МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Руководитель:
Ассистент кафедры вычислительной техники
Волошин А.В.
_____
(подпись)
«___» _____ 20__ г.

К защите допустить:
Доцент кафедры ИБТКС
к.т.н., А.П. Плёнкин
_____
(подпись)
«___» _____ 20__ г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ТВОРЧЕСКОМУ ПРОЕКТУ

ПО ДИСЦИПЛИНЕ «ВвИД»

на тему: Разработка программно-аппаратного комплекса шагающего робота

Команда: Build N Learn

Выполнили:

Ледерер Пётр Алекссевич, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Кочубей Даниил Сергеевич, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Жалнин Дмитрий Игоревич, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Пучкова Анастасия Денисовна, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Калитин Алексей Вячеславович, КТбо2-6
(подпись, фамилия, имя, отчество, группа)

Москаленко Андрей Сергеевич, КТбо2-6
(подпись, фамилия, имя, отчество, группа)

Неприн Михаил Андреевич, КТбо2-1
(подпись, фамилия, имя, отчество, группа)

Таганрог 2021 г.

Реферат

Пояснительная записка содержит 60 страниц, 18 рисунков, 4 таблицы и 8 источников.

Суть проекта заключается проектировке мобильного робота и разработке информационно-управляющей системы, состоящей из алгоритмов управления и программного обеспечения для шагающего робота.

Результатом выполнения проекта является демонстрационный прототип мобильного робота.

Целью работы является изучение высокоуровневых языков программирования Python, Java, C++.

Содержание

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Спроектировать мобильного робота и разработать информационно-управляющую систему. Максимальный размер роботизированной платформы —40см х 50см х 50см;

1. Кинематика базового шасси и программно-аппаратный комплекс системы управления выбирается командой самостоятельно без каких-либо ограничений, за исключением ограничений безопасности;
2. Управление роботом осуществляется посредством использования технологий Bluetooth или WI-FI.
3. Использовать высокоуровневые объектно-ориентированный языки программирования C++,Python, Java.

Результат выполнения проекта: демонстрационный прототип шагающего робота.

ВВЕДЕНИЕ

Динамичный темп жизни современного человека с каждым годом набирает все большие обороты. Работа, учёба, семья, спорт – в круговороте событий сложно найти свободную минуту. Если раньше поход в ресторан или кафе можно было назвать особым событием, то сегодня это уже утратило прежнюю уникальность и стало частью нашей повседневной жизни. Также угас интерес посетителей к индустрии питания. Проще и привычней для людей стала доставка на дом. Качественного сервиса и гибкой ценовой политики уже недостаточно для привлечения клиентов. Перед владельцами ресторанного бизнеса встала непростая задача поиска новых стратегий повышения спроса на свои услуги.

Продукт нашего проекта способен решить данную задачу. Нашей целью является создание прототипа робота-официанта, который сможет полностью заменить человека. Прообразом нашего робота стал главный герой анимационного научно-фантастического фильма «ВАЛЛ-И». Полюбившийся многим зрителям трогательный образ может служить яркой рекламой ресторана, а функциональность робота-официанта позволит повысить эффективность работы персонала и снизить трудозатраты.

Кроме того, после введения режима самоизоляции в обществе появилась острая потребность свести контакт людей друг с другом к минимуму, которую способен удовлетворить наш умный официант.

1 РАСПРЕДЕЛЕНИЕ РОЛЕЙ

Только слаженные действия всех членов команды могут привести к успеху команды в целом. Выбор ролей для каждого её члена является очень важной и ответственной задачей. Каждый участник проекта должен чётко понимать, какие функции он должен выполнять, какой продукт на выходе получить, на каком этапе сейчас находится проект. Роли должны быть распределены так, чтобы участники команды могли дополнять друг друга, помогать друг другу, таким образом достигая поставленных целей. Если команда разобщена, проект обречён.

Список участников команды с соответствующими им ролями представлен на Таблице 1.

Таблица 1 - распределение ролей

ФИО	Роль в команде
Кочубей Даниил Сергеевич	Программист-конструктор
Ледерер Пётр Алексеевич	Программист-конструктор (капитан команды)
Жалнин Дмитрий Игоревич	back-end разработчик
Пучкова Анастасия Денисовна	Дизайнер, менеджер
Калитин Алексей Вячеславович	Инженер-программист
Неприн Михаил Андреевич	Программист
Москаленко Андрей Сергеевич	Программист

2 АППАРАТНАЯ ЧАСТЬ

Проект реализован на базе микрокомпьютера Raspberry Pi 4 , изображённой на рисунке 1, обеспечивающего достаточно ресурсов для разработки при низкой потребляемой мощности и удовлетворительном быстродействии для поставленной задачи. Микрокомпьютер занимается опросом датчиков, получением с них необходимой информации и обеспечивает связь между ними. Далее все электронные модули будут рассмотрены подробнее.

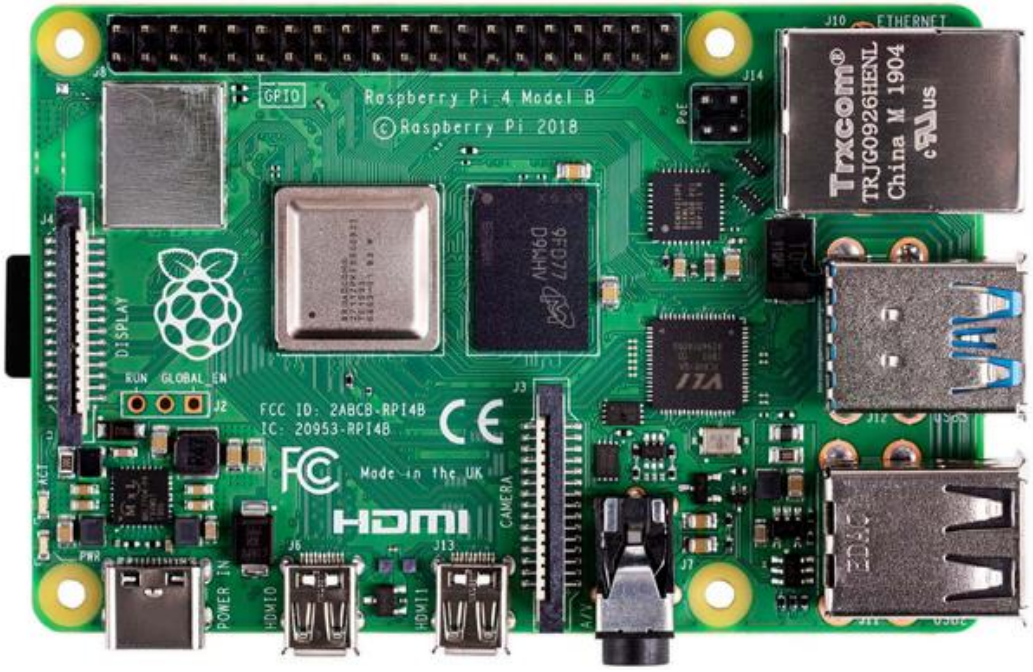

Рисунок 1 – Внешний вид Raspberry Pi 4

2.1 Принципиальная схема устройства

Перед непосредственной сборкой устройства была разработана принципиальная схема, компоновки электронных компонентов в системе Fritizing, на рисунке 2 показана финальная версия устройства.

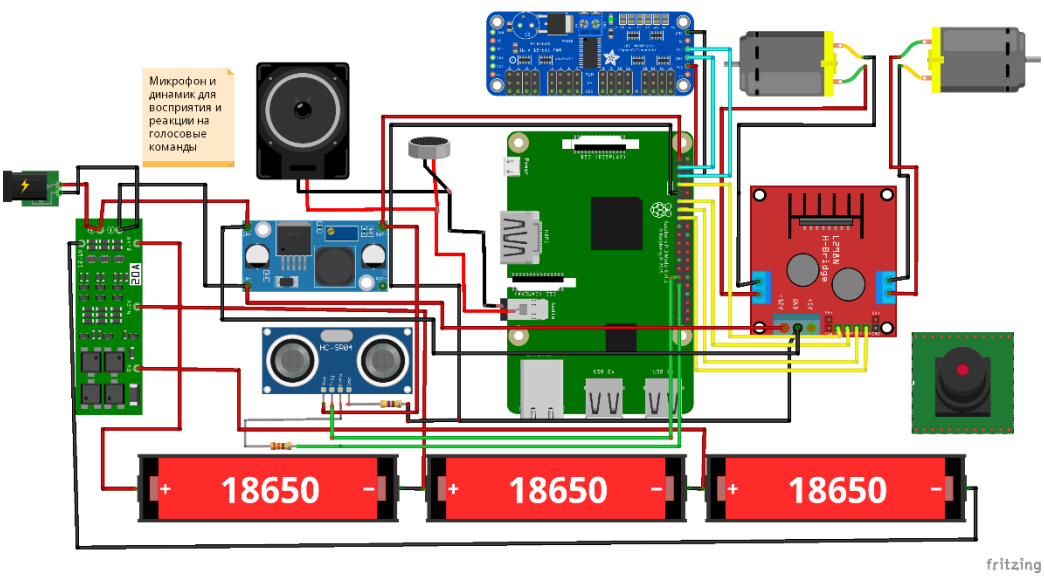

Рисунок 2 – Принципиальная схема

2.2 Стабилизатор напряжения LM2596

Основной модуль системы питания робота. Напряжение выдаваемое 3-мя аккумуляторами формата 18650 приблизительно равно 12 вольтам. LM2596 присутствует в устройстве для понижения питающего напряжения до 5 вольт - номинала требуемого Raspberry Pi. Внешний вид платы представлен на рисунке 3.

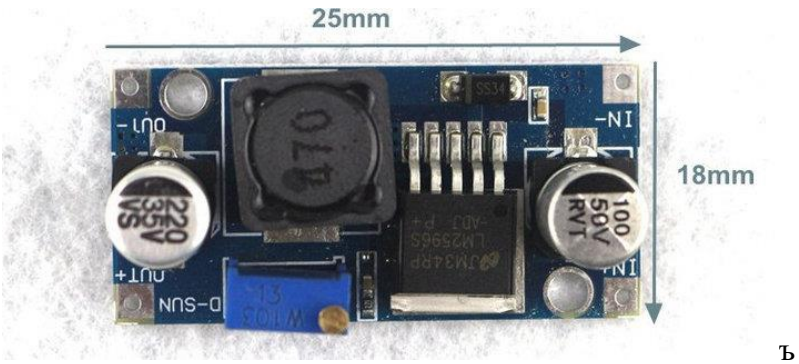

Рисунок 3 – Внешний вид LM2596

2.3 ШИМ контроллер PCA9685

Так как робот имеет большое количество встроенных сервоприводов, для их контроля требуется специализированная плата, такая как PCA9685. Она подключается к Raspberry Pi по протоколу I2C и позволяет управлять сразу 16-ю устройствами работающими с ШИМ. Внешний вид платы представлен на рисунке 4.

Рисунок 4 – Внешний вид PCA9685

2.4 Контроллер заряда с защитой BMS 3S

Так как питание робота осуществляется с помощью аккумуляторов формата 18650, для их зарядки, а так же контроля перегрузки и короткого замыкания, в роботе присутствует плата защиты BMS 3S. Она позволяет снимать с аккумуляторов до 20 А тока, чего с запасом хватает роботу для работы в течение 2 часов в интенсивном режиме. Внешний вид платы представлен на рисунке 5.

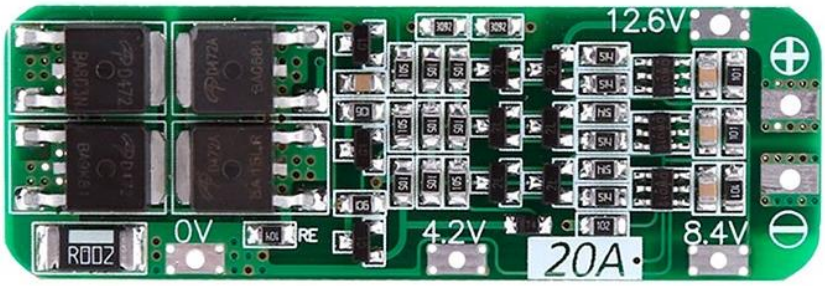

Рисунок 5 – Внешний вид 3S BMS

2.5 Ультразвуковой датчик HC-SR04

Ультразвуковой датчик HC-SR04 способен измерять расстояние и обнаруживать предметы в диапазоне от 2 сантиметров до 4метров. В роботе он служит для определения расстояния до препятствий при автоматическом режиме работы. Если расстояние до объекта меньше критического, то робот остановится и столкновение не произойдёт. Внешний вид платы представлен на рисунке 6.

Рисунок 6 – Внешний вид 3S BMS

2.6 Драйвер двигателей L298N

Для управления электродвигателями постоянного тока в гусеничном шасси используется драйвер двигателей L298N. Он позволяет с помощью широтно-импульсной модуляции регулировать скорость и направление вращения моторов. Внешний вид платы представлен на рисунке 7.

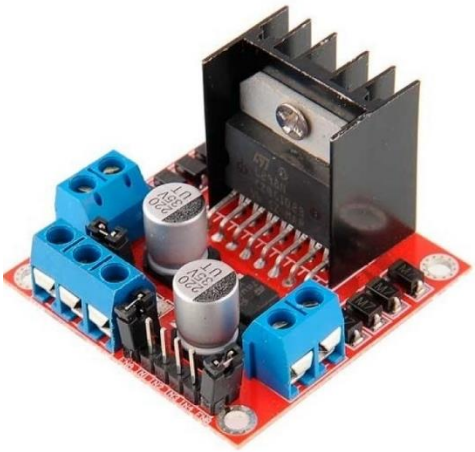

Рисунок 7 – Внешний вид L298N

2.7 Мотор-редукторы на гусеничном шасси

Служат для передвижения робота, номинальное рабочее напряжение 12 вольт. На каждом из моторов так же установлен редуктор, повышающий тягу двигателя за счёт уменьшения числа оборотов. Внешний вид шасси представлен на рисунке 8.

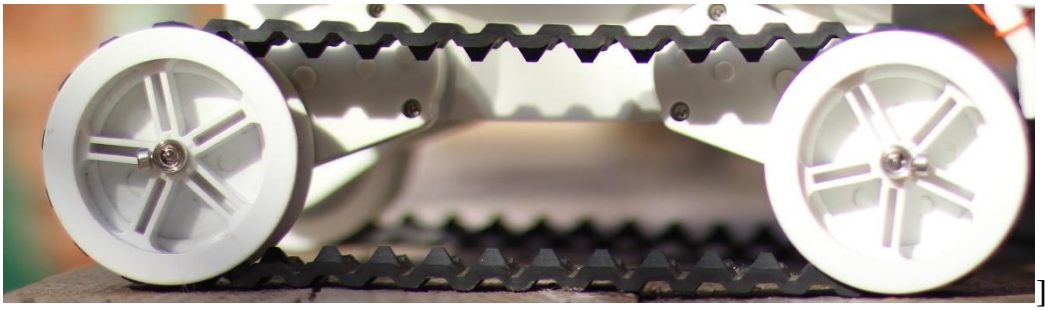]

Рисунок 8 – Внешний вид мотор-редукторов

2.8 Веб-камера Logitech C270

Для организации видео-стрима, а так же автономной работы в роботе присутствует камера Logitech c270. Внешний вид устройства представлен на рисунке 9.

Рисунок 9 – Внешний вид веб-камеры Logitech c270

2.9 Сервоприводы SG90 и Goteck HB2411T

В роботе установлено 8 сервоприводов, они служат для:

1. Поворота рук
2. Активации захвата пальцем
3. Управления углом наклона глаз
4. Управления шейными сервоприводами

В проекте использовалось 2 вида сервоприводов. Для управления руками были выбраны более мощные сервомоторы, так как они должны поднимать больший вес. Оба вида моторов представлены на рисунке 10.

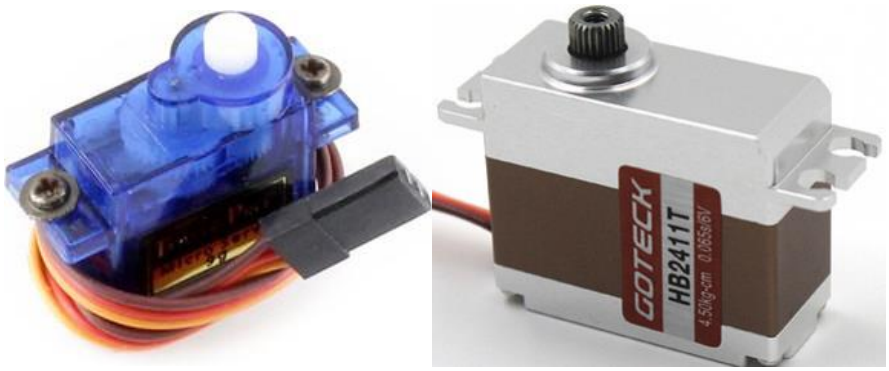

Рисунок 10 – Внешний вид SG90 и Goteck HB2411T

2.10 Корпус робота

Корпус сделан из двух основных материалов – PLA пластик и оргстекло(акрил). Руки и глаза были напечатаны на 3д принтере. На рисунке 11 изображена окончательный вид робота.

Рисунок 11 – Робот в сборе

3 ПРОГРАММНАЯ ЧАСТЬ

По причине наличия большого количества аппаратных систем в составе робота, потребовалось реализовать множество алгоритмов для обработки данных с датчиков и камеры, а так же для управления механической частью всего комплекса. Так как полученная информация с датчиков имеет различный формат, то к процессу обработки отдельных потоков информации применялись разные подходы, так же все механические составляющие робота требуют отличные друг от друга управляющие сигналы, что тоже увеличило количество отдельных программных подсистем проекта. Программная архитектура системы представлена на рисунке 12.

Рисунок 12 – Архитектура системы

3.1 Robot operation system – ядро системы

Сердцем программной части проекта стал фреймворк R.O.S., представляющий из себя подобие клиент-серверной системы, в которой различные программные составляющие робота выступают в качестве клиентов(узлов), работающих независимо друг от друга, и обмениваются между собой необходимыми данными по отдельным независимым каналам(топикам). Главным преимуществом данной системы является независимость отдельных узлов друг от друга, что в случае неполадок в одном модуле не влияет на работоспособность всех других. Так же фреймворк поддерживает многопоточность, что позволяет при необходимости выделять больше вычислительных мощностей на отдельные задачи. В приложении А предоставлен код всех узлов системы, связанных в R.O.S..

3.2 Сервер

Сервер является связующей частью между несколькими программными составляющими проекта. Основной задачей является получение сообщений от административного мобильного приложения, обработка и формирование различных управляющих сигналов для других узлов. Особенностью обработки является распределение сообщений по разным потокам для разных подпрограмм, что позволяет сделать управление механическими составляющими независимым друг от друга.

3.3 Система распознавания и синтеза голоса

Скрипты были написаны на Python. Для реализации распознавания голоса была использована библиотека SpeechRecognition. С помощью неё, полученная с микрофона запись преобразуется в текст, и далее скрипт распознает знакомые слова и в зависимости от запроса пользователя отвечает на команду подготовленным ответом, который воспроизводится с помощью espeak и SpeechSynthesiser.

В роботе установлена веб-камера со встроенным микрофоном, которая и используется для системы распознавания голоса. Код предоставлен в приложении Б.

3.4 Компьютерное зрение и цветовая сегментация

Для ориентирования робота в пространстве при автоматическом режиме работы он оснащён системой компьютерного зрения на основе python-библиотеки OpenCV.

Логика работы следующая: робот ищет цветовую метку, закреплённую на столе, с которого поступил заказ, после чего едет к ней. Для нахождения квадратной метки определенного цвета были использованы черно-белые маски, они накладываются на квадратные контуры и выделяющие их координаты. При попадании центра квадрата правее вертикальной оси изображения робот должен подворачивать вправо, также и для случая с левосторонним расположением объекта. Для того, чтобы другие предметы не мешали обнаружению, используется функция, считающая размер найденного объекта и не реагирующая на него, если он меньше порогового значения, для дополнительной проверки используется ультразвуковой дальномер. Если все условия соблюдены, то робот направляется к метке и оказавшись непосредственно перед ней, останавливается и выполняет необходимые операции с клиентом.

В результате, система формирует управляющие сигналы для механических частей робота, такие как: поворот, движение вперёд/назад, разжим/сжим захвата и тд,

4 РЕЖИМЫ РАБОТЫ РОБОТА

В роботе предусмотрено 2 режима работы – ручной и автоматический. В ручном режиме управляющие команды приходят из специального андроид приложения, оператор должен сам довести робота до нужной точки. В автоматическом режиме робот сам определяет как ему двигаться до цели с помощью системы компьютерного зрения.

4.1 Android-приложение управления

Разработка приложения началась с определения необходимого функционала и выбора языка для разработки приложения.

Для создания приложения был выбран язык Java, выбран именно он, потому что язык существует уже довольно долго и имеет большое комьюнити разработчиков, множество библиотек и готовых решений.

При запуске приложения пользователь попадает на фрагмент с пультом управления, который изображён на рисунке 13. Пульт включает в себя джойстик для изменения направления движения и скорости робота, две кнопки, которые при нажатии сжимают или разжимают руки робота, ползунки для поворота рук головы и глаз.

Рисунок 13 – Фрагмент с пультом управления

При использовании клавиш с пульта роботу посылаются сигналы с соответствующей команде буквой и параметрами. Система команд содержит 8 типов сообщений:

1. Движении джойстика - J <угол> <длина>
2. Движение левой руки - L <угол>

3. Движение правой руки - R <угол>
4. Движение глаз - E <угол>
5. Движение головы - H <угол>
6. Сжатие/разжатие правой руки -T R 1/0
7. Сжатие/разжатие левой руки -T L 1/0
8. Голосовые кнопки - V <номер кнопки>

Для управления роботом к нему необходимо подключиться, для этого был сделан фрагмент для настройки подключения к роботу изображенный на рисунке 14.

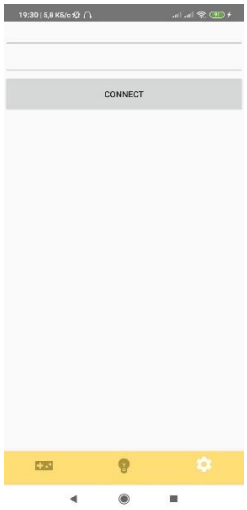

Рисунок 14 – Фрагмент для подключения к роботу

На рисунке видно, что в фрагмент входит два поля для ввода первое для адреса сервера, а второе для порта и кнопка при нажатии, которой происходит попытка подключения, при успешной попытке пользователю выводится сообщение о том, что он подключён, как показано на рисунке 15.

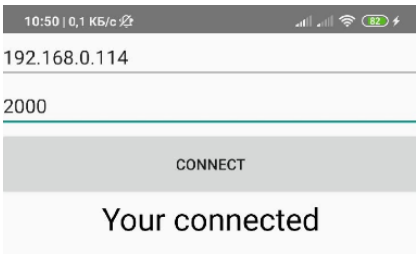

Рисунок 15 – Попытка подключения к серверу

В приложении так же есть фрагмент для управления голосовыми командами. На рисунке 16 видно, что фрагмент для управления голосовыми командами содержит девять кнопок.

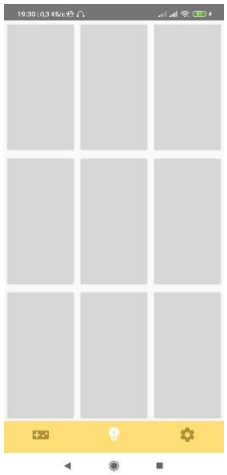

Рисунок 16 – Фрагмент для управления голосовыми командами

Все фрагменты находятся в одном активити и для перехода между фрагментами есть нижнее навигационное меню, которое содержит три кнопки (столько сколько содержит программа фрагментов). Исходный код доступен в приложении В.

4.2 Приложение для совершения заказа

Для автоматического режима работы было создано пользовательское приложение с возможностью выбора блюда и сканером qr-кодов. Сценарий использования:

1. Пользователь приходит за столик и сканирует qr-код находящийся на столе
2. Выбирает из списка нужные блюда.
3. Жмёт кнопку подтверждения и совершает заказ.

Далее приложение отправляет сигнал с кодом столика, взятым из qr-кода, и списком блюд в заказе. Робот получает данные, сопоставляет номер столика с соответствующим цветом и двигается к нему. Интерфейс приложения представлен на рисунке 17. Листинг предоставлен в приложении Г.

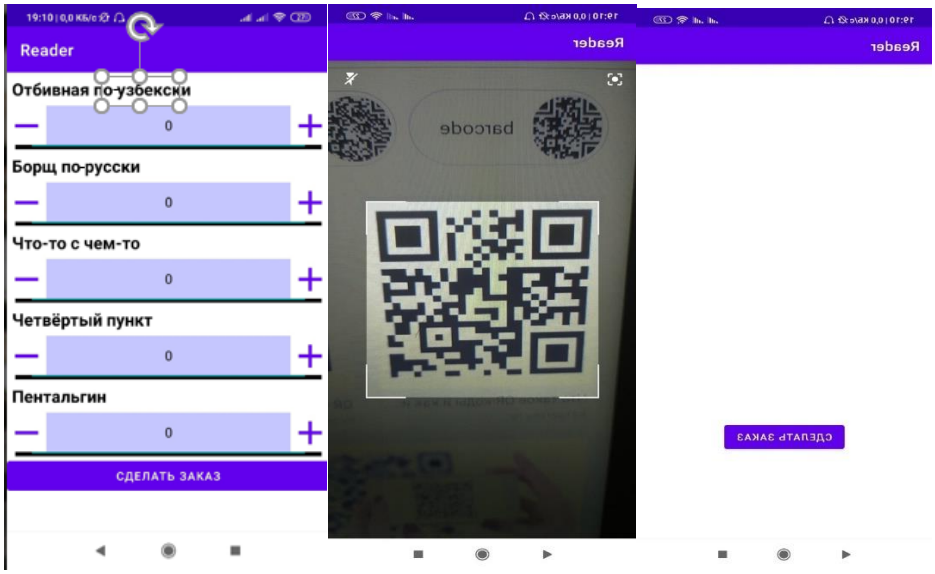

Рисунок 17 – Фрагмент для управления голосовыми командами

5 АДМИНИСТРАТИВНЫЙ САЙТ

Ещё одной частью системы является административный сайт, с возможностью просмотра стрима с камеры и доступа к командной строке ОС Linux из браузера. Сайт нужен для более удобной отладки робота и обеспечения контроля за поведением робота в автоматическом режиме работы.

5.1 Spring Framework

Spring Framework предназначен для создания стабильных защищенных веб-приложений. Фреймворк предоставляет архитектуру приложения, в которую программист может встроить свою функциональность. Еще одной особенностью является слабая связанность различных элементов приложения за счет повсеместного использования подхода внедрения зависимостей.

Спринг фреймверк разбит на модули Spring Data, Spring MVC и Spring Security. Каждый модуль настраивается независимо относительно других. Спринг дата предоставляет дружелюбный к пользователю интерфейс взаимодействия с базами данных. Этот модуль так же дает возможность работать с данными из таблицы как с объектами в объектно-ориентированном подходе программирования. Благодаря Spring Data в проекте имеется возможность быстрого развертывания без привязанности к определенной архитектуре и синтаксису базы данных, а работа с данными в коде сильно упрощается без потери стабильности.

В нашем проекте в качестве базы данных используется PostgreSQL. Для решения задач ORM используется встроенная в Spring Data библиотека Hibernate.

Spring MVC позволяет быстро и просто создавать веб-логику для веб-приложений. Он поддерживает как RESTful архитектуру, с генерацией страниц с помощью JS фреймворков на стороне клиента, так и стандартные ответы в виде шаблонизированных или нет html страниц. Для каждой страницы сайта был создан отдельный независимый контроллер GET и POST запросов от клиента. В качестве шаблонизатора для html был выбран Freemarker, который дает возможность разбивать страницы на отдельные модули/файлы, а так же "на ходу" вставлять в шаблонную страницу необходимые данные. Для оформления UI был выбран набор инструментов Bootstrap.

Spring Security предоставляет для сайта механизм авторизации, а также вводит в приложение понятие ролей с разграничением прав на использование серверных методов тех или иных частей сайта.

Список ролей содержит 3 роли: гость - неавторизированный пользователь, user - авторизованный пользователь и admin - администратор с правами управления другими пользователями. Гости имеют право просматривать только главную страницу сайта и страницу авторизации. Стандартные пользователи могут просматривать страницу с прямой трансляцией с камеры. Администраторы получают доступ к странице управления другими пользователями.

Механизм авторизации выдает каждому авторизированному пользователю токен со сроком действия авторизации, по истечению которого пользователь должен будет авторизироваться повторно. Данные для авторизации пользователей хранятся в защищенном формате. На рисунке 18 представлен скриншот страницы со стримом. В приложении Д предоставлен код системы на Spring Framework.

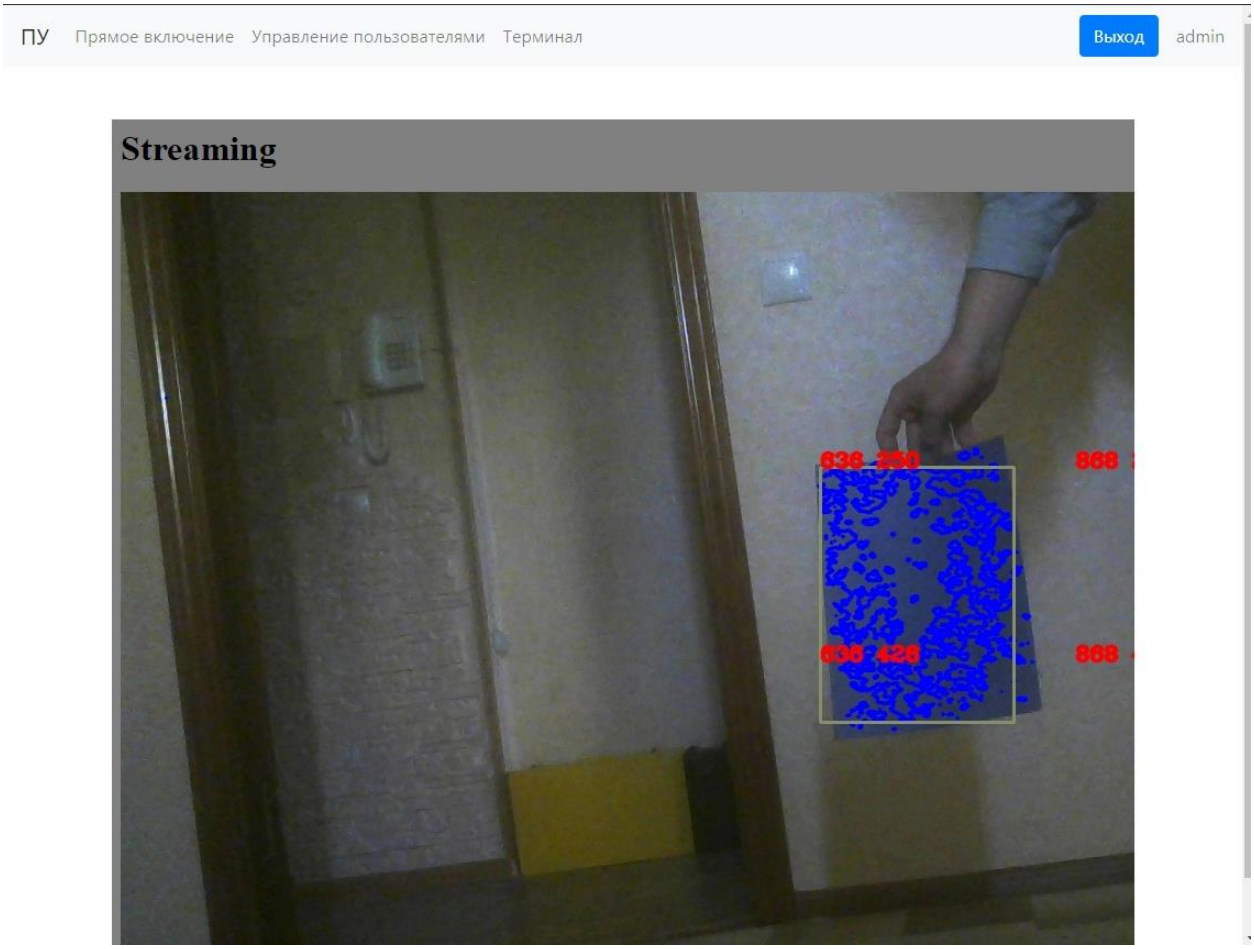

Рисунок 18 – Стрим, робот распознаёт цветовой маркер

5.2 GOTTY

Для “расшаривания” Linux-терминала в веб-интерфейс используется утилита с открытым исходным кодом GOTTY. Она написана на языке Go и предоставляет достаточную для отладки скорость работы. На рисунке 19 представлен скриншот работы системы.

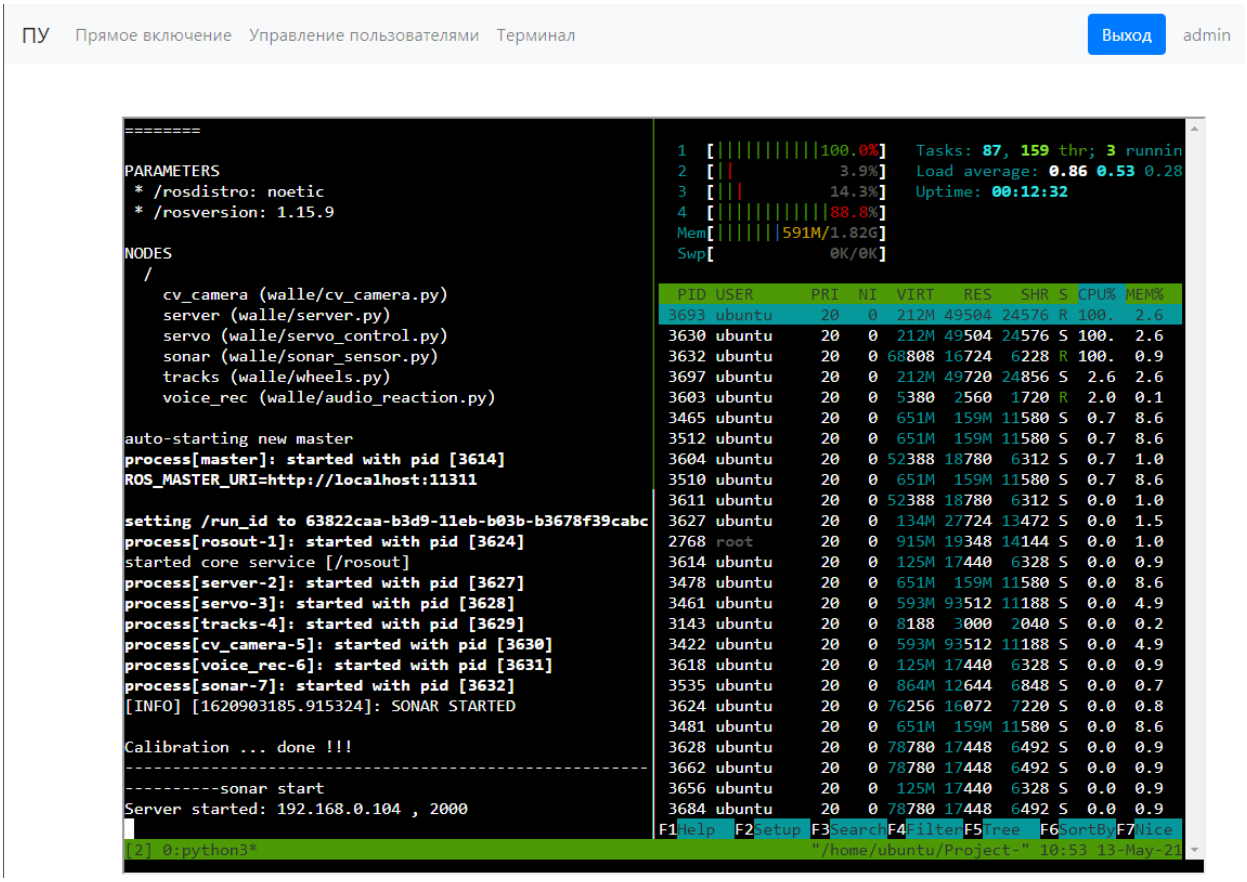

Рисунок 19 – Демонстрация работы GOTTY

5.3 Запуск сторонних систем

Так как Spring Framework и GOTTY являются самостоятельными системами, не подключёнными к ROS, для их запуска в терминале следует выполнить следующие команды:

1. gotty --port 8081-w tmux
2. mvn spring-boot:run

Их так же можно запустить из специального bash-скрипта start.sh

6 РАСЧЁТ СТОИМОСТИ РАЗРАБОТКИ

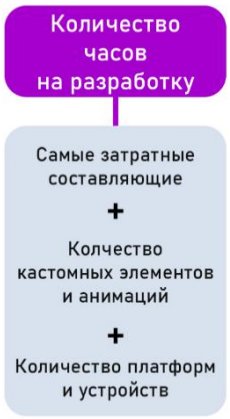

Таблица 2. Расчет количества часов на разработку

№	Название этапа разработки	Количество человек, задействованных в разработке этапа	Количество часов, потраченных на этап
1	Проработка плана работы	7	5
3	Анализ ресурсов и возможностей. Заказ плат и микрокомпьютера.	7	4
4	Создание прошивки для внедрения в микрокомпьютер и платы	6	60
5	Прошивка и работа с микрокомпьютером	6	55
6	Продумывание дизайна робота	2	21
7	Проектирование, печать, сборка корпуса робота	4	43
8	Тестирование работоспособности робота	7	37

9	Обработка и исправление ошибок	7	52
10	Приведение устройства в надлежащий вид	7	15
	ИТОГО:		292

Итогом блока «Количество часов на разработку» будет суммарное количество часов.

На данном этапе проанализирован рынок рабочей силы, занимающегося тем или иным этапом.

Таблица 3. Расчет количества часов на разработку и цена часа работы

№	Название этапа разработки	Количество человек, задействованных в разработке этапа	Количество часов, потраченных на этап	Цена часа работы (руб.)	Стоимость разработки
1	Построение архитектуры Корпуса, его изготовление	4	43	800	14000
2	Создание прошивки и прошивка устройства	6	124	800	60000
3	Тестирование и отладка системы	7	89	800	50000
	ИТОГО				1240000

Затраты на материалы и комплектующие

Этап имеет перечень основных деталей, комплектующих и материалов, задействованных в создании конечного продукта. Таблица 4.

Таблица 4. Расчет количества часов на разработку

№	Название материала/комплектующих	Цена за ед. (руб.)	Количество в готовом изделии	Общая стоимость (руб.)
1	Плата зарядки	1700	1	1700
2	Сервоприводы	1300	3	3900
3	Плата расширения для сервоприводов	760	3	2280
	ИТОГО			79800

ЗАКЛЮЧЕНИЕ

В результате выполнения творческого проекта нашей командой был создан прототип мобильного робота официанта, с автоматическим и ручным режимами работы, а так же множеством дополнительных систем призванных облегчить и сделать более удобным, как использование так и администрирование роботизированного комплекса. С актуальным исходным кодом можно ознакомится по ссылке: https://github.com/FriZIk/Project-WALLE.

Дальнейшим улучшением системы может переход на другую аппаратную платформу, так как в ходе испытаний было установлено, что мощности raspberry pi 4 не достаточно для одновременной обработки видеопотока и его трансляции в веб-интерфейс. Такой платформой может стать Jetson AGX Xavier, Так же для лучшего распознавания объектов следует использовать свёрточные нейронные сети.

СПИСОК ИСТОЧНИКОВ

1. Официальная документация по ROS // URL: http://wiki.ros.org/ru
2. Введение в Spring, или что делать, если по всему проекту @Autowired и @Component, а вы не понимаете, что это // Habr URL: https://habr.com/ru/post/455794/
3. Datasheet Raspberry Pi 4 Model B // raspberrypi.org URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi
4. Цветовая сегментация для чайников // Habr URL: https://habr.com/ru/post/332464/
5. Синтез русской речи в Linux // Habr URL: https://habr.com/ru/post/60977/
6. Ультразвуковой дальномер HC-SR04: подключение, схема и примеры работы // Амперка URL: http://wiki.amperka.ru/продукты:hc-sr04-ultrasonic-sensor-distance-module
7. Adafruit 16-Channel 12-bit PWM/Servo Driver with Raspberry Pi // learn.adafruit.com URL: https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi
8. Введение в Robot Operating System // Stepik URL: https://stepik.org/course/3222/syllabus

ПРИЛОЖЕНИЕ А. ПОДСИСТЕМЫ R.O.S.

СЕРВЕР

```
import os
import socket
import errno
import time
import rospy
from std_msgs.msg import String, Float32
from socket import error as SocketError
from playsound import playsound

import RPi.GPIO as GPIO
from adafruit_servokit import ServoKit

global moveAngle
moveAngle = 0

def on_board_systems_check():
    #pubVoice.publish("play robo-short-64")
    plstr = "/home/ubuntu/Project-
WALLE/catkin_ws/src/walle/scripts/sounds/robo-short-64.wav"
    playsound(plstr)

    mess = "L 90"
    for i in range(180):
        mess = 'L' + ' ' + str(i)
        pubServos.publish(mess)
        time.sleep(0.01)
    for i in range(180):
        mess = 'R' + ' ' + str(i)
        pubServos.publish(mess)
        time.sleep(0.01)

def publisher_setup():
```

```
            global pubVoice
            pubVoice = rospy.Publisher("voice", String, queue_size=1)
            global pubBase
            pubBase = rospy.Publisher("base", String, queue_size=16)
            global pubServos
            pubServos = rospy.Publisher("joints", String, queue_size=16)
            pubDist = rospy.Publisher("distance", String, queue_size=50)

        # Parsing the received command
        def check_parse(messArr):
            global moveAngle

            parsedLine = messArr.split(" ")

            if parsedLine[0] == 'J':
                mess = ""
                mess += parsedLine[1] + " " +  parsedLine[2]
                # mess.append(int(parsedLine[2]))
                # mess.append(int(parsedLine[1]))
                # rospy.loginfo(mess)
                if (int(parsedLine[1]) > (moveAngle+5)) or (int(parsedLin
e[1]) < (moveAngle-5)) or (int(parsedLine[2]) == 0):
                    pubBase.publish(mess)
                    rospy.loginfo(" | ".join(parsedLine))
                    if not(int(parsedLine[2]) == 0):
                        pubVoice.publish("play robo-short-56")
                # rospy.loginfo("play robo-short-64")
                moveAngle = int(parsedLine[1])
                rate.sleep()
            elif parsedLine[0] == 'L' or parsedLine[0] == 'R':
                mess = ""
                mess += parsedLine[0] + " " + parsedLine[1]
                # rospy.loginfo(mess)
                pubServos.publish(mess)
```

```
            rate.sleep()
        elif parsedLine[0] == 'T':
            mess = ""
            mess += parsedLine[0] + " " + parsedLine[1] + " " + parse
dLine[2]
            # rospy.loginfo(mess)
            pubServos.publish(mess)
            rate.sleep()
        if parsedLine[0] == 'E':
            return 1

    def main():
        port = 2000

        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("8.8.8.8", 80))
        ip = s.getsockname()[0]
        s.close()

        server = socket.socket(
            socket.AF_INET,
            socket.SOCK_STREAM,
        )

        server.bind((ip, port))
        print("Server started:",ip,',',port)

        server.listen()
        ret = 0
        while True:
            user_socket, address = server.accept()
            user_socket.send("Your connected".encode("utf-8"))
            messArr = []
            try:
```

```
            data = user_socket.recv(2048)
        except SocketError as e:
            if e.errno != errno.ECONNRESET:
                RAISE
            pass
        messArr = data.decode("utf-8")
        ret = check_parse(messArr)
        if ret == 1:
            exit(0)
        print(data.decode("utf-8"))

if __name__ == '__main__':
    rospy.init_node('server')
    global rate
    rate = rospy.Rate(150)
    print("\nCalibration ... ",end='')

    publisher_setup()
    print("done !!!")
    on_board_systems_check()

    # Start listening
    main()
```

КОМПЬЮТЕРНОЕ ЗРЕНИЕ

```
#!/usr/bin/env python3
from flask import Flask, render_template, Response
from camera import VideoCamera
import numpy as np
import cv2
import rospy
```

```
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

# greeen
# hsv_min = np.array((51,49,47), np.uint8)
# hsv_max = np.array((84,255,255), np.uint8)

# red
# hsv_min = np.array((169,97,35), np.uint8)
# hsv_max = np.array((255,255,255), np.uint8)

# blue
# hsv_min = np.array((90,134,65), np.uint8)
# hsv_max = np.array((120,209,255), np.uint8)



def gen(camera):
    while True:
        USLOVNAYA_PEREMENNAYA = 2
        RECTCOLOR = (103, 143, 134)
        RTHICK = 2
        if(USLOVNAYA_PEREMENNAYA == 0):
            hsv_min = np.array((51,49,47), np.uint8)
            hsv_max = np.array((84,255,255), np.uint8)
        elif USLOVNAYA_PEREMENNAYA == 1:
            hsv_min = np.array((169,97,35), np.uint8)
            hsv_max = np.array((255,255,255), np.uint8)
        elif USLOVNAYA_PEREMENNAYA == 2:
            hsv_min = np.array((90,134,65), np.uint8)
            hsv_max = np.array((120,209,255), np.uint8)
```

```
                BLOBSIZE_STOP = 40000
                BLOBSIZE = 700
                my_color = (0,0,255)
                crange = [0,0,0, 0,0,0]

                def checkSize(w, h):
                    if w * h > BLOBSIZE:
                        return True
                    else:
                        return False

                def stop(w,h):
                    if w*h>BLOBSIZE_STOP:
                        return True
                    else:
                        return False

                height = 720
                width = 1280
                width_center=width/2;
                img = camera.get_frame()
                hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV )
                thresh = cv2.inRange(hsv, hsv_min, hsv_max )
                contours, hierarchy = cv2.findContours(thresh.copy(), cv2
.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
                kernel = np.ones((5, 5), np.uint8)
                cv2.dilate(thresh, kernel, iterations = 1)
                cv2.erode(thresh, kernel, iterations = 1)
                cv2.drawContours(img, contours, -
1, (255, 0, 0), 2, cv2.LINE_AA, hierarchy, 0)
                cv2.drawContours(img, contours, -
1, (255, 0, 0), 2, cv2.LINE_AA, hierarchy, 2)
                center_rectangle = 2
```

```
            x,y,w,h = 1,2,3,4
            if len(contours) != 0:
                c = max(contours, key = cv2.contourArea)
                x,y,w,h = cv2.boundingRect(c)
                if checkSize(w, h):
                    # выводим его str("UP left")str("UP right")str("d
own left")str("down right")
                    cv2.rectangle(img, (x, y), (x+w, y+h), RECTCOLOR,
 RTHICK)
                    cv2.putText(img, str(x)+" "+str(y), (x,y), cv2.FO
NT_HERSHEY_COMPLEX_SMALL, 1, my_color, 2, cv2.LINE_AA)
                    cv2.putText(img, str(x+h)+" "+str(y), (x+h,y), cv
2.FONT_HERSHEY_COMPLEX_SMALL, 1, my_color, 2, cv2.LINE_AA)
                    cv2.putText(img, str(x)+" "+str(y+w), (x,y+w), cv
2.FONT_HERSHEY_COMPLEX_SMALL, 1, my_color, 2, cv2.LINE_AA)
                    cv2.putText(img, str(x+h)+" "+str(y+w), (x+h,y+w)
, cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, my_color, 2, cv2.LINE_AA)
                    center_rectangle = x + h/2;
                    if stop(w,h):
                        print("STOP")
                    else:
                        if(width_center < center_rectangle and abs(wi
dth_center-center_rectangle)>width_center/2):
                            print("GO RIGHT")
                        else:
                            if(width_center > center_rectangle and ab
s(width_center-center_rectangle)>width_center/2):
                                print("GO LEFT")
                            else:
                                print("GO FORWARD")


            ret, jpeg = cv2.imencode('.jpg', img)
            yield (b'--frame\r\n'
```

```
                    b'Content-
Type: image/jpeg\r\n\r\n' + jpeg.tobytes() + b'\r\n\r\n')

    @app.route('/video_feed')
    def video_feed():
        return Response(gen(VideoCamera()),
                        mimetype='multipart/x-mixed-
replace; boundary=frame')

    if __name__ == '__main__':
        rospy.init_node("cv_camera")
        app.run(host='0.0.0.0', debug=True)
```

УПРАВЛЕНИЕ СЕРВО-МОТОРАМИ

```
    #!/usr/bin/env python3
    """Simple test for a standard servo on channel 0 and a continuous
 rotation servo on channel 1."""
    import time
    import rospy
    from adafruit_servokit import ServoKit
    from std_msgs.msg import String

    # Set channels to the number of servo channels on your kit.
    # 8 for FeatherWing, 16 for Shield/HAT/Bonnet.
    kit = ServoKit(channels=16)
    head = 4
    handRight = 1
    handLeft = 7
    hinge = 5
    fingaLeft = 14
    fingaRight = 15

    def move_smth(data):
        rospy.loginfo(data.data)
```

```
print(data.data)
value = data.data
value = value.split(" ")


if value[0] == 'R':
    ang = int(value[1])
    portNumber = handRight
    ang = 180-ang
elif value[0] == 'L':
    ang = int(value[1])
    portNumber = handLeft
elif value[0] == 'T':
    if value[1] == 'R':
        if int(value[2]) == 1:
            ang = 180
        else:
            ang = 0
        portNumber = fingaRight
        #   ang = 180-ang
    elif value[1] == 'L':
        if int(value[2]) == 1:
            ang = 180
        else:
            ang = 0
        portNumber = fingaLeft

kit.servo[portNumber].angle = ang

# for i in range(0, 180, 2):
#   kit.servo[portNumber].angle = i
#   time.sleep(0.01)
#   print("I like to move it move it", i, end='\r')
# time.sleep(1)
```

```
    # for j in range(180, 0, -2):
    #   kit.servo[portNumber].angle = j
    #   time.sleep(0.01)
    #   print("I like to move it move it", j, end='\r')

def move_hand(handNum):
    kit.servo[handNum].angle=0
    time.sleep(2)
    kit.servo[handNum].angle = 0
    # time.sleep(0.05)
    # kit.servo[handNum].angle = 80
    # time.sleep(0.05)
    # kit.servo[handNum].angle = 80

def hand_home(arm=0, hand=0):
    if arm:
        kit.servo[handLeft].angle = 0
        kit.servo[handRight].angle = 180
    else:
        kit.servo[handLeft].angle = 180
        kit.servo[handRight].angle = 0

    if hand:
        kit.servo[fingaLeft].angle = 0
        kit.servo[fingaRight].angle = 180
    else:
        kit.servo[fingaLeft].angle = 180
        kit.servo[fingaRight].angle = 0

def listen():
    rospy.init_node("servos")
    rospy.Subscriber("joints", String, move_smth)

    rospy.spin()
```

```
if __name__ == '__main__':
    hand_home(1, 0)
    listen()
```

УЛЬТРАЗВУКОВОЙ ДАТЧИК

```
#!/usr/bin/env python3

import RPi.GPIO as gpio
import time
import sys
import signal
import rospy
from std_msgs.msg import Float32

def signal_handler(signal, frame): # ctrl + c -> exit program
        print('You pressed Ctrl+C!')
        sys.exit(0)
signal.signal(signal.SIGINT, signal_handler)

class sonar():
    def __init__(self):
        rospy.init_node('sonar')
        rospy.loginfo("SONAR STARTED")
        self.distance_publisher = rospy.Publisher('/sonar_dist',F
loat32, queue_size=1)
        self.r = rospy.Rate(150)
    def dist_sendor(self,dist):
        data = Float32()
        data.data=dist
        self.distance_publisher.publish(data)
        rospy.loginfo(dist)
```

```
        gpio.setmode(gpio.BCM)
        trig = 24 # 7th
        echo = 23 # 6th

        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        sensor=sonar()
        time.sleep(0.5)
        print ('------------------------------------------------------------
--------sonar start')
        try :
            while True :
                gpio.output(trig, gpio.LOW)
                time.sleep(0.1)
                gpio.output(trig, gpio.HIGH)
                time.sleep(0.00001)
                gpio.output(trig, gpio.LOW)
                while gpio.input(echo) == 0 :
                    pulse_start = time.time()
                while gpio.input(echo) == 1 :
                    pulse_end = time.time()
                pulse_duration = pulse_end - pulse_start
                distance = pulse_duration * 17000
                if pulse_duration >=0.01746:
                    #print('time out')
                    rospy.loginfo("timeout")
                    continue
                elif distance > 300 or distance==0:
                    #print('out of range')
                    rospy.loginfo("out of range")
                    continue
                distance = round(distance, 3)
```

```
        #print ('Distance : %f cm'%distance)
        sensor.dist_sendor(distance)

        sensor.r.sleep()

except (KeyboardInterrupt, SystemExit):
    gpio.cleanup()
    sys.exit(0)
except:
    gpio.cleanup()
```

УПРАВЛЕНИЕ ХОДОВОЙ ЧАСТЬЮ

```
import time
import RPi.GPIO as GPIO
import rospy
from std_msgs.msg import String

def setup_gpio():
    GPIO.setwarnings(False)
    GPIO.cleanup()
    GPIO.setmode(GPIO.BCM)

    GPIO.setup(17, GPIO.OUT)
    GPIO.setup(22, GPIO.OUT)
    GPIO.setup(27, GPIO.OUT)
    GPIO.setup(10, GPIO.OUT)

    GPIO.output(17, GPIO.LOW)
    GPIO.output(22, GPIO.LOW)
    GPIO.output(27, GPIO.LOW)
    GPIO.output(10, GPIO.LOW)

def move(data):
    #change data input from messArr to data.data
```

```
    rospy.loginfo(data.data)
    value = data.data
    value = value.split(" ")
    # GPIO.output(17, GPIO.LOW)
    # GPIO.output(22, GPIO.LOW)
    # GPIO.output(27, GPIO.LOW)
    # GPIO.output(10, GPIO.LOW)
    ang = int(value[0])
    amp = int(value[1])
    if int(amp) > 50:
        if ang in range(75, 105):
            GPIO.output(17, GPIO.HIGH)
            GPIO.output(22, GPIO.HIGH)

        elif ang in range(0, 75)or ang in range(286, 360):
            GPIO.output(22, GPIO.HIGH)
            GPIO.output(10, GPIO.HIGH)
        elif ang in range(106, 256):
            GPIO.output(17, GPIO.HIGH)
            GPIO.output(27, GPIO.HIGH)
        elif ang in range(256, 286):
            GPIO.output(10, GPIO.HIGH)
            GPIO.output(27, GPIO.HIGH)
    else:
        GPIO.output(17, GPIO.LOW)
        GPIO.output(22, GPIO.LOW)
        GPIO.output(27, GPIO.LOW)
        GPIO.output(10, GPIO.LOW)
    # time.sleep(0.005)

def listen():
    rospy.init_node("wheels")

    rospy.Subscriber("base", String, move)
```

```
        rospy.spin()

    if __name__ == '__main__':
        setup_gpio()
        listen()
```

ЗВУКВЫЕ РЕАКЦИИ СИСТЕМЫ

```
    #!/usr/bin/env python3
    import rospy
    from std_msgs.msg import String
    from playsound import playsound
    from espeak import espeak

    def say(data):
        message = data.data
        rospy.loginfo(data.data)
        spMessage = message.split(" ")
        rospy.loginfo(spMessage)
        if spMessage[0] == "talk":
            pass
        elif spMessage[0] == "play":
            plstr = "/home/ubuntu/Project-
WALLE/catkin_ws/src/walle/scripts/sounds/"+ spMessage[1]+".wav"
            rospy.loginfo(plstr)
            playsound(plstr)


    def do():
        rospy.Subscriber("voice", String, say)
        rospy.spin()
    if __name__ == '__main__':
        rospy.init_node('voice_rec')
        do()
```

ПРИЛОЖЕНИЕ Б. СИСТЕМА РАСПОЗНАВАНИЯ И СИНТЕЗА ГОЛОСА.

MAIN

```
from speech_recog.speech_recog import SpeechRecognizer
from speech_synthesis.speech_synthesis import SpeechSynthesiser
from commands import commandControl

sr = SpeechRecognizer()
sr.calibrate()

ss = SpeechSynthesiser()

cmdCtrl = commandControl(ss, sr)
#INSTALL ESPEAK, FLACK
# sudo apt update
# sudo apt install python-espeak

# Работает приоритет. Команды выше сработают первыми.
opts = {"cmds": {
    "greetings": ("привет", "здорово"),
    "menu": ("меню"),
    'bill': ("счет", "счёт"),
    'order': ("заказ", "заказать"),
    'call': ("официант", "валли"),
    "none": (""), # На случай если робот распознает пустую строку
.
    }}

# Проверка наличия вербальных команд в голосовом запросе пользова
теля.
# Если слово найдено - возвращает имя команды, иначе возвращает и
мя команды "not_understand"
def recognizeCommand(text):
    print(text)
    # cmd - слово слева, verbose_cmds - список слов справа в opts
```

```
    for cmd, verbose_cmds in opts['cmds'].items():

        # Это позволяет писать в opts одну вербальную команду
        if (isinstance(verbose_cmds, tuple) == False):
            verbose_cmds = list([verbose_cmds])

        for verbose_cmd in verbose_cmds:
            if(verbose_cmd in text):
                return cmd

    return "none"

def do_command(cmd):
    cmdCtrl.do(cmd)

if __name__ == "__main__":
    while True:
        do_command(recognizeCommand(sr.recognize().lower()))
```

SETTINGS

```
######## Recognition settings ##############################
#IMPORTANT CHECK FIRST YOUR DEVICE INDEX
dev_index = 1       #microphone device_index
recognition_language = 'ru-RU'

######## Synthesis settings ##############################
voicepack = 'ru'
rate = 120           #talking speed
volume = 1

######## Restourant settings ##############################

menu = ['Голубцы куринные', 'Борщ зеленый']
```

COMMANDS

```
from settings import menu

class commandControl:
    def __init__(self, ss, sr):
        self.ss = ss        # Speech Synthiser
        self.sr = sr        # Speech Recognizer
        self.cmds = {
    'greetings': self.sayHi,
    'menu': self.sayMenu,
    'bill': self.sayBill,
    'call': self.sayComing,
    'order': self.getOrder,
    'not_understand': self.notUnderstand,
    'none': self.none
}


    def do(self, cmd):
        ret = ""
        if (cmd in self.cmds):
            print(self.cmds[cmd])
            ret = self.cmds[cmd]()
        if ret:
            return ret



    def sayHi(self):
        self.ss.say("Привет, я робот Валли!")

    def sayMenu(self):
        text = "Мы подаем "
```

```
        for i in range(len(menu)):
            text += menu[i] + ", "
        text += "."
        self.ss.say(text)

    def sayBill(self):
        text="Ваш счет "
        text += "310 рублей."
        self.ss.say(text)

    def getOrder(self):
        listen = True
        text = ""
        while listen:
            text = self.sr.recognize().lower()
            if ("все" in text):
                self.ss.say("На этом все?")
                if ("да" in self.sr.recognize().lower):
                    listen = False

    def sayComing(self):
        self.ss.say("Я еду!")

    def notUnderstand(self):
        self.ss.say("Я не понимаю о чем вы говорите.")

    def none(self):
        self.ss.say("Я не могу понять, что вы сказали.")
```

ПРИЛОЖЕНИЕ В. АДМИНИСТРАТИВНОЕ АНДРОИД ПРИЛОЖЕНИЕ.

MAIN_ACTIVITY

```
package com.example.wally;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

import android.os.Bundle;
import android.view.MenuItem;

import
com.google.android.material.bottomnavigation.BottomNavigationView;

public class MainActivity extends AppCompatActivity {


    private Fragment j = new Joystick();
    private Fragment s = new Stream();
    private Fragment set = new Setting();
    private Fragment m = new VoiceManage();
    private BottomNavigationView bottonnav;

    private String _adress;
    private int _port = -1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        bottonnav = findViewById(R.id.bottomNavigationView);
```

```
bottonnav.setOnNavigationItemSelectedListener(navListner);
          }

          private  BottomNavigationView.OnNavigationItemSelectedListener
navListner =
                   new
BottomNavigationView.OnNavigationItemSelectedListener() {
                       @Override
                       public  boolean  onNavigationItemSelected(@NonNull
MenuItem item) {
                           Fragment selectedFragment = null;

                           switch(item.getItemId()){
                               case R.id.joystick:
                                   selectedFragment = j;
                                   break;
                               case R.id.stream:
                                   selectedFragment = s;
                                   break;
                               case R.id.setting:
                                   selectedFragment = set;
                                   break;
                               case R.id.voice:
                                   selectedFragment = m;
                           }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_p
lace, selectedFragment).commit();
                           return true;
                       }
                   };

          public void portAndAdress(String adress, int port){
```

```
            this._adress = adress;
            this._port = port;
        }

        public void giveCommand(String command){
            if(this._adress != "" && this._port != -1) {
                Client c = new Client(command, this._adress,
this._port);
                c.start();
            }
        }

    }
```

joystick

```
package com.example.wally;

import android.app.Activity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.ToggleButton;
import androidx.fragment.app.Fragment;
import
io.github.controlwear.virtual.joystick.android.JoystickView;

public class Joystick extends Fragment implements
SeekBar.OnSeekBarChangeListener, OnCheckedChangeListener
{
```

```
        private SeekBar rh;
        private SeekBar lh;
        private SeekBar hh;
        private SeekBar eh;

        private TextView rt;
        private TextView lt;
        private TextView ht;
        private TextView et;

        private ToggleButton rht;
        private ToggleButton lht;

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                                 Bundle savedInstanceState) {
            View view = inflater.inflate(R.layout.fragment_joystick,
                    container, false);

            rh = view.findViewById(R.id.righthand);
            lh = view.findViewById(R.id.lefthand);
            hh = view.findViewById(R.id.headrotate);
            eh = view.findViewById(R.id.eye);

            rt = view.findViewById(R.id.rhg);
            lt = view.findViewById(R.id.lhg);
            ht = view.findViewById(R.id.hrg);
            et = view.findViewById(R.id.erg);

            lht = view.findViewById(R.id.lht);
            rht = view.findViewById(R.id.rht);
```

```
            rh.setOnSeekBarChangeListener(this);
            lh.setOnSeekBarChangeListener(this);
            hh.setOnSeekBarChangeListener(this);
            eh.setOnSeekBarChangeListener(this);

            JoystickView joystick = (JoystickView)
view.findViewById(R.id.joystickView);
            joystick.setOnMoveListener(new
JoystickView.OnMoveListener() {
                @Override
                public void onMove(int a, int s) {
                    Activity activity = getActivity();
                    if (activity != null && !activity.isFinishing() &&
activity instanceof MainActivity) {
                        ((MainActivity) activity).giveCommand("J" + "
" + Integer.toString(a) + " " +  Integer.toString(s));
                    }
                }
            });
            return view;
        }

        @Override
        public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) { }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) { }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            String command = "";
            switch (seekBar.getId()){
                case R.id.righthand:
```

```
rt.setText(String.valueOf(seekBar.getProgress()));
                    command         =         "R         "         +
String.valueOf(seekBar.getProgress());
                    break;
                case R.id.lefthand:

lt.setText(String.valueOf(seekBar.getProgress()));
                    command         =         "L         "         +
String.valueOf(seekBar.getProgress());
                    break;
                case R.id.headrotate:

ht.setText(String.valueOf(seekBar.getProgress()));
                    command         =         "H         "         +
String.valueOf(seekBar.getProgress());
                    break;
                case R.id.eye:

et.setText(String.valueOf(seekBar.getProgress()));
                    command         =         "E         "         +
String.valueOf(seekBar.getProgress());
                    break;
            }
            Activity activity = getActivity();
            if (activity != null && !activity.isFinishing() && activity
instanceof MainActivity) {
                ((MainActivity) activity).giveCommand(command);
            }
        }

        @Override
        public   void   onCheckedChanged(CompoundButton   buttonView,
boolean isChecked) {
```

```
                String command = "";
                if (isChecked){
                 switch (buttonView.getId()) {
                    case R.id.lht:
                        command = "T 1";
                        break;
                    case R.id.rht:
                        command = "t 1";
                        break;
                    }
                }
                else{
                    switch (buttonView.getId()) {
                        case R.id.lht:
                            command = "T 1";
                            break;
                        case R.id.rht:
                            command = "t 1";
                            break;
                    }
                }
                Activity activity = getActivity();
                if (activity != null && !activity.isFinishing() && activity
instanceof MainActivity) {
                    ((MainActivity) activity).giveCommand(command);
                }
            }
        }
```

ПРИЛОЖЕНИЕ Г. АНДРОИД ПРИЛОЖЕНИЕ ДЛЯ ЗАКАЗОВ

MainActivity

```
package com.example.reader;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.budiyev.android.codescanner.CodeScanner;
import com.budiyev.android.codescanner.CodeScannerView;
import com.budiyev.android.codescanner.DecodeCallback;
import com.google.zxing.Result;

public class MainActivity extends AppCompatActivity {

    private CodeScanner mCodeScanner;
    private String answer = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        CodeScannerView scannerView =
findViewById(R.id.scanner_view);
        mCodeScanner = new CodeScanner(this, scannerView);
        mCodeScanner.setDecodeCallback(new DecodeCallback() {
            @Override
            public void onDecoded(@NonNull final Result result) {
```

```
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            Toast.makeText(MainActivity.this,
result.getText(), Toast.LENGTH_SHORT).show();
                            answer = result.getText();
                        }
                    });
                }
            });
            scannerView.setOnClickListener(new View.OnClickListener()
{
                @Override
                public void onClick(View view) {
                    mCodeScanner.startPreview();
                    Intent intent = new Intent(MainActivity.this,
ListActivity.class);
                    intent.putExtra("number", answer);
                    startActivity(intent);
                }
            });
        }

        @Override
        protected void onResume() {
            super.onResume();
            mCodeScanner.startPreview();
        }

        @Override
        protected void onPause() {
            mCodeScanner.releaseResources();
            super.onPause();
        }
```

```
}
```

StartActivity

```
package com.example.reader;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.view.View.OnClickListener;

public class StartActivity extends AppCompatActivity implements
OnClickListener {

    private Button start;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_start);

        start = (Button)findViewById(R.id.start);

        start.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if(v.getId() == R.id.start){
            Intent intent = new Intent(StartActivity.this,
MainActivity.class);
```

```
                startActivity(intent);
            }
        }
    }

    ConfirmActivity

    package com.example.reader;

    import androidx.appcompat.app.AppCompatActivity;

    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import android.content.Intent;
    import android.view.View.OnClickListener;

    public class ConfirmActivity extends AppCompatActivity implements
OnClickListener {

        //private Client c;

        private Button confirm;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_confirm);

            confirm = findViewById(R.id.confirm);

            confirm.setOnClickListener(this);
        }
```

```
        @Override
        public void onClick(View v) {
            String data = getIntent().getStringExtra("zakaz");
            String number = getIntent().getStringExtra("number1");
            String[] apn = number.split(" ");
            Client c;
            c = new Client(apn[2] + " " + data, apn[0],
Integer.parseInt(apn[1]));
            c.start();
        }
    }
```

Zakaz

```
package com.example.reader;

public class Zakaz {

    private int count[] = new int[5];

    public Zakaz(){
        for(int i = 0; i < 5; i++) count[i] = 0;

    }

    public void setPosAdd(int pos) {
        count[pos]++;
    }

    public void setPosRem(int pos) {
        count[pos]--;
    }

    public int getPos(int pos){
```

```
        return count[pos];
    }

    public String code(){
        String result = "";
        for(int i = 0; i < 5 ; i++){
            result += Integer.toString(count[i]);
            result += " ";
        }
        return result;
    }
}
```

Client

```
package com.example.reader;
import android.content.res.Resources;
import android.widget.TextView;
import android.widget.Toast;

import java.net.*;
import java.io.*;

import androidx.appcompat.app.AppCompatActivity;

public class Client extends Thread{

    private String _data = "";
    private String _address;
    private int _port;
    private AppCompatActivity _app = null;
    private TextView text;
    private boolean t = false;
```

```
        public Client(String command, String address, int port){
            this._data = command;
            this._address = address;
            this._port = port;

        }

        public void run(){
            try {
                this.main();
                //text.setText("OK");
            } catch (IOException e) {
                t = false;
                e.printStackTrace();
                text.setText("NE OK");
            }
        }
        public void main() throws IOException {
            Socket client_socket = new Socket(this._address,
this._port);

client_socket.getOutputStream().write(this._data.getBytes());
            client_socket.close();
        }

    }
```

ПРИЛОЖЕНИЕ Д. SPRING FRAMEWORK

```
package com.project.controller;

import com.project.domain.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.Authenticatio
nPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.util.Map;
import java.util.UUID;

@Controller
public class MainController {
    @GetMapping("/")
    public String greeting(Map<String, Object> model){
        return "main";
    }

    @GetMapping("/main")
    public String main(
            Model model,
            @AuthenticationPrincipal User user
    ){
        model.addAttribute("url","/main");
        return "main";
```

```
        }
        @GetMapping("/terminal")
        @PreAuthorize("hasAuthority('ADMIN')")
        public String terminal(
                Model model,
                @AuthenticationPrincipal User user
        ){
            model.addAttribute("url","/terminal");
            return "terminal";
        }
       @PostMapping("/main")
       public String add(
               @AuthenticationPrincipal User user,
               @RequestParam(value = "file", required = true) Multipa
rtFile file,
               BindingResult bindingResult,
               Model model
       ) throws IOException {
           saveFile(file);
           model.addAttribute("url","/main");
           return "main";
       }

       private void saveFile(MultipartFile file) throws IOException {
           if (file != null && !file.isEmpty() && file.getSize()!=0 &
& !file.getOriginalFilename().isEmpty() && !file.getOriginalFilename()
.equals("") && !file.getOriginalFilename().equals(" ")) {
               File uploadDir = new File(uploadpath);

               if (!uploadDir.exists()) {
                   uploadDir.mkdir();
               }
               String uuidFile = UUID.randomUUID().toString();
```

```
                String resultFileName = uuidFile + "." +  file.getOrig
inalFilename();

                file.transferTo(new File(uploadpath + "/" + resultFile
Name));
            }
        }



    }
```