Предварительная версия текста

**Вступление**

Добрый день, мы студенты Московского Авиационного Института, кафедры 307. (Представиться?). Сегодня мы расскажем вам о нашем проекте – разработке прототипа системы управления автономным роботом на базе Kaspersky OS.

**Слайд 2. Актуальность, цель работы.**

Всё более широкое применение начинают приобретать автономные роботы. Например, можно уже сейчас увидеть на дорогах роботов-доставщиков, производящих доставку еды и не только. Кроме уже реализованных решений существует большое количество ещё не автоматизированных задач , которые можно делегировать роботам.

Целью нашей работы является разработка автономной системы управления роботами для их навигации в закрытых помещениях. Разрабатываемая система должна обладать достаточной гибкостью для настройки под конкретную задачу и обеспечивать хорошую защиту от взлома.

**Слайд 3. Описание системы. КТС.**

Для достижения поставленной цели спроектирована информационная система, состоящая из сервера, камер видеонаблюдения, подключенных через Ethernet и мобильных роботов, подключенных к серверу по Wi-Fi. Сервер обрабатывает видеопоток, поступающий с камер, находящихся на стенах помещения, распознает положение роботов и направляет им соответствующие команды.

В ходе разработки мы ориентировались на роботов на платформе Raspberry PI. Связь между сервером и роботами осуществляется по протоколу TCP/IP. Робот принимает команду, декодирует её и выполняет. Потенциально в данной системе можно использовать MQTT, что позволит возможно вести лог команд и сделать их более комфортными для восприятия человеком. Сервер работает на базе операционной системы Linux, однако его можно запустить и на KOS, что позволит повысить безопасность. В любом случае роботы находятся под управлением операционной системы KasperskyOS. Её мы выбрали как раз потому, что это отечественная кибериммунная система, а именно….

**Слайд 4. Решение проблемы должной безопасности. Почему именно KOS?**

Данная операционная система вместила в себя максимум тех разработок в сфере безопасности, которые были доступны на текущий момент. Она основана на “трех слонах” – архитектуре MILS, архитектуре FLASK и других дополнениях от команды Kaspersky. Благодаря этому она является достаточно защищенной от известных уязвимостей и хорошо подходит для решений в сфере IoT.

Архитектура MILS предполагает разделение…

**Слайд 5. VFS и GPIO**

**Слайд 6. Диаграмма последовательностей**

На данной диаграмме представлена последовательность взаимодействия робота с сервером. Когда робот включается, он пробует подключиться по заданной сети к серверу. Затем производится аутентификации и установка подключения. Когда канал передачи готов, сервер обрабатывает снимки с камер, находит данного робота, отправляет ему команду. Затем робот выполняет команду, тем временем сервер продолжает обрабатывать снимки, отправляет новую команду и так далее.

**Слайд 7. Диаграмма классов.**

На данном слайде представлено программное устройство робота. Каждый робот содержит в себе Command Receiver и некоторое количество моторов, которые приводят его в движение. Для более удобной отладки мы создали по шаблону проектирования Strategy 2 типа двигателей. Console Engine подразумевает то, что работа мотора будет отображена в виде вывода в консоль. GPIO Engine подразумевает отправку команды через GPIO на реальный мотор робота.

Аналогично для удобства отладки были реализованы классы CommandReceiver и мы можем переключаться между ручным вводом команд и принятием команд от сервера по TCP/IP.

**Слайд 8. Выводы по имеющемуся.**

Подводя итоги, в результате работы представлен рабочий прототип системы, который демонстрирует возможность разработки защищенных управляющих приложений, использующих FLASK архитектуру для обеспечения контроля выполнения политики безопасности.

**Слайд 9. Перспективный взгляд на проект.**

В будущем мы планируем и далее развивать систему. А именно конечной целью мы подразумеваем создание абсолютно гибкой системы. Она позволит не только применять её для управления реальными роботами, но и использовать её модификацию для обучения студентов или специалистов в IoT сфере.

Конкретно это будет происходить так: мы планируем подключить существующий прототип к игровому движку, например UE4, в нем у нас будет имитация камеры и самого робота. Модель камеры будет передавать данные на сервер, сервер будет их обрабатывать, посылать на робота команды, робот (а именно стимуляция Raspberry PI под QEMU) будет отрабатывать абсолютно так же, как бы это делал реальный робот, однако при отправке сигнала на двигатели, сигнал пойдёт на их модели, которые будут передавать среде UE4, как быстро и насколько едет робот. Тем самым мы сможем создавать различные виртуальные окружения и в том числе тестировать на нем нашу систему.

**Заключение**

На этом все. Спасибо за внимание. Мы готовы ответить на ваши вопросы.