

Санкт-Петербургский государственный университет

Кафедра Системного программирования

Группа 21.Б10-мм

Разработка механизма обмена сообщениями для коммуникации роботов ТРИК

Миняев Роман Русланович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
старший преподаватель кафедры системного программирования,
Кириленко Яков Александрович

Санкт-Петербург
2024

Оглавление

Введение	3
1. Обзор	4
1.1. Поддержка сетевого взаимодействия роботов	4
1.2. Архитектура и иерархия классов	5
1.3. Устройство сети	6
1.4. Недостатки текущей реализации	7
2. Постановка задачи	9
3. Реализация	10
3.1. Поддержка нового блока	10
3.2. Улучшение сетевого взаимодействия	10
3.3. Остальное	12
Заключение	14
Список литературы	15

Введение

TRIK [2] — набор инструментов для разработки и программирования робототехнических проектов. TRIK включает в себя конструкторы, программное обеспечение и различные учебные материалы, которые позволяют пользователям создавать и программировать роботов, а также изучать основы механики, электроники и программирования.

TRIK Studio [3] — среда программирования робототехнических моделей, собранных на основе популярных контроллеров (TRIK, LEGO EV3, LEGO Mindsorms NXT 2.0 и других). Она предоставляет пользователям удобный интерфейс для создания программ с использованием визуального программирования, а также текстового кода на языках Python и JavaScript, которые можно отлаживать во встроенном симуляторе движения робота.

В настоящее время, для образовательных целей TRIK важной задачей является внедрение в продукты, включая демонстрационные и учебные материалы, заданий и примеров, непосредственно затрагивающих коммуникацию роботов.

Имеющаяся реализация базовых возможностей взаимодействия роботов не удовлетворяет техническим требованиям, предъявляемым к перспективному продукту. Список замечаний от пользователей включает жалобы на стабильность, удобство настройки и производительность, а также пожелания к расширению возможностей. Новая версия механизма взаимодействия должна позволять реальному роботу обмениваться сообщениями с виртуальным роботом в режиме имитационного моделирования.

1. Обзор

В данном разделе будут рассмотрена текущая реализация сетевого взаимодействия роботов, а также проблемы, обнаруженные при изучении исходного кода и архитектуры приложения.

1.1. Поддержка сетевого взаимодействия роботов

В TRIK Studio взаимодействие между роботами происходит посредством механизма почтовых ящиков, реализованного в модуле `trikNetwork` репозитория `trikRuntime`, а также поддержкой кодогенерации для определенных блоков, описанных ниже.

Для поддержки сетевого взаимодействия роботов, необходимо явно разрешить использование в настройках TRIK Studio. После этого приложение сможет открывать порты для взаимодействия экземпляров по протоколу TCP, а также сделает доступными блоки `TrikSendMessage` и `TrikWaitForMessage` визуального языка программирования.

Каждый экземпляр TRIK Studio/робот должен иметь определенный идентификатор в сети, который называется бортовым номером. При помощи бортового номера пользователь получает возможность в некоторой степени абстрагироваться от остальных параметров сети.

Кодогенерация	Описание	Имя блока
send	Отправить сообщение роботу с определенным бортовым номером/всем роботам	TrikSendMessage
receive	Получить сообщение блокирующим/неблокирующим способом	TrikWaitForMessage
myHullNumber	Получить бортовой номер	—
connect	Подключение к роботу по заданному IP-адресу и порту	—

Пример простейшей программы для получения сообщения на визуальном языке показан на рис. 1.

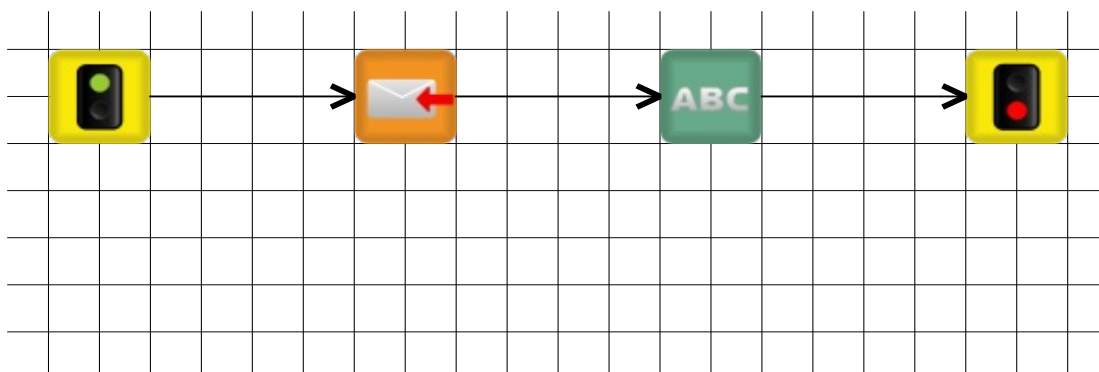


Рис. 1: Пример программы для получения сообщения

1.2. Архитектура и иерархия классов

Упомянутый выше `trikNetwork` модуль реализован при помощи модуля `QtNetwork`, предоставляющего возможности для запуска сервера, записи/чтения в TCP-сокеты и других возможностей, необходимых для написания приложения, использующего стек TCP/IP.

Реализацию сервера представляет класс `TrikServer`, ответственный за его работоспособность, обработку и управление входящих соединений. Класс `MailboxServer` расширяет его функциональность, предоставляя возможности по установке TCP-соединения с сервером в качестве клиента, управлению исходящими соединениями, а также содержит таблицу доступных роботов и сетевые параметры. Класс `Connection` инкапсулирует в себе TCP-сокеты, реализуя обработчики для корректной инициализации, освобождения соединений, keep-alive обработку соединения. В свою очередь, `MailboxConnection` реализует обработчики для различных команд, отправляемых на сервер. Например, команды «register» для регистрации робота в сети или команды «connection» для обновления информации о соединении. Класс `Mailbox` управляет поведением `MailboxServer`, выделяя под него отдельный поток и является реализацией `MailboxInterface`, представляющего из себя интерфейс объекта «mailbox», описанного в документации TRIK Studio. Диаграмма

классов модуля trikNetwork показана на рис. 2.

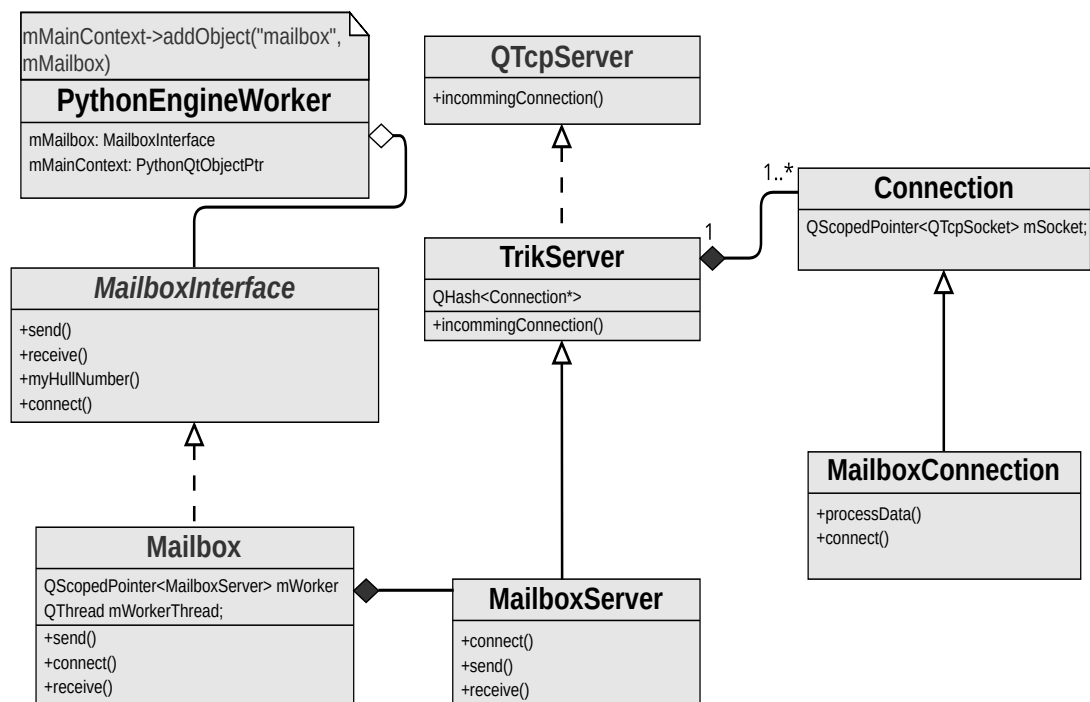


Рис. 2: Диаграмма классов trikNetwork

1.3. Устройство сети

В данной реализации сеть является одноранговой, то есть в ней все узлы общаются между собой без центрального элемента. Также, в сети не предусмотрена маршрутизация, то есть для того, чтобы один робот мог отправить другому пакет, необходимо установить между ними прямое соединение, что делает сеть достаточно устойчивой к падениям. Однако, это может стать проблемой при использовании большого количества роботов в сети, так как ее сложность растет квадратично, а каждое соединение требует отдельный поток, но учитывая опыт практического использования модуля, это не является значимым ограничением.

При подключении нового робота в сеть, робот, к которому подключаются (это может любой элемент сети), отправляет список всех известных ему роботов с их IP-адресами и портами, которые регистрируются в таблице робота, инициировавшего соединение. Однако, сокет при будет создан в нужный момент, то есть при прямом запросе или отправке сообщения.

1.4. Недостатки текущей реализации

В текущей реализации параметры сети для настройки MailboxServer лежат в глобальном конфигурационном файле. Иначе говоря, параметры каждого элемента сети являются глобальными свойствами, а не свойствами проекта, что приводит к следующим проблемам:

1. Воспроизведение существующих примеров становится тяжелым, так как каждая демонстрация примера подразумевает под собой ручное изменение параметров через пользовательский интерфейс TRIK Studio. Ситуация ухудшается тем, что значение бортового номера не синхронизировано со значением в конфигурационном файле (который, на самом деле, не имеет отношения к TRIK Studio).
2. При настройке нескольких элементов сети на одном устройстве, они будут конкурировать за этот файл и поведение скриптов перестанет быть ожидаемым.

Также, в текущей реализации сеть становится неработоспособной из-за некоторых проблем с использованием TCP-соединений:

1. Несмотря на архитектурное решение использовать одноранговые сети, между каждым элементом сети поднимается два соединения клиент-сервер вместо использования одного двунаправленного.
2. При использовании нескольких подряд идущих операций, требующих валидного соединения, (например, `connect -> setHullNumber`

-> send -> send), соединения и TCP-сокеты, которые они в себе инкапсулируют, инициализируются по несколько раз, что приводит к неработоспособности всех соединений.

3. Из-за неаккуратного использования механизма сигналов/слотов Qt, запуск каждого потока для нового соединения сопровождается многократной переинициализацией сокета для других соединений.

К остальным проблемам можно отнести.

1. Не все компоненты объекта «mailbox» поддерживают кодогенерацию и/или доступны из скриптов. Например, после генерации Python или JavaScript кода для блоков send/receive может потребоваться вручную добавлять метод connect.
2. Отсутствие содержательных примеров и более подробной документации, что иногда приводит к некорректному использованию в существующих примерах.
3. Полное отсутствие тестов, что мешает оценить регресс при обновлении системы.
4. Зарегистрированы падения (crash) TRIK Studio при использовании mailbox.
5. При настройке нескольких элементов сети с одного устройства трудно отследить какому экземпляру принадлежит определенный порт, что мешает эффективно писать скрипты.

2. Постановка задачи

Целью работы является стабилизация работы модуля trikNetwork и добавление новой функциональности для настройки взаимодействия роботов в сети.

Для её выполнения были поставлены следующие задачи.

1. Провести обзор модуля trikNetwork для выявления направлений оптимизаций, рефакторинга.
2. Провести рефакторинг модуля trikNetwork и добавить новую функциональность в TRIK Studio для решения перечисленных выше проблем.
3. Написать тесты на модуль trikNetwork, внедрив их в CI репозитория trikRuntime.

3. Реализация

В данном разделе будет описана реализация и шаги, предпринятые для достижения поставленных ранее целей.

3.1. Поддержка нового блока

Как было отмечено выше, параметры сети хранятся в глобальных в конфигурационных файлах, а не являются свойствами проекта, что приводит к множеству ошибок. Возможным решением данной проблемы является добавление блока, который содержал бы в себе все необходимые параметры для настройки сети с последующей поддержкой кодогенерации для Python или JavaScript, по следующим причинам:

- Воспроизводимость примеров, то есть возможность исполнять программу без необходимости ручной настройки параметров.
- Свойства блока описаны в проекте, что означает отсутствие конфликтов между несколькими экземплярами TRIK Studio на одном хосте при настройке сети.
- Единый интерфейс для настройки параметров сети.
- Дополнительные возможности по смене бортовых номеров робота во время исполнения скрипта.
- Решение проблемы с кодогенерацией для mailbox.connect.

В связи с перечисленными выше преимуществами данного подхода, посредством уже существующих технологий по генерации кода, было решено реализовать TrikJoinNetwork блок.

Пример использования блока TrikJoinNetwork на рис. 3.

3.2. Улучшение сетевого взаимодействия

В первую очередь, была снижена сложность сети, вызванная созданием двух ТСР-соединений для каждой пары хостов путем изменения

Параметр блока	Описание
IP-адрес	IP-адрес при необходимости подключения, иначе пустая строка
Порт	Конкретный порт для подключения, -1 для подключения к стандартному порту
Бортовой номер	Замена бортового номера робота, -1 при отсутствии необходимости замены

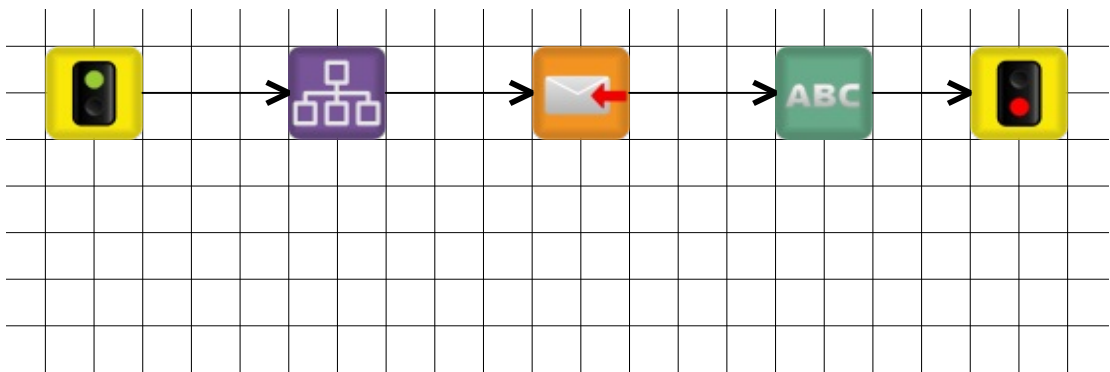


Рис. 3: Использование TrikJoinNetwork блока

способа заполнения таблиц на работе. Вместо переиспользования существующих соединений, производилась попытка подключения к роботу по порту, который он указал как доступный для подключения, в то время как между хостами уже существовало валидное соединение в обратную сторону (в терминологии клиент-сервер).

Несмотря на то что сам MailboxServer однопоточный (не в том смысле, что у него все соединения лежат в одном QThread, а в том, что он создает потоки последовательно и помещает туда соединения), при создании нового соединения, переинициализировались другие соединения из-за использования сервером единого сигнала. При старте потока, содержащего соединение, производился Q_EMIT сигнала сервера, на который подписан обработчик каждого соединения. То есть для каждого соединения в очередь обработки ставилось несколько идентичных задач. Это могло решаться обычным Qt::UniqueConnection для пары

MailboxServer, MailboxConnection, однако существовала проблема создания нескольких идентичных соединений с теми же параметрами подключения во время последовательного исполнения операций, требующих валидного соединения. Соединение не успевало стать валидным, в то время как следующая операция создавала новый объект на основании «невалидности» предыдущего. Для решения этой проблемы был изменен порядок инициализации соединения.

В ходе учебной практики была восстановлена функциональность по отправке сообщений всем роботам с одинаковым бортовым номером и исправлена ошибка с падением TRIK Studio из-за преждевременного освобождения потоков соединений.

На модуль trikNetwork были написаны тесты с помощью фреймворка Google Test [1], покрывающие основные пользовательские сценарии использования.

Были написаны учебные примеры, показывающие взаимодействие хостов сети, циклическую отправку сообщений между 2-3 роботами, массовую рассылку сообщений, мультикаст, изменение бортовых номеров во время исполнения и т.д.

Реализацию можно найти в master ветках репозитория [trik-studio](#) и [trikRuntime](#).

3.3. Остальное

В данном разделе перечислены некоторые задачи, которые были сделаны в ходе учебной практики, но не относятся к основной теме.

3.3.1. PythonQt

Актуальной задачей была миграция с Qt5.12 на Qt5.15 [3] с последующей поддержкой Python3.12 в скриптах, что осложнялось архитектурными недостатками PythonQt, а именно отсутствием возможности корректной очистки ресурсов после исполнения скриптов.

Вклад и состояние проектов можно увидеть в master ветках репозитория [trik-pythonqt](#) и [pythonqt](#).

3.3.2. TRIK Checkapp

На олимпиадах/соревнованиях по робототехнике или для проверки домашних заданий необходимо уметь проверять, что программа, написанная учеником, работает корректно и соответствует определенным ограничениям (время работы не превосходит заданного порога, робот посетил нужные участки поля, робот остановился в нужном месте и другое). Был проведен рефакторинг утилиты TRIK Checkapp для проверки соответствия решения определенным ограничениям.

Вклад и состояние проекта можно увидеть в master ветке репозитория [trik-checkapp](#).

3.3.3. CI

Репозиторий TRIK Studio был полностью переведен с Microsoft Azure на GHA, были добавлены новые пайплайны для Altlinux, macOS-12. В CI TRIK Checkapp был внедрен самописный Github Action для clazy, написаны конфигурации Clang-Format и Uncrustify для trikRuntime.

Реализацию github action для clazy можно найти в master ветке репозитория [clazy-standalone-action](#).

3.3.4. Мелочи

В ходе учебной практики было решено множество небольших, но достаточно актуальных проблем: починка инсталлятора, баги с подпрограммами на macOS, постоянные проблемы, возникающие из-за несовершенства текущего способа упаковки приложения (например, отсутствие .dll для стороннего модуля с несколькими уровнями вложенности, что потребовало разобраться в WinDbg).

Заключение

В результате проделанной работы были выполнены следующие задачи.

- Проведён обзор модуля `trikNetwork` для выявления направлений оптимизаций, рефакторинга.
- Проведен рефакторинг модуля `trikNetwork` и добавлена новую функциональность в виде блока `TrikJoinNetwork`.
- Написаны тесты на модуль `trikNetwork`.
- Решено несколько полезных подзадач не относящихся к теме учебной практики, но улучшающих работу `TRIK Studio`.

Список репозиторий:

<https://github.com/trikset/trik-studio>

<https://github.com/trikset/trikRuntime>

<https://github.com/trikset/trik-checkapp>

<https://github.com/trikset/trik-pythonqt>

<https://github.com/MeVisLab/pythonqt>

<https://github.com/MinyazevR/clazy-standalone-action>

Разработка ведется под аккаунтом [MinyazevR](#)

Список литературы

- [1] Google. Google Test. — URL: <https://github.com/google/googletest> (online; accessed: 2024-10-07).
- [2] КиберТех. TRIK. — URL: <https://trikset.com/> (online; accessed: 2024-10-07).
- [3] КиберТех. TRIKStudio. — URL: <https://github.com/trikset/trik-studio> (online; accessed: 2024-10-07).