Программное обеспечение для моделирования ЭОС можно поделить на три группы в зависимости от интенсивности моделируемого пучка заряженных частиц: для расчета слаботочных электронных пучков (учитывающих только влияние собственного объемного заряда), для расчета сильноточных электронных пучков и для расчета сильноточных электронных пучков в сложных физических условиях.

Для расчета сильноточных пучков существует несколько пакетов прикладных программ, такие как: MAGIC (США), MAFIA (Германия, США), CEM (США), KARAT (Россия, США), POISSON (Новосибирск, США), TAU (Санкт-Петербург), ЭРА (Новосибирск), BEAMCAD (Москва) [2].

Данные ППП используют различные методы дискретизации потока заряженных частиц, MAGIC, MAFIA, CEM, KARAT используют для этого модель больших частиц, POISSON, BEAMCAD и ЭРА, в свою очередь, применяют модель недеформируемых трубок тока, а TAU использует или первый, или второй метод [3].

Все ППП кроме TAU режимы эмиссии с ограничением тока объемным зарядом (ρ-режим) или эмиссионной способностью катода (Т-режим). TAU в свою очередь использует эмиссию с термо-, авто-, фотокатода [4].

Ни один из описанных выше ППП не учитывает в модели вторичной электронно-электронной и ионно-электронной эмиссии.

Однако, в работах, посвященных моделированию электронно-оптических систем всё так же не учитывается компенсирующий заряд для электронных пучков обеспечивающийся извне, моделирование оптических характеристик пучков в системе формирования пучков. Кроме того, в существующем программном обеспечении не учитывается магнитное поле, которое способно влиять на формирование пучков. Что особенно может сказаться в системах с подвижным эмиттером.

Заключение. Исходя из вышеизложенного, можно сделать вывод, что существует необходимость в доработке и дальнейшем развитии программного обеспеченияс его функционалом. Развитие пакетов прикладных программ для моделирования процессов, протекающих в электронно-лучевых установках, поможет в развитии данной технологии и соответственно даст толчок к развитию отраслей промышленности и науки, тесно связанных с ней.

1. Шиллер, З. Электронно-лучевые технологии / З. Шиллер, У. Гайзиг, З. Панцер. – М.: Энергия, 1980. – 528 с.
2. Иванов, В.Я. Новый подход к решению задач взаимодействия заряженных частиц с электромагнитным полем / В.Я. Иванов // Прикладная физика. – 1997. – № 2–3. – С. 128–136.
3. Numerical Simulation of Diodes with Plasma Electrodes / V. Astrelin [et al.] // Proceedings of 15th International Symposium on High Current Electronics. – Tomsk: Publishing house of the IAO SB RAS, 2008. – P.11–15.
4. Tregubov V.F., Tregubov A.V. Program package TAU. Structure and applications for electron device simulation. Proceedings of the 7-th International conference on electron beam technologies. 2003. Varna. Bulgaria. P.39–41.

ВОПРОСЫ УДАЛЕННОГО УПРАВЛЕНИЯ ДЛЯ РОБОТА-МАНИПУЛЯТОРА

Бирюкова Д.В., Шидловский А.В.,

магистранты ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь

Научный руководитель – Маркова Л.В., канд. физ.-мат. наук, доцент

Использование мета-операционной системы Robot Operating System при разработке собственного робота-манипулятора позволяет на основе уже большого количества проверенных решений, быстро и с малыми затратами начать разработку собственной бизнес логики процессов управления. Разработка программного обеспечения для управления движениями робота ведется посредством реализации независимых узлов, каждый из которых предоставляет решение в виде отдельного сервиса. ROS основан на архитектуре графов, поэтому каждый узел может обмениваться информацией с другим узлом используя встроенные механизмы транспорта и пользовательские типы сообщений [1].

Управление изнутри системы является результатом работы отдельного сервиса, который публикует или запрашивает команды перемещения от сервиса исполнения. Вне действия мастера ROS, который выполняет роль связки между узлами, воздействие на систему затруднено, что может потребоваться при управлении сразу парой таких манипуляторов.

Репозиторий ВГУ

Поэтому требуется рассмотреть способы удаленного управления для системы такого робота-манипулятора.

Целью исследования является анализ и выбор программных решений для удаленного взаимодействия с интерфейсами робота-манипулятора.

Материал и методы. Материалом для исследования послужили модели роботов-манипуляторов под управлением мета-операционной системы ROS. Программные интерфейсы для общения между сервисами реализуемые на базе Robot Operating System. Протоколы обмена между независимыми программными сервисами. Использовались методы компьютерного программирования.

Результаты и их обсуждение. Для управления движением робота-манипулятора, а также воздействия на концевой эффектор, требуется сформировать программу для контроллера робота. Формат входной программы строго описан протоколом и является пользовательским типом сообщения в рамках ROS. Следуя из этого можно сделать вывод, что для удаленного управления системой требуется разработать дополнительный сервис, который будет являться посредником в процессе приема данных из вне и их конвертацией в требуемый формат исполнителя.

На сегодняшний день существует большое количество протоколов, позволяющих обеспечить взаимодействие между сервисами приложений в рамках различных архитектур. К наиболее популярным относят протоколы Message Queuing Telemetry Transport (MQTT), Apache Kafka, фреймворк для удаленного вызова процедур gRPC [2].

Протокол MQTT является легковесным открытым протоколом обмена данными на удаленных узлах, где требуется небольшой размер кода и есть ограничения по пропускной способности канала. Обмен сообщениями осуществляется между клиентом (client) и брокером (broker) сообщений. Клиент может выступать в роли издателя (publisher) или подписчика (subscriber). Издатель отправляет данные на MQTT брокер, указывая в сообщении определенную тему, топик (topic). Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики.

Apache Kafka использует схожий принцип обмена сообщениями MQTT, но на собственном сетевом протоколе. Особенностью является хранение огромных объемов данных на диске и допуске потребления в режиме реального времени или позже (до тех пор, пока данные все еще доступны на диске). На уровне топика доступны настройки конфигурационных параметров, которые позволяют определить объем хранимых данных, факторы избыточности и т.п.

Данные протоколы хоть и являются популярными, но они больше нацелены на постоянный обмен между сервисами, что допустимо при реализации удаленного управления роботом - манипулятором, но являются довольно избыточными, так как планируется запуск определенного маршрута, по типу запрос ответ. Так же протоколы использует JSON для обмена между клиентами, что требует определенной договоренности, чтобы правильно интерпретировать сообщения поступаемые из топиков.

Используя фреймворк для удаленного вызова процедур gRPC возможно однозначно определить обмен сообщениями между клиентом и сервисом посредством использования языка определения интерфейса (IDL). Для gRPC в основном языком определения интерфейса является буфер протокола Protobuf. Он позволяет хранить свои данные и функциональные контракты в виде прото-файла. Поскольку это форма контракта, и клиент, и сервер должны иметь один и тот же прото-файл. Файл proto действует как посреднический контракт для клиента для вызова любых доступных функций с сервера. Фреймворк gRPC позволяет определить 4 метода обслуживания, одним из которых является унарный RPC, где клиент отправляет один запрос на сервер и получает один ответ обратно, как при обычном вызове функции [3]. Данная концепция лучше всего позволяет удаленно запускать робота, при этом формирование программы возможно на уровне как и клиента, так и сервера, в зависимости от описания файла proto.

Для использования gRPC в проекте ROS был написан сервер на языке C++, который по запросу, описанному в соответствии с proto сервера, позволит от любого клиента запустить выполнение пользовательской программы на роботе-манипуляторе.

Заключение. При использовании технологий MQTT, Apache Kafka, фреймворка для удаленного вызова процедур gRPC, возможно удаленное управление сервисами ROS. В случае за-

Репозиторий БГУ

пуска робота по траектории преимущественно использовать архитектуру запрос - ответ, которую предоставляет унарный RPC фреймворка gRPC.

1. Использование средств визуальной и физической симуляции при разработке робота-манипулятора/ Д.В. Бирюкова, А.В Шидловский // Молодость. Интеллект. Инициатива : материалы VIII Междунар. науч.-практ. конф. студентов и магистрантов, Витебск, 22 апреля 2020 г. – Витебск : ВГУ имени П. М. Машерова, 2020. – С. 5-7.
2. Понимание брокеров сообщений [Электронный ресурс] Режим доступа: https://habr.com/ru/post/471268/
3. gRPC в качестве протокола межсервисного взаимодействия [Электронный ресурс] Режим доступа: https://habr.com/ru/company/yandex/blog/484068/

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ

Валевич И.А.,

студент 4 курса ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь

Научный руководитель – Сергеенко С.В., ст. преп.

На данный момент существует проблема массовой оценки знаний студентов способом тестирования при аудиторных занятиях с использованием бумажных бланков, ведь данный процесс крайне затратен по времени и требует постоянной концентрации внимания преподавателей при личной проверке, проверка же самими студентами вызывает также проблему возможной фальсификации результатов тестирования.

Цель работы – автоматизировать процесс проверки тестирования с использованием бумажных бланков и позволить выполнять его удаленно путем анализа изображения бланка ответов соответствующей информационной системой.

Материал и методы. Материалом исследования выступает машинное зрение как система для автоматизации тестирования с использованием бумажных бланков. При создании системы автоматизации тестирования использована клиент-ссерная архитектура приложения. Для разработки серверной части используется платформа Firebase [3].

Для создания бланков тестирования используется нативное iOS приложение. Приложение имеет два основных модуля, модуль для работы с бланками и модуль проверки результатов тестирования.

После создания списка вопросов преподаватель может импортировать бланк в PDF документ.

Во время аудиторного занятия студенты заполняют бланк с вопросами. Далее для автоматизации проверки используется модуль распознавания изображения на базе библиотеки Vision [2]. Данный модуль переводит изображение с камеры мобильного устройства в модель теста и автоматически выставляет оценку работе.

Само iOS приложение написано на языке Swift 5 с использованием библиотеки для построения пользовательского интерфейса SwiftUI [1]. В качестве среды разработки используется Xcode 13.

Приложение позволяет хранить подготовленные тесты в облачном хранилище Firebase Cloud Firestore. Уже готовые тестовые бланки хранятся в Firebase Storage, что упрощает работу с документами и позволяет получить доступ по ссылке через Gmail аккаунт.

Результаты и их обсуждение. Разрабатываемая информационная система позволяет осуществлять автоматизировать процесс тестирования. Приложение реализует разделение пользователей на две роли: администратор и преподаватель.

Так же администратор имеет доступ к платформе Firebase, что позволяет ему получать доступ к работе с всеми аккаунтами приложения.

Модуль работы с бланками тестирования позволяет создать бланк с вопросами и экспортировать его в PDF документ. После преподаватель может раздать распечатанные бланки для выполнения тестов во время аудиторного занятия, лично контролируя процесс выполнения задания для избежание возможной фальсификации результатов тестирования.

Преподавателю доступны функции создания, редактирования и просмотра тестов по разным темам. Для получения результата теста необходимо навести камеру на бланк и просканировать изображение. Форма бланка позволяет успешно преобразовывать изображение с камеры в модель данных при помощи машинного зрения. Модуль распознавания использует нейрон-

Репозиторий ВГУ