBOT (Франция) предлагает подобное расширение функциональных возможностей для своих роботов серии WIFIBOT LAB V2, WIFIBOT M.

# Разработка структуры системы

## Особенности применения устройства Sony PSP

PSP обладает удобной эргономичной формой, имеет 10 клавиш для управления, 7 кнопок, выполняющих сервисные функции, один джойстик, а также имеет световые индикаторы, микрофон и динамики для воспроизведения звука.

Данная приставка обладает wifi модулем, поддерживающим алгоритмы шифрования WEP и WPA. Приставка обладает цветным 4,3 дюймовым ЖК экраном с разрешением 480х272 точек. На рис. 1.9 представлен внешний вид приставки. Также приставка обладает коммуникационным разъемом стандарта miniUSB и разъемом для карты памяти стандарта MS Duo.

Устройство снабжено процессором с тактовой частотой 333 Мгц, 64Мб оперативной памяти и аккумулятором емкостью 1200 мА\*ч.

Заявленное производителем время работы без подзарядки составляет 6 часов, а вес 189 грамм. По способностям отображения информации об объекте и весу, данное устройство представляет собой отличную альтернативу переносному компьютеру.

Для запуска приложений на приставке «из коробки» необходимо установить виртуальную прошивку – запустив специально сконфигурированную программу с карты памяти. После запуска прошивки становится доступной функция запуска пользовательских приложений, которые копируются на карту памяти приставки.

CPU разработан Sony специально для PlayStation Portable и представляет собой высокоинтегрированный чип, в котором, помимо центрального процессора, находится дополнительное ядро для декодирования мультимедийного контента. Оба ядра базируются на архитектуре MIPS R4000. Основное ядро (Allegrex) содержит в себе FPU и VFPU (векторный FPU). Дополнительное ядро (Media Block) содержит в себе процессор для декодирования аудио и видео (Virtual Mobile Engine) и 2 МБ EDRAM

Основное ОЗУ системы обладает объемом 32Мб, а так же встроенную DRAM объемом в 4Мб. Так же имеется возможность расширения памяти с помощью карточек Memory Stick Duo объемом до 32Гб.

Данная игровая консоль позволяет запускать не только игровые приложения, но обладает и рядом предустановленных программ, таких как веб-браузер, приложение для голосового общения Skype. Таких программ ограниченно количество и не всегда они удовлетворяют нуждам пользователей, поэтому существует стороннее ПО решающее прикладные задачи, чтение электронных книг, калькуляторы, и другие. Для запуска таких пользовательских приложений требуется модифицированная прошивка для приставки, официальная версия не позволяет запускать самописные приложения.

Имеется несколько способов установки сторонней прошивки, это перепрограммирование устройства на уровне флэш памяти, и запуск виртуальной прошивки. Виртуальная прошивка позволяет снизить риски нарушения работы приставки, притом предоставляет все возможности что и в первом случае. К преимуществам виртуальной прошивки можно отнести ее универсальность, такую прошивку можно установить на любую модель PSP с различными чипами защиты.

Существует некоторое количество программ позволяющих использовать PSP как пульт управления для ПК, например программа WiFiController. Пример работы такого приложения приведен на рисунке 2.1.



Рис. 2.1 Использование PSP в качестве контроллера для ПК

Таким образом игровая приставка Sony PSP предоставляет не только развлекательные возможности для пользователя, но и позволяет решать прикладные задачи.

## 2.2 Цикл проектирования программного обеспечения для PSP

Данное устройство является не типичным для операторских пультов управления, и цикл проектирования приложений для такой платформы обладает рядом особенностей. Для создания приложения необходимо скачать набор компиляторов и библиотек [5], на сайте поставщика имеются установочные пакеты как для ОС семейства windows так и для debian/linux, для debian доступны пакеты для 32 и 64 разрядных ОС.

Использование этих библиотек позволяет использовать для разработки приложений стандартизированный процедурный язык C. Помимо этого языка можно создавать приложения на интерпретируемом языке Lua, для этого необходимо установить специальное приложение интерпретатор – Phoenix Game Engine. При использовании этого приложения разработчику становится доступной обширная библиотека функций для работы с периферийными устройствами приставки, и средства для работы с графикой.

Использование языка С или С++ открывает перед разработчиком более широкие возможности, это увеличение доступных функций для работы с графикой, возможность использовать многие функции стандартной библиотеки, а так же использование данного языка оправдано с точки зрения оптимизации программного кода, и возможности использования подходов объектно-ориентированного программирования.

Представим обобщенную структуру рабочего места разработчика для создания прикладного приложения для Sony PSP, с использованием инструментальной ЭВМ. Такая структура представлена на рис.2.2.

**ИЭВМ**

ОС Windows/Linux

Набор компиляторов

ИСР

USB

Программный эмулятор

Драйвер Sony PSP

**Приставка Sony PSP**

MiniUSB

Карта памяти

Лабораторный стенд с МК

COM

Мобильный робот

Wi-Fi

Shell51

Рис.2.2 Обобщенная структура места разработчика

В качестве инструментальной ЭВМ (далее ИЭВМ) можно использовать любой персональный стационарный или мобильный компьютер под управлением операционных систем семейства Windows или Linux, для каждой системы существует свой набор компиляторов. Еще одним обязательным блоком в ИЭВМ является наличие установленного драйвера для соединения с приставкой, данный драйвер поставляется с набором компиляторов. Для связи с PSP используется порт USB и порт miniUSB на стороне приставки, для связи используется кабель usb-miniusb.

Необязательными элементами являются наличие программного эмулятора приставки, а также интегрированной среды разработки (ИСР). Но данное ПО значительно упрощает процесс разработки, а также увеличивает скорость решения поставленной задачи создания прикладной программы по нескольким причинам. Первая это наличие средств в ИСР для проверки исходного текста программы на предмет синтаксических ошибок. Другой причиной является наличие существенных временных затрат при копировании разработанной программы на реальное устройство, обусловленное спецификой подключения игровой консоли к ПК. Так, для соединения ПК и копирования программы в память приставки необходимо проделать несколько действий: подключить кабель USB, выбрать на приставке режим USB, скопировать приложение в память устройства, выключить режим USB, запустить программу на выполнение. Использование программного эмулятора позволяет сократить такую последовательность до двух действий: выбор программы для выполнения в эмуляторе, запуск на выполнение. Что существенно снижает временные затраты.

К сожалению, у программного эмулятора имеются и недостатки, прежде всего это относится к тому, что отсутствуют коммуникационные возможности для отладки приложений требующих доступа к приемопередатчику приставки, и отладка таких приложений становится невозможной. Но для отладки частей программы, где имеется работа с файлами, или графикой, программный эмулятор позволяет сэкономить время.

На рис.2.2 помимо описанных необязательных блоков присутствует блок «Лабораторный стенд» который тоже необязателен но крайне рекомендуем при разработке. Стенд представляет собой такую же архитектуру как и та что используется в мобильном роботе, структура СУ объекта совпадает со стендовой за исключением отсутствия системы технического зрения (СТЗ).

Выбор того или иного языка зависит от требований предъявляемых к приложению, скриптовый язык Lua предпочтительнее использовать в достаточно простых приложениях не требующих сложных алгоритмов, или в приложениях где временные задержки не критичны. Данное средство позволяет развернуть приложение более короткие сроки, чем приложение аналогичное по своей сложности, но созданное с помощью языка С. Так, опытным путем было установлено, что использование функций передачи данных посредством wifi, с помощью Lua практически не контролируемо. Для посылки сообщения используются блокирующие системные вызовы, которые при ошибке передачи выжидают большой промежуток времени (около 5 минут) во время которого работа с органами управления становится недоступна, и отобразить или получить данные от пользователя невозможно. Тогда как при использовании языков С/С++ есть возможность использовать потоки, и распараллеливать работу приложения по потокам, тем самым предоставляя пользователю управление пультом в любой момент времени, и что немаловажно извещать его о том что происходит в системе.

Так как при управлении мобильным гусеничным роботом требуется надежность, и скорость обработки информации на пульте оператора (часть информации можно обрабатывать на пульте, так как в данном случае пульт обладает большими вычислительными способностями, нежели система управления робота) то был выбран язык С/С++ как наиболее гибкое средство разработки.

Особенностью при создании приложения на С/С++ является создание правильно сконфигурированного makefile файла, который сдержит настройки для компилятора и линковщика. В данном файле указываются какие библиотеки требуется подключить и скомпилировать для получения готового рабочего приложения для Sony PSP.

Далее после компиляции приложения на выходе оказываются два файла с расширениями .elf и .pbp, которые требуются для запуска, эти файлы копируются в рабочую директорию приставки, и запускается как обычная игра из меню.

Редактор

Транслятор

Действия по размещению программы в памяти

Память PSP

Средство отладки

Копирование с помощью ПК

на карту памяти по usb

Память эмулятора

Выбор файла из меню

эмулятора

Отладчик

ИСР

Рис.2.3 Структура разработки с помощью ИСР

На рисунке 2.3 изображена структура цикла разработки с помощью интегрированной среды разработки, также показан блок программного эмулятора и средство отладки для PSP. Для отладки приложения требуется установить на приставку приложение PSPLink которое предоставляет интерфейс, позволяющий выполнять пошаговое выполнение программы с помощью отладчика GNU Debugger (gdb).

Еще одной особенностью является использование библиотечных функций специфических только для данной платформы, поиск необходимых функций облегчается наличием html версии тематически категорированной документации.

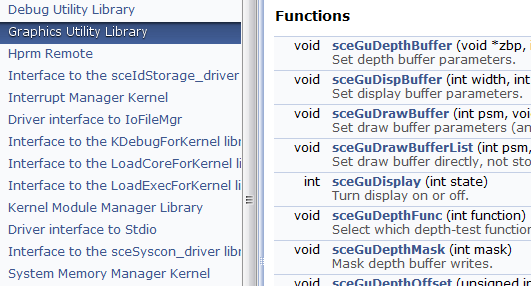


Рис. 2.4 Вид страницы документации

## 2.3 Структура программного обеспечения

Для начала необходимо разработать структуру существующего библиотечного комплекса готовых решений для данной задачи. Представим это в виде нескольких блоков.

На рис. 2.5 приведена схема полной библиотеки разрабатываемых технических решений для осуществления управления мобильным роботом. Красным цветом выделен блок который разрабатывается в данной работе. Как видно из рисунка, это управление роботом с помощью портативной игровой консоли Sony PSP по технологии WiFi. Для этого робот обладает помимо модулей Bluetooth, WiFi модулем. Раскроем блок PSP и WiFi более подробно.

оператор

объект

ПК

PSP

Моб. Тел.

WiFi

BT

WiFi

BT

эфир

ZigBee

Рис.2.5 Схема библиотеки технических решений

Дисплей и органы управления

Пользовательское приложение

Драйвер

Приёмопередатчик

Эфир

Рис. 2.6 Блок PSP и WiFi

Как видно из рисунка 2.6, управление объектом происходит в несколько этапов, первым звеном в цепочке является интерфейс с пользователем, после того как оператор задал команду с помощью интерфейса PSP приложение обрабатывает запрос и обращается к драйверу приемопередатчика посредством системных вызовов с помощью библиотечных функций. Аналогичным образом происходит обработка полученных данных от объекта, и отображение полученной информации на дисплее приставки. Диаграмма управления приведена на рис.2.7.

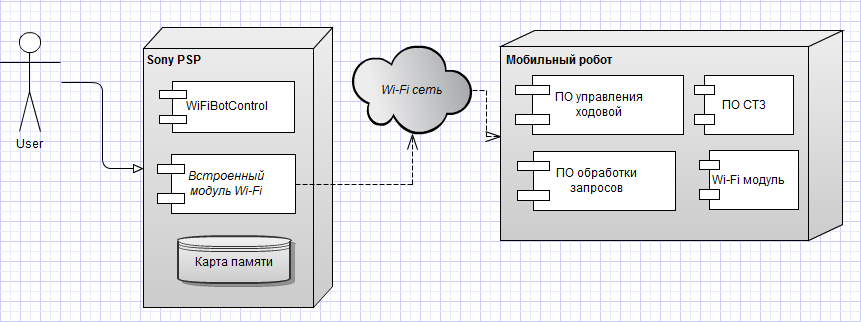


Рис.2.7 Диаграмма управления роботом

Раскроем схему изображенную на рисунке 2.6 более подробно.

Рис.2.8 Структурная схема ?????

Процесс управления осуществляется по нескольким путям, прежде всего оператор вызывает модуль получения карты от мобильного робота (путь 1-2-3), по завершению его работы, на карте памяти формируется файл содержащий данные об объектах находящихся перед роботом. Далее вызывается модуль отображения считанной карты на экран пульта оператора, это происходит сразу после получения последнего значения считанной карты. Перед отображением результаты обрабатываются согласно нелинейной характеристике системы технического зрения, которая приведена в приложении 2.

После выбора направления движения с помощью клавиш необходимо запустить подпрограмму, которая обработает полученную информацию о выбранном направлении и передаст необходимые управляющие команды мобильному роботу по каналу Wi-Fi.

# Программные реализации

## 3.1 Создание каркаса приложения

Портативная игровая консоль является не типовым устройством, поэтому следует остановиться подробнее на некоторых ключевых особенностях создания программы. Каркас приложения состоит из нескольких потоков, один из них обрабатывает выход из программы по нажатию клавиши Start, другой поток – главный поток программы функции main.

Для функционирования приложения необходимо установить обратные вызовы (call backs). Представим схему заготовки программы, рис.3.1.

Начало

Установка обратных вызовов

Инициализация экрана

Завершение потока

Конец

Рис. 3.1 Схема программы каркаса приложения

Приведем фрагмент программы.

Листинг 3.1 Установка обратных вызовов

// обратный вызов выхода из программы

int exit\_callback(int arg1, int arg2, void \*common)

{

sceKernelExitGame();

return 0;

}

// установка обратных вызовов, созание потока

int SetupCallbacks(void)

{

int thid = 0;

thid = sceKernelCreateThread("update\_thread", CallbackThread,

0x11, 0xFA0, PSP\_THREAD\_ATTR\_USER, 0);

if(thid >= 0)

{

sceKernelStartThread(thid, 0, 0);

}

return thid;

}

И определяем главную функцию программы main().

Листинг 3.2 Главная функция программы

int main(int argc, char \*\*argv)

{

// установка обратных вызовов

SetupCallbacks();

// активация экрана

pspDebugScreenInit();

// завершение потока

sceKernelExitDeleteThread(0);

return 0;

}

Это является стандартной частью при создании любого приложения решающего прикладные задачи. Удобно создать файл-шаблон, и при создании нового приложения достаточно выбрать такой файл и проект будет автоматически создан и будет содержать файл исходного кода с содержанием файла-шаблона.

Также необходимо отметить, что для компиляции приложения необходимо сформировать makefile файл, приведем фрагмент такого файла.

Листинг 3.3 Фрагмент makefile файла

TARGET = netsample

OBJS = main.o

EXTRA\_TARGETS = EBOOT.PBP

На представленном фрагменте (листинг 3.3) необходимо отметить ключ TARGET, данный ключ содержит название программы, OBJS определяет какие будут созданы объектные файлы, ключ EXTRA\_TARGETS определяет исполняемый файл.

## 3.2 Работа с интерфейсом портативной приставки

Для ввода данных от пользователя используются органы управления приставкой которые были описаны в первой главе, основными элементами ввода являются кнопки, так как управление голосом достаточно ресурсоемко и сопряжено с трудностями настройки для различных голосов.

Составим схему программы и приведем небольшой фрагмент подпрограммы опроса пульта оператора, данная подпрограмма работает как отдельный поток.

Начало

Установка обратных вызовов

Инициализация экрана

Заполнение структуры нажатых клавиш

Нажата клавиша?

нет

да

Обработка нажатия клавиши

Рис.3.2 Схема подпрограммы обработки нажатых клавиш

Приведем фрагмент исходного текста подпрограммы.

Листинг 3.4 Подпрограмма опроса клавиш

…

while (1)

{

sceCtrlReadBufferPositive(&pad, 1);

printf("Analog X = %d ", pad.Lx);

printf("Analog Y = %d \n", pad.Ly);

if (pad.Buttons != 0){

if (pad.Buttons & PSP\_CTRL\_LTRIGGER) {

// обработка нажатия левой кнопки под указательным пальцем

printf("L-trigger pressed \n");

}

if (pad.Buttons & PSP\_CTRL\_RTRIGGER) {

// обработка нажатия правой кнопки под указательным пальцем

}

if (pad.Buttons & PSP\_CTRL\_CROSS) {

// обработка нажатия кнопки крестик

}

if (pad.Buttons & PSP\_CTRL\_UP) {

// обработка нажатия стрелки вверх

}

if (pad.Buttons & PSP\_CTRL\_DOWN) {

// обработка нажатия стрелки вниз

}

}

}

return 0;

}

…

После компиляции программы формируется .elf и .pbp файлы, которые загружаются в приставку, или в программный эмулятор, демонстрация работы такого простейшего приложения приведена на рисунке 3.3.

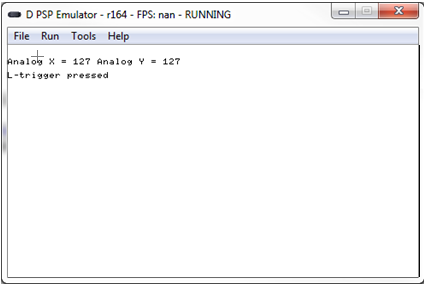


Рис.3.3 Обработка нажатий клавиш

## 3.3 Работа с графическими примитивами

Данная приставка обладает огромными мультимедийными способностями, так как изначально разрабатывалась для использования в игровой сфере, впрочем на сегодняшний день это и есть основное направление развития данного устройства – игровая индустрия. Но эти возможности можно использовать и в операторских пультах, создавая трехмерную и двумерную графику, с различными эффектами для большего удобства оператора и наглядности. Большой экран и разрешение позволяют расположить достаточное количество справочной информации необходимой для процесса управления.

Во время разработки программного обеспечения приставки для управления мобильным роботом, как было упомянуто выше, использовался принцип постепенного наращивания функциональности с помощью добавления новых модулей. Одним из таких является модуль отображения карты (рис. 2.8), приведем некоторые особенности, которые были обнаружены при его создании.

Отправной точкой при разработке являлось приложение обрабатывающее нажатие клавиш, была добавлена работа с графикой, и создано приложение по управлению графическим элементом «Х». Пример такого приложения приведен на рис.3.3.

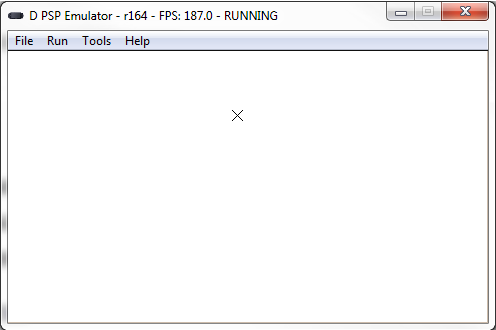


Рис.3.4 Экран программного эмулятора с запущенным приложением «Бегунок»

Особенностью является использование специальной библиотеки graphics.h которая содержит функции для создания графических примитивов, так же в мейкфайл необходимо добавить создание объектных файлов graphics и framebuffer.

OBJS = main.o graphics.o framebuffer.o

А также указать подключаемые библиотеки:

LIBS = -lpspgu -lpng -lz –lm

Составим схему программы.

Начало

Установка обратных вызовов

Инициализация экрана

Заполнение структуры нажатых клавиш

Нажата клавиша?

нет

да

Клав ВЛЕВО?

Клав ВПРАВО?

Отобразить вектор на экране

Изменить координаты по

часовой стрелке

да

нет

Изменить координаты против часовой стрелки

Рис.3.5 Схема программы «Бегунок»

Дальнейшим шагом было отображение карты из локального файла на карте памяти, в файле располагаются коды расстояний до препятствия, снятые СТЗ мобильного робота.

Наиболее удобным способом отображения координат является способ в полярных координатах. Так как система технического зрения мобильного робота анализирует расстояния до предметов в развороте 180 градусов, то и для отображения этих координат будем использовать полуокружность. Так же добавим несколько дуг для градации расстояний до предмета.

Считывание данных из файла производится обычным способом как и в программах для любой другой системы, с помощью функций fgets().

Так как в sdk не оказалось функции для рисования окружностей, был написан метод по алгоритму Брезенхема для рисования окружностей[[6]](#footnote-6) и добавлен в библиотеку. Исходный текст нового метода:

Листинг 3.5 Метод рисования окружности

void drawCircle(int x0, int y0, int radius, Color color) {

int x = 0;

int y = radius;

int delta = 2 - 2 \* radius;

int error = 0;

while(y >= 0) {

putPixelScreen(color, x0 + x, y0 + y);

putPixelScreen(color, x0 + x, y0 - y);

putPixelScreen(color, x0 - x, y0 + y);

putPixelScreen(color, x0 - x, y0 - y);

error = 2 \* (delta + y) - 1;

if(delta < 0 && error <= 0) {

++x;

delta += 2 \* x + 1;

continue;

}

error = 2 \* (delta - x) - 1;

if(delta > 0 && error > 0) {

--y;

delta += 1 - 2 \* y;

continue;

}

++x;

delta += 2 \* (x - y);

--y;

}

}

Далее наносим точки на график по методу, что координата x=cos(угол)\*координата, а y = sin(угол)\*координата. Шаг изменения угла равен 0,024рад. Исходный текст данной программы «Radar Example» приведен в приложении 3. Здесь приведем результат работы данного приложения в эмуляторе.

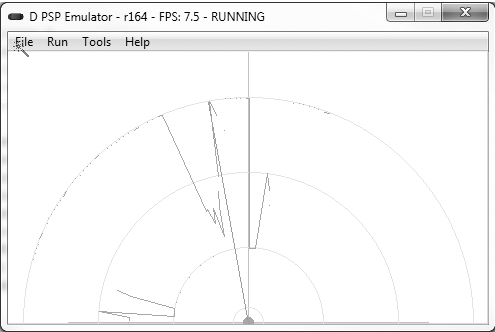


Рис. 3.6 Результат работы программы в эмуляторе

Здесь, точки, которые лежат за пределом кода 150 СТЗ, обозначены серым цветом, что означает, что данное направление свободно для движения.

Более близкие точки и объекты выделены красным цветом и соединены красной линией, для того, чтобы иметь возможность оценить размеры препятствий. Данная программа была проверена на работоспособность на реальном устройстве, и показала работу, как и ожидалось.

## 3.4 Работа с сетевыми средствами

В качестве сетевого средства обмена между устройствами (ППО и мобильный робот) используются сокеты. Следуя методологии последовательного усложнения. Прежде всего было создано простейшее приложение, которое позволило ознакомиться с методами отправки и посылки данных через беспроводную сеть Wi-Fi.

По существу работа с сокетами не отличается от работы на любой другой платформе. Особенностью является выполнение определенных действий для подключения к беспроводной сети. Помимо автоматической настройки через интерфейс приставки необходимо выбрать в программе доступную сеть, и подключиться к ней. На данный момент в программе не реализована поддержка выбора необходимой сети, и задается жестко константой.

Приведем методы для выбора сети и получения IP адреса, при условии что в сети работает DHCP-сервер.

Исходные коды программ представлены в приложении 4, здесь приведем схемы программ подключения к точке доступа (ТД) и получения IP адреса.

Инициализация сети

(загрузка модулей)

Подключение к ТД

Получение IP адреса

IP != 0

начало

конец

Возврат ошибки

да

нет

Рис. 3.6 Схема программы подключения к точке доступа

Инициализация сетевых средств и подключение к точке доступа происходит в несколько этапом. Прежде всего загружаются модули для работы с сетью, далее необходимо получить статус выбранной сети, если ошибки не возникает, DHCP выдает IP адрес.

Приведем пример отправки и приема информации через сокет. Для этого используются системные вызовы socket(), send(), recv(). Для работы с сокетами необходимо подключение дополнительных хидеров:

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

Приведем фрагмент программы при отправке некоторых данных через сокет.

// отправляемые данные

char esc=27

// отправка данных через сокет sock

If (send(sock, &esc, 1, 0)) printf(“Message sented”);

Здесь используется синхронная блокирующая отправка данных, это означает что поток будет заблокированным до момента принятия сообщения другим устройством. В библиотеке присутствуют средства для осуществления асинхронной передачи данных, но на данном этапе работы достаточно использования синхронной передачи.

## 3.5 Схема программы

В результате разработанных примитивных блоков, которые требуются для построения приложения управления мобильным роботом, становится возможным их объединение в единое целое.

Приведем схему программы организующей управление роботом по сети Wi-Fi.

начало

Инициализация сети

Подключение к ТД

Отображение карты



Заполнение структуры

нажатых клавиш

Отправка задания

Обновить карту

Запросить карту

Изменить вектор

Рис.3.7 Схема программы BotControl

Полный исходный текст программы приведен на прилагаемом компакт-диске. На рисунке 3.8 приведен внешний вид разработанного приложения для Sony PSP.

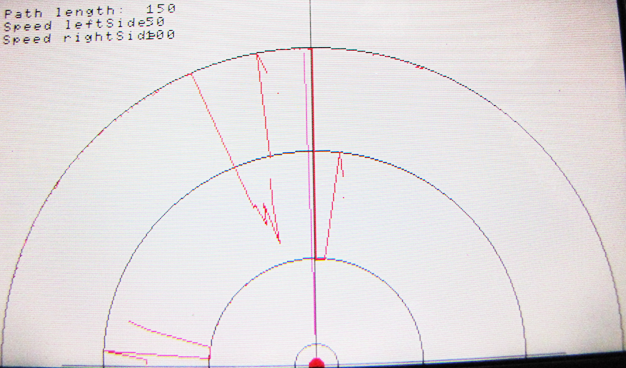


Рис.3.8 Внешний вид приложения

На приведенном выше рисунке видны все основные параметры необходимые для принятия решений о движении в ту или иную сторону. Так, в полярных координатах отображаются препятствия перед роботом, СТЗ мобильного робота определяет расстояние в пределах от 20 до 150 см, препятствия представляющие опасность соединены красной линией.

# **Проведение экспериментальных исследований**

## План испытаний

Для проведения испытаний необходимо составить план их проведения, так чтобы покрыть все необходимые модули, и протестировать работу приложения.

Испытания будем проводить последовательно для различных модулей составляющих приложение. Вследствие того, что приложение разрабатывалось последовательным наращиванием функциональности, и последовательным усложнение структуру ПО, и состоит из нескольких элементарных подпрограмм, логично проводить испытания также последовательно для каждого из модулей.

Одним из первых разработанных модулей для приложения было создание программы обеспечивающей взаимодействие с пользователем, прием информации от него и отображение ответных реакций на экран приставки. Тестирование будет проводиться последовательным нажатием различных управляющих клавиш и анализ полученных результатов на экране пульта. По сравнению теоретически ожидаемых и экспериментальных результатов будет сделан выбор о правильной или неправильной работе модуля.

Следующим блоком программы является модуль обмена информацией по беспроводному каналу между пультом оператора и мобильным роботом. Так как разрабатываемая платформа является нетиповой и отсутствует опыт разработки подобных приложений, разработка данного модуля была разделена на несколько более простых подпрограмм. Испытания также будут проводиться согласно последовательности разработки, а именно:

* испытания передачи одного байта в связке PSP – ПК
* испытания записи одного байта в ячейку стенда PSP – стенд - ПК
* испытания реализованного протокола в связке PSP-стенд

Последним испытанием является проведение комплексного тестирования разработанного приложения для пульта. Во время проведения этого испытания исследуется работа в связке PSP – Мобильный робот. В ходе данной проверки с пульта оператора задается программа (считывание карты, отправка команды движения) и отправляется роботу, по результатам формируется вывод об успешности или неудаче при проведении такого испытания.

## **Испытания работы интерфейса и графических средств**

Для испытания графического модуля, первоначально использовался программный эмулятор разработанный на языке D[[7]](#footnote-7) разработчиком именуемым себя soywiz. На рисунке 4.1 представлен внешний вид эмулятора PSP.

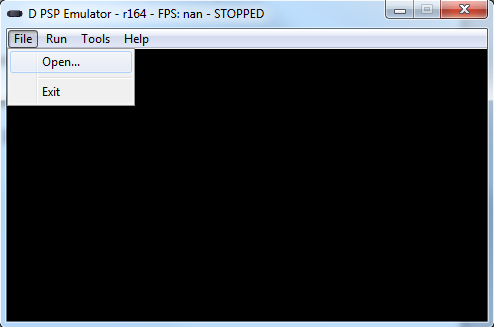


Рис. 4.1 Внешний вид эмулятора

Для проведения испытания графических средств загрузим в эмулятор скомпилированную программу состоящую из модуля обработки нажатий клавиш, который выводит на экран информацию о нажатой клавише. Исходный текст представлен в приложении 5. Последовательно нажимаем клавиши «L-триггер» и клавишу «Вверх».

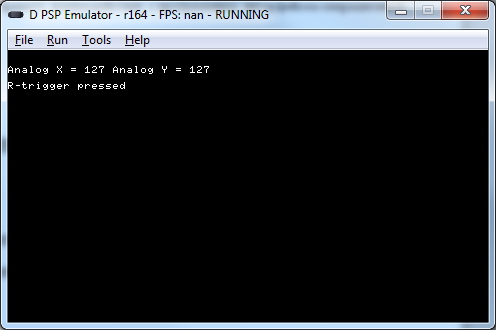


Рис.4.2 Работа модуля обработки нажатий клавиш, нажата клавиша «L\_trigger» в эмуляторе PSP

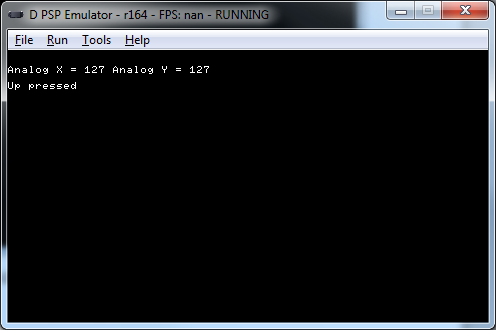


Рис.4.3 Работа модуля обработки нажатий клавиш, нажата клавиша «Вверх», в эмуляторе PSP

Из рис.4.2-3 видно, что программа работает верно, и на экране отображается ожидаемая информация. Для испытания динамической работы с интерфейсом было создано приложение обеспечивающее движение графического объекта с помощью клавиш (в качестве объекта используется крестик), исходный текст программы приведен в приложении 5. Результат работы данного приложения изображен на рисунке 4.4, а работа приложения на реальном устройстве приведена на рис.4.5.

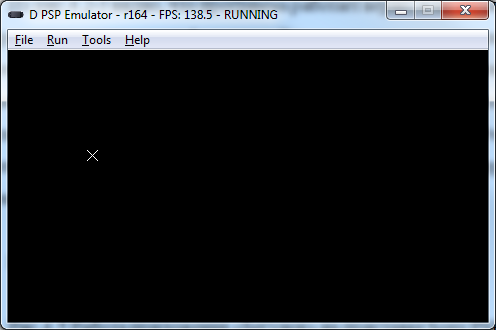


Рис.4.4 Работа приложения «Бегунок» в эмуляторе



Рис.4.5 Работа приложения «Бегунок» на приставке Sony PSP

На рисунках видно, что крестик отображается на экране, нажатие кнопок «влево» и «вправо» приводило к изменению положения графического объекта на экране, что совпало с теоретическими ожиданиями.

Таким образом, испытания графической и интерфейсной части можно считать завершенными успешно, все модули работают как и ожидалось, это подтверждается испытаниями на двух платформах: на программном эмуляторе, и реальном устройстве.

## Испытания работы сетевых средств

Как было указано в §4.1 испытания сетевых средств разделяются на 3 составляющих:

* испытания передачи одного байта в связке PSP – ПК
* испытания реализованного протокола в связке PSP-ПК
* испытания записи одного байта в ячейку стенда PSP – стенд – ПК
* испытания реализованного протокола в связке PSP-стенд

Проведем испытания простейшей отправки байт с приставки на ПК, на котором запущена программа эмулирующая работу стенда. Программа создает сокет, и переходит в режим ожидания клиентов. Исходный текст данной программы приведен в приложении 6. Создание подобной программы обусловлено удобством ее использования в условиях отсутствия лабораторного стенда или мобильного робота. Данная программа позволяет организовать любой протокол для обмена с приставкой, или любым другим приложением, использующим сокеты и подключающимся в качестве клиента.

Исходный текст приложения для отправки байта серверу приведен в приложении 7. Для исследования правильности работы отправим с помощью пульта десятичное число серверу, и проанализируем полученные результаты.

Структура экспериментальной системы приведена на рисунке 4.6.

Портативный

пульт

оператора

ПК с запущенным

сервером

Рис. 4.6 Структура испытуемой системы

На стороне ПК запустим приложение сервер, которое откроет сокет на 5555 порту, и будет ожидать сообщений. На стороне пульта оператора запустим приложение «netsample», дождемся подключения к сети, и нажмем клавишу «X» для отправки байта по сети. Результат работы программы сервера приведен н рисунке 4.7.

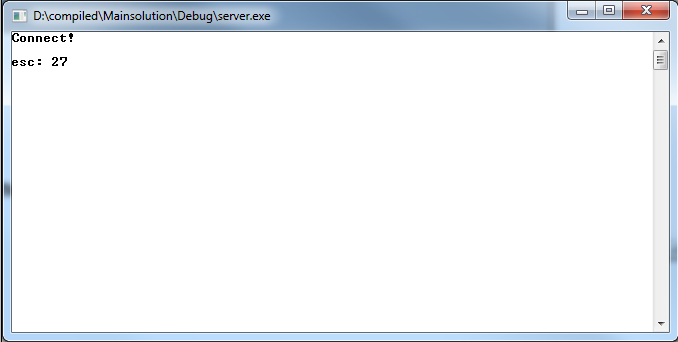


Рис.4.7 Прием десятичного 27 на стороне сервера

Как видно из рисунка 4.7, байт успешно принят. Так как это простейшее приложение, то отправление ответа пульту оператора не предусмотрено, и приложение просто завершает свою работу.

Исследуем работу передачи данных между пультом оператора и сервером согласно протоколу обмена. Структура системы остается неизменной, и изображена на рис.4.6.

На стороне ПК запускается приложение сервер, которое в зависимости от принятой команды отвечает некоторой константой (для команды чтения из памяти), или принимает байт данных (для команды записи в память), если принята команда старта, то отправляется константа. Фрагмент исходного текста программы сервера, реализующего обмен по протоколу, представлен в приложении 8.

Запустим сервер на ПК подключенном к беспроводной сети Wi-Fi, программа будет ожидать подключения клиента. Далее запустим приложение на Sony PSP, дождемся подключения к сети, выберем данные для отправки – адрес, код команды, данные – нажмем клавишу «Х» и проанализируем полученные результаты. Результат работы программы представлен на рисунке 4.8.

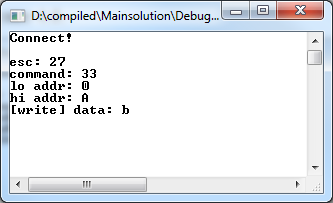


Рис.4.8 Результат работы сервера для принятой команды записи

Изображение, демонстрирующее действия, происходящие на экране пульта оператора, представлено на рисунке 4.9.

Рис.4.9 Результат выполнения отправки команды записи

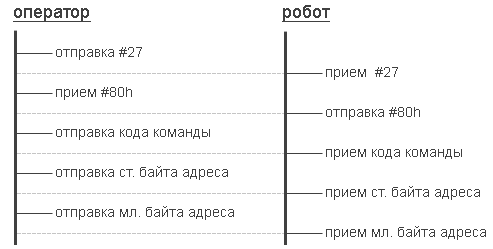
## 4.4 Комплексные испытания

# Список литературы

1. Публикации журнала "Специальная Техника" №6 1999 год.
2. http://www.airwar.ru/enc/bpla/dozor.html
3. Борисов Н.А. История развития беспроводных сетей / доклад. – СпбГУ ИТМО, СПб, 2009. – 29 с.
4. Калинина Е.Б, Выпускная работа
5. <http://www.jetdrone.com/minpspw>

# Приложение 1. Протокол обмена информацией с роботом

В протоколе существует одна общая часть, независимо от того какая функция выполняется, будет это команда записи, или команда чтения. Приведем эту общую часть.



При записи во внешнюю или внутреннюю память данных этот протокол дополняется следующим:



При чтении из внешней или внутренней памяти, к основному блоку добавляется следующее:



Последовательность действий для задания движения робота при командном управлении:

* код скорости правого борта
* код пути правого борта, старший байт
* код пути правого борта, младший байт
* код скорости левого борта
* код пути левого борта, старший байт
* код пути левого борта, младший байт
* пуск

Коды скорости: 1..127 – вперед, 128..255 – назад.

Коды пути: 256 – 124см дистанции – при движении прямо, 88 – поворот на 180 градусов при повороте.

Коды скорость необходимо записывать в память робота по следующим адресам:

* адрес кода скорости правого борта: 80
* адрес кода пути правого борта ст.б: 82
* адрес кода пути правого борта мл.б: 83
* адрес кода скорости левого борта: 81
* адрес кода пути левого борта ст.б: 84
* адрес кода пути левого борта мл.б: 85
* адрес пуска: 33034

# Приложение 2. Структура робототехнического комплекса

Микроконтроллер

Программы управления

Устройство сопряжения

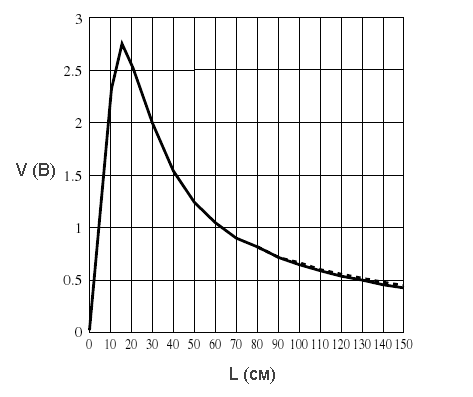
Привод СТЗ

СТЗ

Ходовые приводы

ФИД

Робот имеет систему технического зрения Sharp GP2Y0A02YK; спецификации данного модуля представлены ниже, необходимо отметить особенность графика функции , где V – выходное напряжение с датчика, L – расстояние до объекта, график приведен на рисунке. Данную характеристику необходимо учитывать при отображении карты на экране пульта оператора.



# Приложение 3. Тестовое приложение Radar Example

/\* функции возврата и выхода из программы

подключение заголовоных файлов \*/

int **main**(void)

{

SetupCallbacks();

pspDebugScreenInit();

initGraphics();

sceCtrlSetSamplingMode(PSP\_CTRL\_MODE\_ANALOG);

SceCtrlData pad;

FILE \*stream;

int coord[130]; // содержит набор значений расстояний

int radarX=240;

int radarY=50;

double corner=0;

// откроем файл с координатами и

// отрисуем, до начала работы с кнопками

start:

if ((stream=fopen("file.txt","r"))==NULL)

{

printf("can't open file");

} else {

char buf[500];

int i=0;

char \*pch;

while(!feof(stream))

{

fgets(buf, 500, stream);

}

pch = strtok(buf,",");

// получим список координат

//в массиве coord[i]

while (pch != NULL)

{

coord[i]=atoi(pch);

//printf("%d - ", coord[i]);

i++;

pch = strtok(NULL, ",");

}

// получен массив координат из файла

/\* инициализация переменных для отрисовки линий на график \*/

label:

// нарисуем полукруг с осями

// ось Y

drawLineScreen(240, 271, 240, 0, RGB(100,100,100));

// ось X

drawLineScreen(60, 270, 420, 270, RGB(100,100,100));

// полукруг равнй 150 см дистанции

int j;

drawCircle(240, 270, 150\*1.5, RGB(50,50,50));

drawCircle(240, 270, 100\*1.5, RGB(50,50,50));

drawCircle(240, 270, 50\*1.5, RGB(50,50,50));

drawCircle(240, 270, 10\*1.5, RGB(50,50,50));

drawFilledCircle(240, 270, 5, RGB(255,0,0));

int k,x0,y0,x1,y1;

x0=0; x1=20; y0=150; y1=150; k=0;

int tmp=0;

// нарисуем координаты на полуокружность

double A, B;

k=0;

x0=240;

y0=270;

int x2,y2;

x1=240-150\*1.5;

y1=270;

double corn=0; // угол поворота

while(k<129)

{

// вычисляем А

A=sin(corn)\*coord[k];

// вычисляем B

B=cos(corn)\*coord[k];

//printf("%f | ", A);

k++;

corn=corn+0.024;

//putPixelScreen(RGB(255,0,0), x0-(int) B\*1.5, y0-(int) A\*1.5);

x2=x0-(int) B\*1.5;

y2=y0-(int) A\*1.5;

if (coord[k]>149 || y2>265) {

putPixelScreen(RGB(255,0,51), x0-(int) B\*1.5, y0-(int) A\*1.5);

} else {

drawLineScreen(x1, y1, x2, y2, RGB(255,0,51));

}

x1=x2;

y1=y2;

}

// перерисуем линию

drawLineScreen(240, 271, radarX, radarY, RGB(255,0,204));

flipScreen();

clearScreen(RGB(255,255,255));

int bit=0;

while(!done)

{

sceDisplayWaitVblankStart();

sceCtrlReadBufferPositive(&pad, 1);

if (pad.Buttons != 0){

if (pad.Buttons & PSP\_CTRL\_LEFT){

// тащим линию влево

// тоесть меняем координату x-5 например

if(radarX-5<0) radarX=480;

radarX=radarX-5;

}

if (pad.Buttons & PSP\_CTRL\_RIGHT){

// тащим вправо

if (radarX+5>480){

radarX=0;

}

radarX=radarX+5;

}

if (pad.Buttons & PSP\_CTRL\_UP){

if(radarY-5<0) radarY=270;

radarY=radarY-5;

}

if (pad.Buttons & PSP\_CTRL\_DOWN){

if(radarY+5>270) radarY=0;

radarY=radarY+5;

}

if (pad.Buttons & PSP\_CTRL\_TRIANGLE){

// загрузка нового файла

}

}

goto label;

}

}

sceKernelExitGame();

return 0;

}

# Приложение 4. Подключение к точке доступа и получение IP адреса

# Приложение 5. Исходный текст программы «Бугунок»

#include <malloc.h>

#include <pspkernel.h>

#include <pspdebug.h>

#include <pspdisplay.h>

#include <pspdebug.h>

#include <pspctrl.h>

#include <pspgu.h>

#include <pspgum.h>

#include <psprtc.h>

#include <stdio.h>

#include "graphics.h"

/\* Define the module info section \*/

PSP\_MODULE\_INFO("Radar Example", 0, 1, 1);

typedef unsigned char byte;

#define RGB(r,g,b) ((u32)((byte)(r)|((byte)(g) << 8)|((byte)(b) << 16)))

/\* Define the main thread's attribute value (optional) \*/

PSP\_MAIN\_THREAD\_ATTR(THREAD\_ATTR\_USER | THREAD\_ATTR\_VFPU);

/\* Define printf, just to make typing easier \*/

#define printf pspDebugScreenPrintf

void dump\_threadstatus(void);

int done = 0;

/\* Exit callback \*/

int exit\_callback(int arg1, int arg2, void \*common)

{

done = 1;

return 0;

}

/\* Callback thread \*/

int CallbackThread(SceSize args, void \*argp)

{

int cbid;

cbid = sceKernelCreateCallback("Exit Callback", exit\_callback, NULL);

sceKernelRegisterExitCallback(cbid);

sceKernelSleepThreadCB();

return 0;

}

/\* Sets up the callback thread and returns its thread id \*/

int SetupCallbacks(void)

{

int thid = 0;

thid = sceKernelCreateThread("update\_thread", CallbackThread,

0x11, 0xFA0, 0, 0);

if(thid >= 0)

{

sceKernelStartThread(thid, 0, 0);

}

return thid;

}

int main(void)

{

SetupCallbacks();

pspDebugScreenInit();

initGraphics();

sceCtrlSetSamplingMode(PSP\_CTRL\_MODE\_ANALOG);

SceCtrlData pad;

int x=0, y=0;

while(1) {

sceDisplayWaitVblankStart();

sceCtrlReadBufferPositive(&pad, 1);

if (pad.Buttons & PSP\_CTRL\_UP) {

if (y-1 < 0){

y=272;

printf("y=0");

}

y -= 1;

}

if (pad.Buttons & PSP\_CTRL\_DOWN)

{

if (y+1 > 272){

y=0;

printf("y=0");

}

y += 1;

}

if (pad.Buttons & PSP\_CTRL\_LEFT) x -= 1;

if (pad.Buttons & PSP\_CTRL\_RIGHT) x += 1;

drawLineScreen(x + 10, y + 10, x + 20, y + 20,

RGB(255, 255, 255));

drawLineScreen(x + 10, y + 20, x + 20, y + 10,

RGB(255, 255, 255));

flipScreen();

clearScreen(RGB(0, 0, 0));

}

sceKernelExitGame();

return 0;

}

# Приложение 6. Исходный код сервера обработки входящих запросов на ПК

#include <sys/types.h>

#include <winsock2.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#pragma comment(lib, "Ws2\_32.lib")

int main( void )

{

struct sockaddr\_in local;

int s;

int s1;

int rc;

char buf[ 1 ];

char str;

char comm, lo, hi, data;

int err;

WSADATA wsa\_data;

err = WSAStartup (MAKEWORD(2,2), &wsa\_data);

local.sin\_family = AF\_INET;

local.sin\_port = htons( 5555 );

local.sin\_addr.s\_addr = htonl( INADDR\_ANY );

// создание сокета на 5555 порту

s = socket( AF\_INET, SOCK\_STREAM, 0 );

if ( s < 0 ){

perror( "socket failed" );

exit( 1 );

}

rc = bind( s, ( struct sockaddr \* )&local, sizeof( local ) );

if ( rc < 0 ){

perror( "bind failure" );

exit( 1 );

}

while(1){

// слушаем сокет

rc = listen( s, 5 );

if ( rc ){

perror( "listen call failed" );

exit( 1 );

}

// принимаем соединение

s1 = accept( s, NULL, NULL );

if ( s1 < 0 ){

perror( "accept call failed" );

exit( 1 );

}

else{

printf("Connect!\n");

}

//обработка входящих сообщений

while(1){

// прием данных

rc = recv( s1, buf, 1, 0 );

printf( "\nRecv message: %d", buf[ 0 ] );

break;

}

}

exit( 0 );

}

# Приложение 7. Фрагмент исходного текста приложения netsample

int create\_socket()

{

int sock;

struct sockaddr\_in echoserver;

char buffer[1024];

unsigned int echolen;

int received = 0;

int snd\_flag=0; // флаг для контроля отправки данных

char buf;

char send\_byte[]={27, '\0'};

char snd[1];

char recv\_byte[1];

/\* Create the TCP socket \*/

if ((sock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP)) < 0) {

printf("Failed to create socket");

} else {

printf("\n Socket created");

}

/\* Construct the server sockaddr\_in structure \*/

memset(&echoserver, 0, sizeof(echoserver)); /\* Clear struct \*/

echoserver.sin\_family = AF\_INET; /\* Internet/IP \*/

echoserver.sin\_addr.s\_addr = inet\_addr(SERVER\_IP); /\* IP address \*/

echoserver.sin\_port = htons(5555); /\* server port \*/

/\* Establish connection \*/

if (connect(sock, (struct sockaddr \*) &echoserver,

sizeof(echoserver)) < 0) {

printf("\nFailed to connect with server");

} else {

printf("\nConnected...");

}

char esc=27, hiAddr=0x00, loAddr=0x00;

// send ESC

send(sock, &esc, 1, 0);

printf("\n sented");

close(sock);

return 0;

}

# Приложение 8. Фрагмент исходно кода программы сервера реализующего протокол

int create\_socket()

{

int sock;

struct sockaddr\_in echoserver;

char buffer[1024];

unsigned int echolen;

int received = 0;

int snd\_flag=0; // флаг для контроля отправки данных

char buf;

char send\_byte[]={27, '\0'};

char snd[1];

char recv\_byte[1];

/\* Create the TCP socket \*/

if ((sock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP)) < 0) {

printf("Failed to create socket");

} else {

printf("\n Socket created");

}

/\* Construct the server sockaddr\_in structure \*/

memset(&echoserver, 0, sizeof(echoserver)); /\* Clear struct \*/

echoserver.sin\_family = AF\_INET; /\* Internet/IP \*/

echoserver.sin\_addr.s\_addr = inet\_addr(SERVER\_IP); /\* IP address \*/

echoserver.sin\_port = htons(5555); /\* server port \*/

/\* Establish connection \*/

if (connect(sock, (struct sockaddr \*) &echoserver,

sizeof(echoserver)) < 0) {

printf("\nFailed to connect with server");

} else {

printf("\nConnected...");

}

char esc=27, hiAddr=0x00, loAddr=0x00;

// Разделим адрес на старший и младший

hiAddr = memAddr / 0x100;

loAddr = memAddr % 0x100;

// send ESC

send(sock, &esc, 1, 0);

printf("\n sented");

// receive 80h

recv(sock, &buf, 1, 0);

printf("\nReceive: %X", buf);

// send Command

snd\_flag=send(sock, &comm, 1, 0);

// send hiAddr

snd\_flag=send(sock, &hiAddr, 1, 0);

// send loAddr

snd\_flag=send(sock, &loAddr, 1, 0);

// если это команда записи, то отправим данные

if (comm==writeMem or comm==writeRam) {

snd\_flag=send(sock, &data, 1, 0);

// если команда чтения, то примем данные

} else if (comm==readMem or comm==readRam) {

snd\_flag=recv(sock, &buf, 1, 0);

// значит команда старта ??? что отправлять, адрес откуда стартовать?

} else {

snd\_flag=send(sock, &data, 1, 0);

}

// проверка на отправку

if (!snd\_flag) {

printf("\n [Error] When send bytes");

}

// receive info

recv(sock, &buf, 1, 0);

printf("\nReceive: %X", buf);

close(sock);

return 0;

}

1. Определяется наличием зоны покрытия опетатора сотовой связи [↑](#footnote-ref-1)
2. Среднее энергопотребление модулей приёмопередатчиков, обеспечивающих реализацию представленных технологий [↑](#footnote-ref-2)
3. По стандарту 802.11g 54Мбит/с, по стандарту 802.11n до 600 Мбит/с с полосой частот шириной 40 МГц [↑](#footnote-ref-3)
4. При увеличении количества подключений скорость пропорционально уменьшается, других ограничений на данный момент стандарт не предусматривает [↑](#footnote-ref-4)
5. Ограничения на количество подключений зависят от способности принимающего устройства диспетчеризировать запросы, от ёмкости базовой станции [↑](#footnote-ref-5)
6. В этом алгоритме строится дуга окружности для первого квадранта, а координаты точек окружности для остальных квадрантов получаются симметрично. На каждом шаге алгоритма рассматриваются три пикселя, и из них выбирается наиболее подходящий путём сравнения расстояний от центра до выбранного пикселя с радиусом окружности. [↑](#footnote-ref-6)
7. D — [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения, предназначенный для [прикладного](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [системного программирования](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Разработан [Уолтером Брайтом](http://ru.wikipedia.org/wiki/%D0%A3%D0%BE%D0%BB%D1%82%D0%B5%D1%80_%D0%91%D1%80%D0%B0%D0%B9%D1%82" \o "Уолтер Брайт) и его фирмой [Digital Mars](http://ru.wikipedia.org/wiki/Digital_Mars" \o "Digital Mars). D является [языком высокого уровня](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%B2%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F), но сохраняет возможности прямого взаимодействия с [программным интерфейсом](http://ru.wikipedia.org/wiki/API) [операционной системы](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и с оборудованием [↑](#footnote-ref-7)