Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Малютин Данила Павлович

Выпускная квалификационная работа

Поддержка отладки текстовых языков в TRIK Studio

Научный руководитель:
к. т. н., доц. Литвинов Ю. В.

Консультант:
ст. преп. Кириленко Я. А.

Рецензент:
программист Мордвинов Д. А.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Information Systems Administration and Mathematical Support
Software Engineering

Malyutin Danila

Support for debugging textual programming languages in TRIK Studio

Graduation Project

Scientific supervisor:
associate professor Yurii Litvinov

Consultant:
senior lecturer Iakov Kirilenko

Reviewer:
software engineer Dmitry Mordvinov

Saint-Petersburg
2017

Оглавление

Введение

Много лет робототехника успешно применяется в школьном образовании. Использование робототехнических конструкторов позволяет не только обучить детей навыкам программирования, но также углубить их познания в математике, физике и других науках. Большую роль в процессе обучения играет не только сам конструктор, но и среда, в которой его предполагается программировать. Одной из таких сред разработки является TRIK Studio.

TRIK Studio[1] — среда обучения основам программирования и кибернетики [9]. Среда позволяет создавать графические программы для роботов Lego Mindstorms NXT 2.0, Lego EV3 [8], TRIK и исполнять эти программы на компьютере, посылая команды роботу через Bluetooth или USB-интерфейс, а также генерировать по диаграммам код на различных языках программирования и закачивать его для исполнения в робота.

Одной из важных особенностей среды TRIK Studio является наличие 2D-модели, позволяющей симулировать поведение робота при исполнении программы в специальном виртуальном окружении, перед тем как проверять работу программы на реальном роботе. Это позволяет производить отладку программ даже без наличия доступа к роботу, что может быть важно, когда количество обучающихся заметно превышает количество роботов. Данная возможность позволяет обучаться программированию вообще без использования робота.

TRIK Studio создана с помощью системы метамоделирования QReal[2], разрабатываемой в рамках научно-исследовательской группы исследования визуальных модельно-ориентированных технологий разработки ПО на кафедре системного программирования Санкт-Петербургского Государственного Университета. Следует отметить, что функционал TRIK Studio, ориентированный на текстовые языки программирования, весьма ограничен. Изначально, данный функционал заключался

[1]Всё о ТРИК: TRIK Studio, URL: http://blog.trikset.com/p/trik-studio.html (дата обращения: 02.02.2017)

[2]CASE and metaCASE system, URL: https://github.com/qreal/qreal (дата обращения: 08.09.2016)

лишь в возможности просмотра и внесения правок в сгенерированный код на специальной вкладке с подсветкой синтаксиса. В частности, интерпретация программ в рамках 2D-модели была возможна лишь для кода, созданного на языке диаграмм. Эти ограничения сильно затрудняют использование TRIK Studio в качестве инструмента для обучения детей и школьников текстовому программированию.

Другими словами, среда TRIK Studio предоставляет адекватный функционал для написания текстовых программ для роботов, но в ней сильно не хватает возможностей для их отладки. Особенно остро необходимость во встроенной отладке ощущается, если нет возможности запустить программу на самом роботе.

1. Постановка задачи

Целью данной работы было создать решение для поддержки отладки текстовых языков программирования роботов в среде TRIK Studio.

Так как самым популярным языком для текстового программирования в TRIK Studio является язык JavaScript, поддерживаемый роботами TRIK [5], было решено начать с него. В связи с этим были поставлены следующие задачи.

- Разработать архитектуру решения.
- Реализовать решение:
  - интерпретатор кода на языке JavaScript в TRIK Studio;
  - отладчик для JavaScript.
- Провести апробацию созданного решения.

2. Обзор

2.1. Обзор существующих средств отладки текстовых языков

2.1.1. GNU Debugger

GNU Debugger — это переносимый отладчик, поддерживающий отладку многих языков программирования, в том числе C, C++, Java, Free Pascal, Ada, Fortran и др[3]. Отладчик GDB выделяет следующие свои четыре возможности, как основные [1]:

- возможность запустить программу, указав что-либо, что может повлиять на её поведение;
- возможность остановить исполнение программы по достижении какого-либо условия;
- возможность исследовать то, что произошло, когда программа остановилась;
- возможность изменить программу в целях исправления найденной ошибки.

Работа с GDB происходит через текстовый интерфейс командной строки. В таблице ниже приведено несколько примеров команд, поддерживаемых отладчиком GDB.

Команда	Значение
gdb program	произвести отладку программы program
break main	установить точку останова на процедуру main
run -v	запустить загруженную программу с параметром -v
info registers	показать содержимое всех регистров

Это далеко не полный список команд отладчика GDB [3], но из них всех можно выделить несколько основных групп команд, отвечающих за отладку:

[3]GDB: The GNU Project Debugger, URL: https://www.gnu.org/software/gdb/, (дата обращения: 15.05.2017)

- команды, отвечающие за установку разного рода точек останова;
- команды, позволяющие выполнять программу по шагам;
- команды, позволяющие просматривать текущее состояние стека программы, её локальные и глобальные переменные;
- команды, позволяющие просмотреть исполняемый в данный момент код, а также вывести его ассемблерный листинг.

Из этого можно сделать вывод, что наиболее важным функционалом отладчика является поддержка точек останова, так как другие возможности мало полезны в отрыве от точек останова, либо могут быть реализованы черех них.

2.1.2. Node.js Debugger

Node.js Debugger[4] — это отладчик, встроенный в Node.js, кроссплатформенную среду выполнения языка JavaScript на стороне сервера. Node.js Debugger не является полноценным отладчиком кода, но реализует минимальный функционал, необходимый для отладки программ. В его возможности входит установка точек останова на различные события, такие как исполнение строки с заданным номером или вызов определённой функции, с последующим исполнением программы по шагам. Стоит отметить, что установка точек останова или отслеживание состояния переменных происходит путём модификации исходного кода программы добавлением управляющих конструкций необходимых для отладки, таких как "debugger;", "setBreakpoint(line)" и т.п.

В новых версиях среды Node.js происходит отказ от этого отладчика в пользу Chrome DevTools[5].

[4]Debugger, URL: https://nodejs.org/api/debugger.html, (дата обращения: 16.05.2017)

[5]V8 Inspector Integration for Node.js, URL: `https://nodejs.org/api/debugger.html#debugger_v8_inspector_integration_for_node_js`, (дата обращения: 16.05.2017)

2.1.3. Chrome DevTools

Chrome DevTools[6] — это набор для написания и отладки веб-приложений, входящий в состав браузера Google Chrome. Помимо многих удобных инструментов для профилирования и отладки веб-сайтов, он также предоставляет удобный графический интерфейс для отладки программ на языке JavaScript (рис. 1).

В отрыве от специфичных для веб-приложений функций, отладка JavaScript с помощью набора Chrome DevTools сводится к использованию тех же инструментов, что и у отладчика GDB: использование точек останова, исполнение кода по шагам, просмотр и отслеживание значений переменных. Только в отличие от интерфейса командной строки, предоставляемого отладчиком GDB, в наборе Chrome DevTools отладка происходит прямо в браузере с использованием графического пользовательско интерфейса, что заметно повышает удобство отладки.

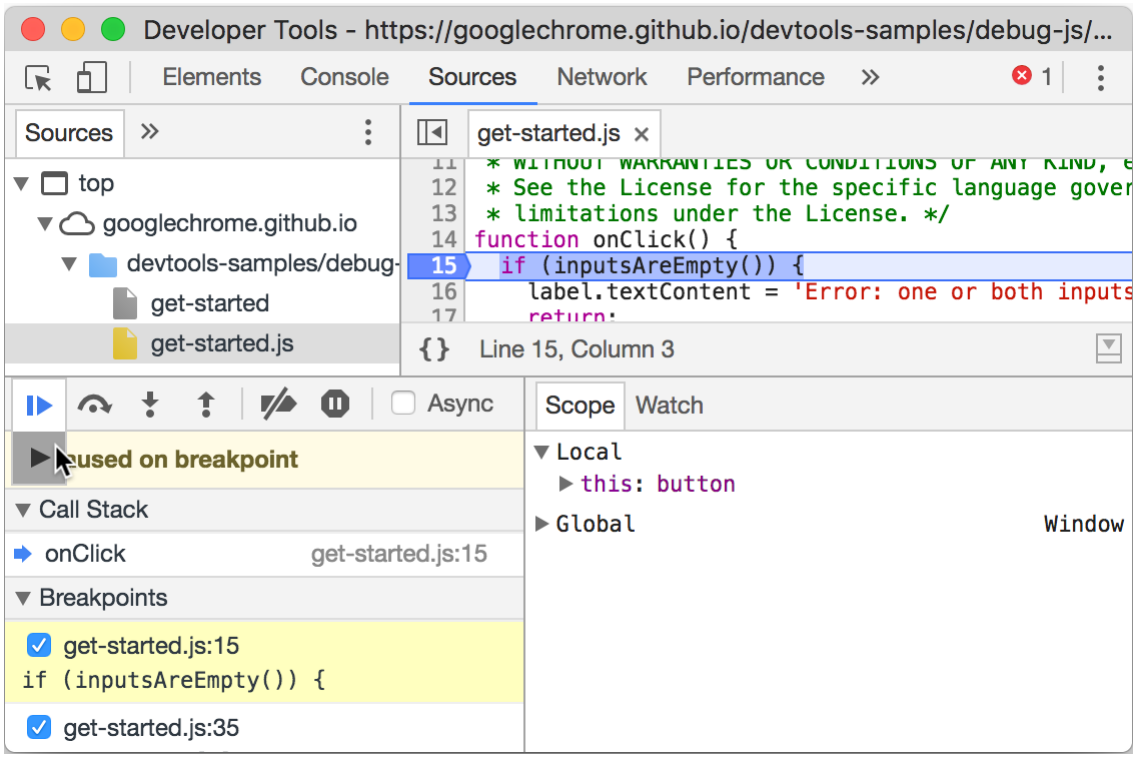

Рис. 1: Отладка программы на языке JavaScript в Chrome DevTools (изображение взято с ресурса `https://developers.google.com/`)

[6]Chrome DevTools | Web | Google Developers, URL: https://developers.google.com/web/tools/chrome-devtools/, (дата обращения: 16.05.2017)

2.2. Описание используемых технологий

2.2.1. TRIK Studio

Среда TRIK Studio основана на проекте QReal [7], metaCASE системе, позволяющей пользователям создавать свои собственные визуальные языки и редакторы к ним. Основная часть TRIK Studio реализована как плагин к QReal. Среда TRIK Studio поддерживает такие робототехнические конструкторы, как Lego Mindstorms NXT 2.0, Lego EV3, TRIK, и позволяет создавать для них программы на интуитивно-понятном языке диаграмм (рис. 2).

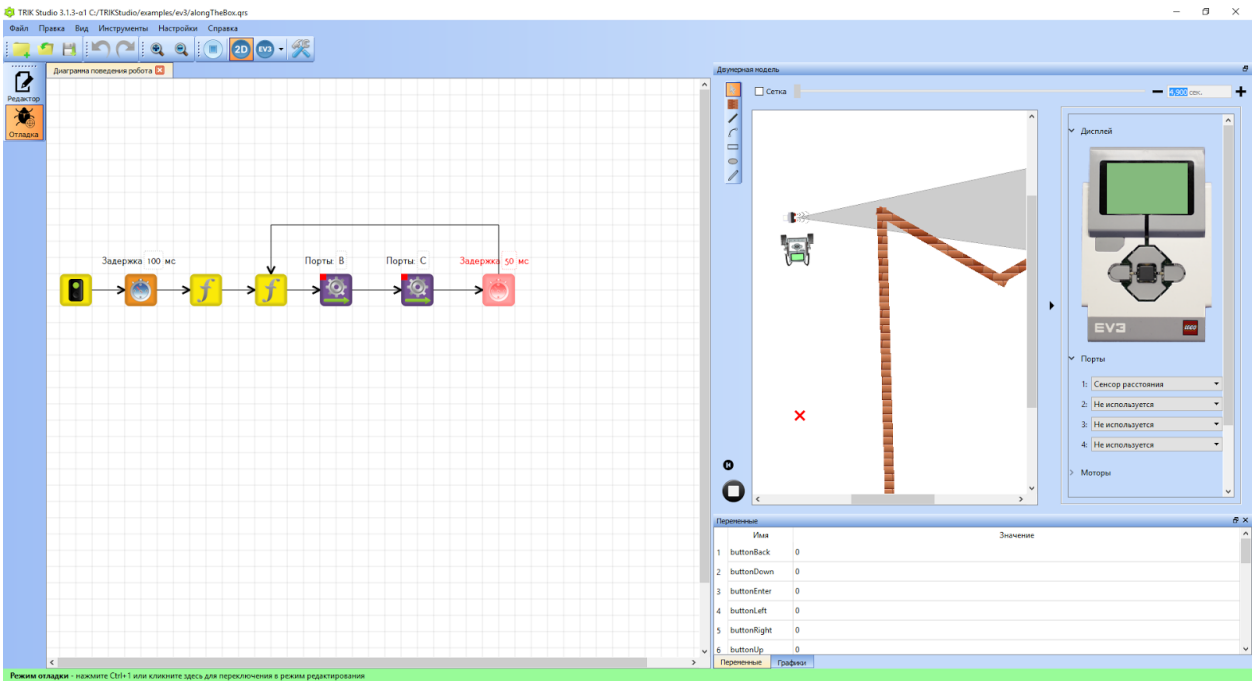

Рис. 2: Пример программы в TRIK Studio (изображение взято с ресурса `http://robots.qreal.ru/`)

Отличительной особенностью среды TRIK Studio является наличие двухмерной модели, позволяющей запустить написанную программу в самой среде, без необходимости в использовании реального робота. Вдобавок к этому, существует специальный язык, основанный на xml, позволяющий, описывая различные условия и ограничения, создавать задачи по программированию, которые можно автоматически проверять на 2D-модели [10].

Архитектура среда TRIK Studio обладает высокой степенью модульности. В частности, поддержка любого робототехнического конструктора осуществляется набором различных, слабо зависящих друг от друга, плагинов, отвечающих за интерпретацию, абстракцию модели робота, генерацию для поддерживаемых роботом языков программирования и так далее [2].

Среда TRIK Studio является проектом с открытым исходным кодом, написанным с использованием кроссплатформенной библиотеки Qt и работает под всеми популярным операционными системами для настольных ПК.

2.2.2. TRIK Runtime

Библиотека TRIK Runtime[7] является написанной на Qt средой выполнения для контроллеров TRIK. Она разделяется на несколько частей (упрощённая диаграмма компонентов представлена на рис. 3), обеспечивающих различную функциональность робота.

trikCommunicator Компонента, ответственная за общение по сети с внешними клиентами (например, QReal). Реализует протокол приёма и исполнения программ на языке JavaScript (далее — скриптов).

trikControl Логическая модель робота, предоставляющая высокоуровневые команды роботу, так, чтобы они были доступны из скриптов.

trikGui Пользовательский интерфейс, содержащий в себе коммуникатор и позволяющий просматривать файлы, IP-адрес, запускать скрипты.

trikHal Уровень абстракции над аппаратным обеспечением робота.

[7]Runtime for TRIK controller, URL: https://github.com/trikset/trikRuntime, (дата обращения: 10.11.2016)

trikKernel Набор утилит, используемых по всему TRIK Runtime. Все остальные компоненты от него зависят.

trikNetwork Код для поддержки сетевой функциональности, не зависящий от аппаратной платформы робота.

trikRun Консольное приложение для запуска скриптов.

trikScriptRunner Интерпретатор QtScript, расширения языка JavaScript с поддержкой дополнительных конструкций из библиотеки Qt, таких как, например, сигналы.

trikServer Простой сервер, прослушивающий заданный порт на предмет полученных команд для робота.

trikTelemetry Сервер телеметрии.

trikWiFi Подсистема управления WiFi-подключением.

В интерпретации программ на языке JavaScript задействованы trikHal, trikKernel, trikControl, trikNetwork и trikScriptRunner.

Несмотря на то, что среда выполнения TRIK Runtime была написана для роботов TRIK, её можно скомпилировать и запустить и на персональных компьютерах под управлением большинства популярных операционных систем (рис. 4). В данном случае в абстракции аппаратного обеспечения используются классы-пустышки. Тем не менее, вся основная функциональность по настройке окружения для интерпретации программ на языке JavaScript работает так же, как и на реальном роботе, что позволило переиспользовать часть компонент среды выполнения в данной работе, подключив TRIK Runtime как внешнюю библиотеку к TRIK Studio.

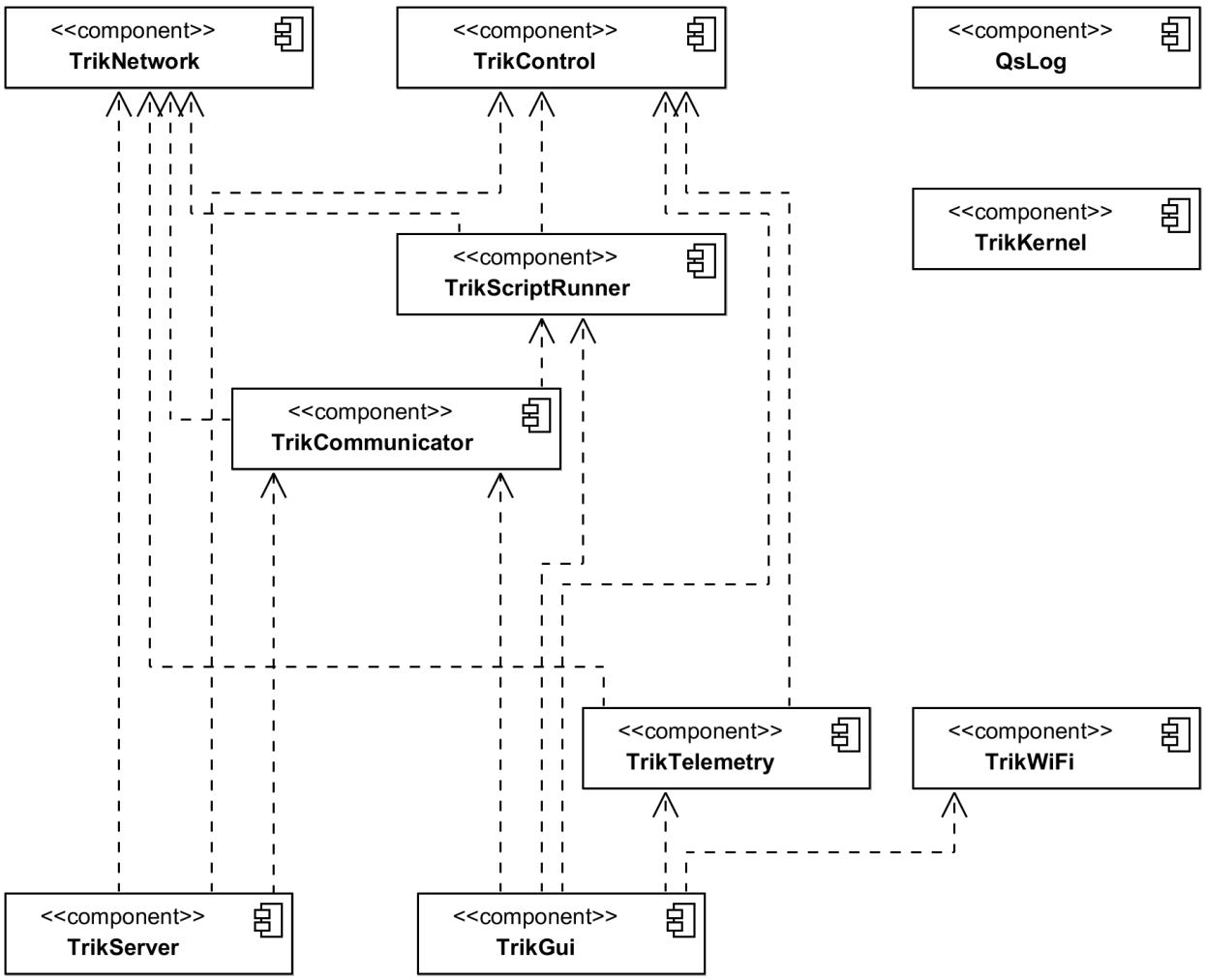

Рис. 3: Упрощённая модель библиотеки TRIK Runtime

2.2.3. QScintilla

Библиотека QScintilla[8] — это адаптация под Qt популярной [4] компоненты для написания редакторов кода Scintilla[9]. На основе библиотеки Scintilla построены такие текстовые редакторы, как Notepad++, Code::Blocks, Geany, SciTE и др.

Библиотека Scintilla поддерживает множество инструментов, позволяющих упростить редактирование кода, включающих в себя настраиваемую подсветку синтаксиса для различных языков программирования, отображение номеров строк в специальном поле, подсветку парных скобок, возможность установки нескольких типов маркеров на каждую

[8]What is QScintilla, URL: https://riverbankcomputing.com/software/qscintilla/intro (дата обращения: 18.10.2016)

[9]Scintilla and SciTE, URL: http://www.scintilla.org/ (дата обращения: 02.12.2016)

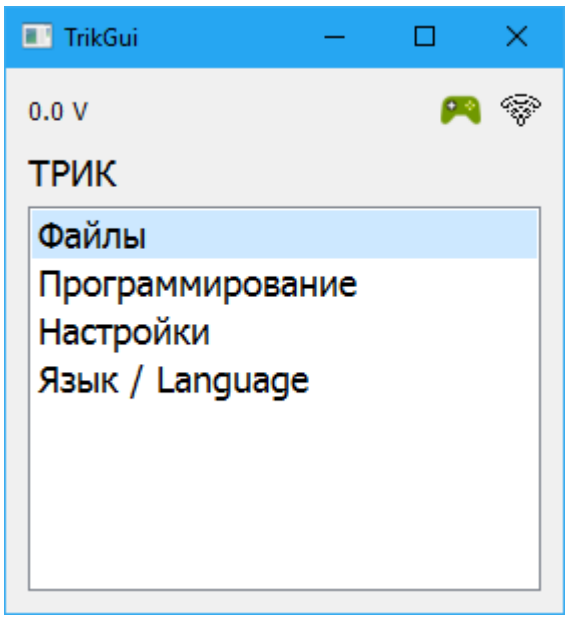

Рис. 4: Графический интерфейс среды TRIK Runtime, запущенной под ОС Windows

строчку и другие полезные функции.

В среде программирования роботов TRIK Studio библиотека QScintilla используется для реализации вкладок с простым редактором кода, обладающим полем с номерами строк, автодополнением и подсветкой синтаксиса.

3. Архитектура

Для поддержки отладки какого-либо текстового языка в TRIK Studio необходимо добавить поддержку его интерпретации на двумерной модели, а так же минимальный набор инструментов для отслеживания состояния выполняющейся программы, который обычно выносится в отдельную компоненту-отладчик.

Как видно из обзора средств отладки текстовых языков программирования, одной из основных возможностей любого отладчика является способность установки точек останова на различные участки кода в целях исследования поведения программы.

Для того, чтобы реализовать поддержку точек останова в TRIK Studio, необходимо было добавить механизм, позволяющий произвольному интерпретатору ставить на паузу 2D-модель TRIK Studio.

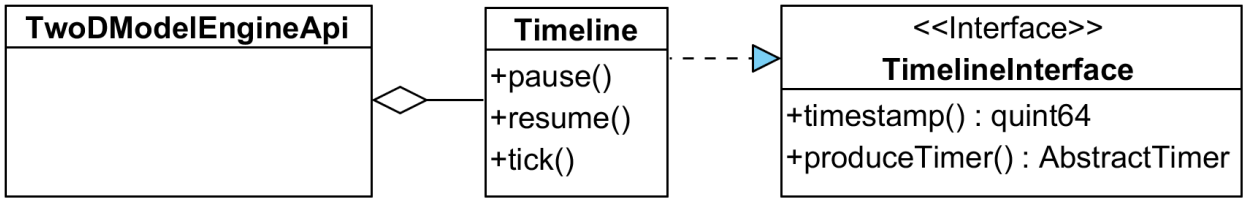

Рис. 5: Диаграмма классов часов 2D-модели

Постановка на паузу модели возможна благодаря тому, что она использует своё собственное время, реализованное через класс Timeline (рис. 5). Timeline является реализацией интерфейса TimelineInterface, предоставляющего собой интерфейс абстрактных часов не привязанных к какому-либо (в том числе и системному) времени. Расчёт физических параметров двумерной модели, изменение положения робота и показаний сенсоров, всё происходит по этим часам. В Timeline был добавлен метод pause(), приостанавливающий отсчёт часов 2D-модели, тем самым останавливая всю симуляцию поведения робота. Интерфейс для управления состоянием двухмерной модели был вынесен в отдельный абстрактный класс TwoDModelDebuggerControlInterface (рис. 6), который затем реализуется в классе Debugger используя добавленную функциональность Timeline.

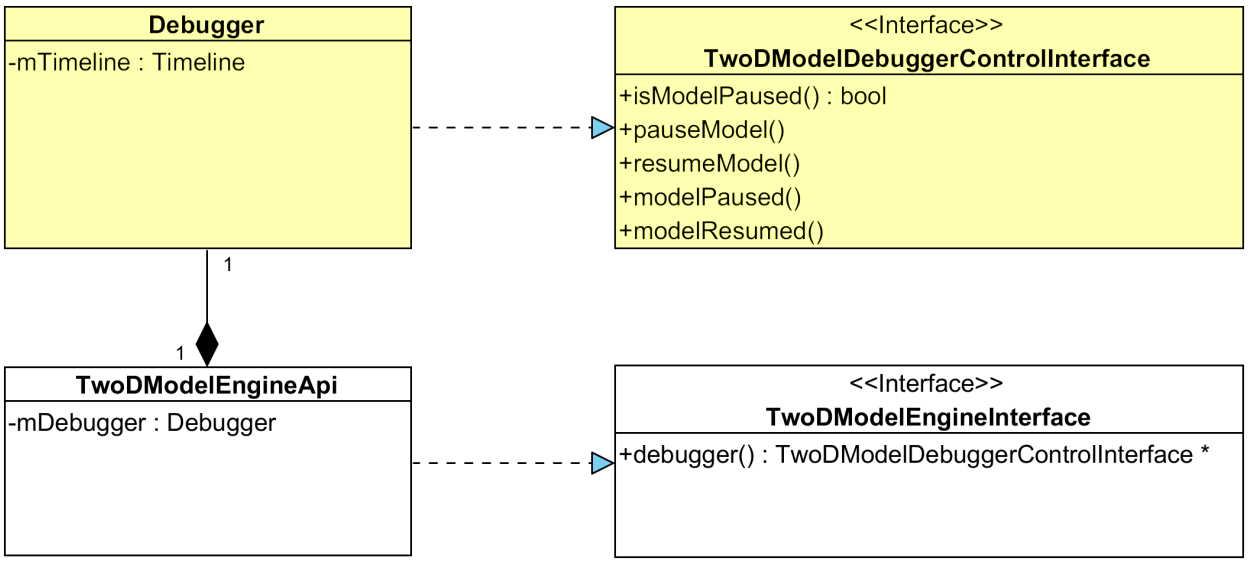

Рис. 6: Диаграмма классов отладчика 2D-модели (здесь и далее добавленные в рамках данной работы классы выделены бледно-жёлтым)

Предоставлением удобного интерфейса для управления интерпретацией языка JavaScript занимается класс TrikQtsInterpreter (рис. 7). За выполнение скриптов отвечает TrikScriptRunner из библиотеки TRIK Runtime. При исполнении программы с помощью TrikScriptRunner общение с контроллером робота осуществляется через интерфейс BrickInterface, предоставляющий методы для доступа к различным устройствам робота. Был написан класс TrikBrick, являющийся реализацией этого интерфейса, который хранит ссылку на двухмерную модель и осуществляет инициализацию классов, эмулирующих устройства реального робота. Эти классы реализуют шаблон проектирования ”Адаптер”, транслируя запросы к контроллеру робота в запросы к частям двухмерной модели.

За интерпретацию диаграмм для роботов TRIK ответственен плагин TrikKitInterpreterCommon, основным классом которого является TrikKitInterpreterPluginBase (рис. 8). Поскольку программы на языке JavaScript поддерживаются только роботами TRIK, то инициализацию и управление интерпретатором TrikQtsInterpreter было решено поместить именно в этот класс. TrikKitInterpreterPluginBase также осуществляет интеграцию интерпретатора скриптов в графический интерфейс TRIK Studio, такую как размещение кнопок интерпретатора на панели ин-

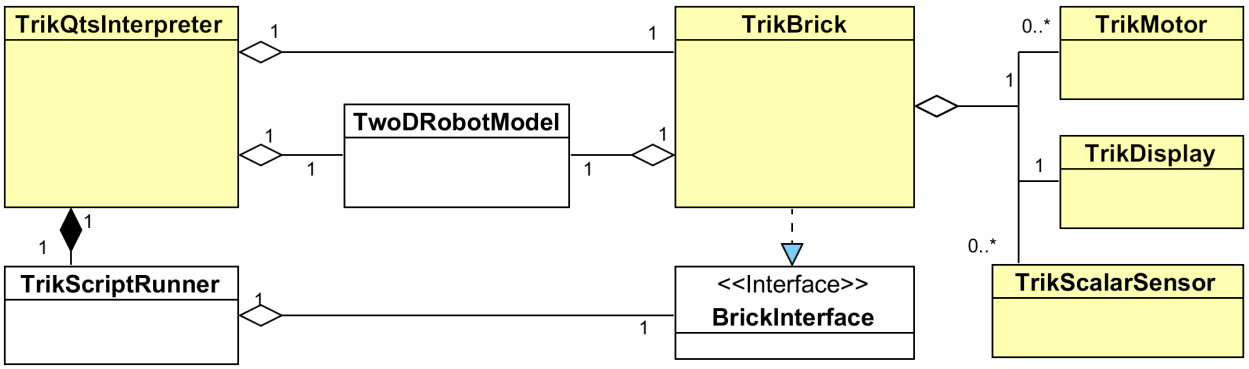

Рис. 7: Диаграмма классов TrikQtsInterpreter

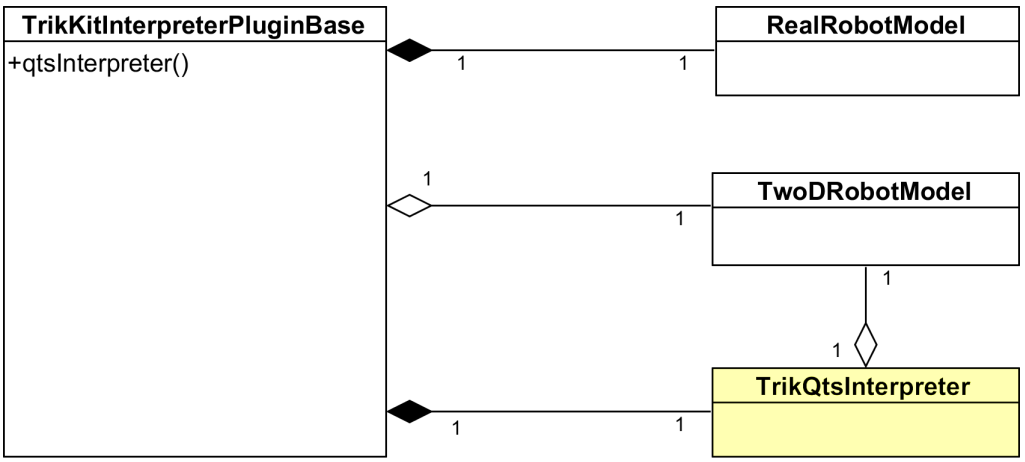

Рис. 8: Диаграмма классов TrikKitInterpreterPluginBase

струментов и передачу текста с текущей открытой вкладки кода интерпретатору. Для интерпретации языка JavaScript TrikScriptRunner использует класс QScriptEngine [6] из поставки Qt. QScriptEngine позволяет отслеживать ход выполнения скрипта с помощью установки класса-агента, унаследованного от QScriptEngineAgent. За отладку выполняемой программы на языке JavaScript отвечает класс TrikQtsDebugger (рис. 9), создаваемый в TrikQtsInterpreter. Этот класс хранит список номеров строк, на которые были поставлены точки останова, и осуществляет регистрацию агента в используемом QScriptEngine, в целях отслеживания номера выполняющейся в данной момент строки кода.

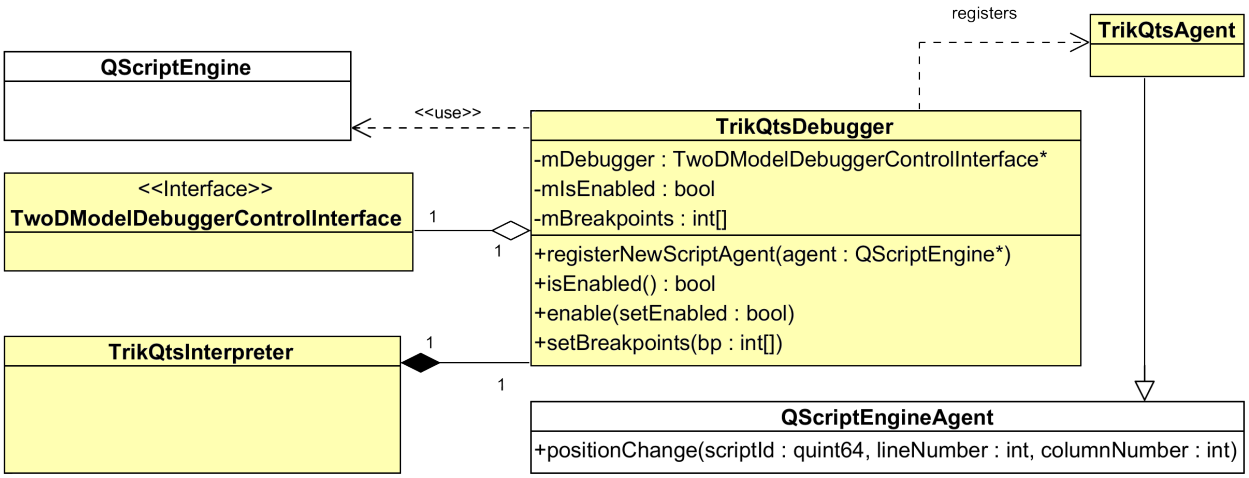

Рис. 9: Диаграмма классов отладчика JavaScript

4. Особенности реализации

4.1. Интерпретатор JavaScript

Для корректной интеграции интерпретатора JavaScript с 2D-моделью была проведена работа по переводу всех методов, доступных из среды выполнения JavaScript, работающих со временем, в том числе с различными таймерами, на использование времени 2D-модели, вместо реального. Это было сделано путём реализации аналогов данных методов в интерфейсе TrikBrick. Затем, в интерпретатор была добавлена возможность добавлять к каждому исполняемому скрипту код, который бы переопределял существующие функции, на соответствующие им новые. Например, заменить функцию, возвращающую таймер реального времени QTimer, на функцию, использующую модельный AbstractTimer.

Данный способ был выбран потому, что было желательно свести изменения в TRIK Runtime к минимуму, так как большинство подобных изменений не имеет смысла для реального робота, использующего TRIK Runtime, а так же, чтобы не добавлять в Runtime зависимости от TRIK Studio. Таким образом, при приостановке модельного времени, приостанавливаются и все таймеры, используемые в интерпретаторе JavaScript, а время, используемое в текстовых программах, согласуется с внутренним временем 2D-модели.

Так же, в процессе интеграции интерпретатора JavaScript в графи-

ческий интерфейс TRIK Studio возникли неожиданные проблемы со стороны кнопок, отвечающих за запуск и остановку интерпретации. Хоть управление их поведением и было вынесено в отдельный класс-менеджер, но фактически вся логика строилась из предположения о том, что существует только один интерпретатор, и это интерпретатор диаграмм. В качестве временного решения, логика отображения кнопок интерпретатора была модифицирована с учётом типа открытой в данной момент вкладки, но с таким решением могут возникнуть проблемы при добавлении новых текстовых интерпретаторов, поэтому в будущем планируется переписать данный компонент полностью.

4.2. Отладчик JavaScript

Для QScriptEngine существует так же свой отладчик QScriptEngineDebugger, однако его использование в случае с интерпретатором TRIK Studio оказалось невозможным. QScriptEngineDebugger может работать только в том же потоке, что и GUI, а QSciptEngine может работать только в том же потоке, что и отладчик, следовательно интерпретатор JavaScript обязан находиться в GUI потоке. Это несовместимо с системой по поддержке многопоточных программ, реализованной в TrikScriptRunner (QScriptEngine изначально не поддерживает многопоточность). Поэтому, для отладки интерпретатора используется TrikQtsAgent, реализующий интерфейс QScriptEngineAgent, позволяющий отслеживать ход выполнения программы на интерпретаторе.

В TRIK Studio, для отображения и редактирования кода программ используется библиотека QScintilla. В ней, у каждого редактора есть гибко настраиваемые поля, которые в случае с TRIK Studio используются для отображения номеров строк. Для каждого поля можно задать несколько типов маркеров. Маркер — это специальный символ настраиваемого вида, который можно ставить на строки в заданном поле. С помощью данного механизма маркеров была реализована постановка точек останова на вкладке с редактором кода.

Был создан маркер, отображаемый в виде красного круга, который

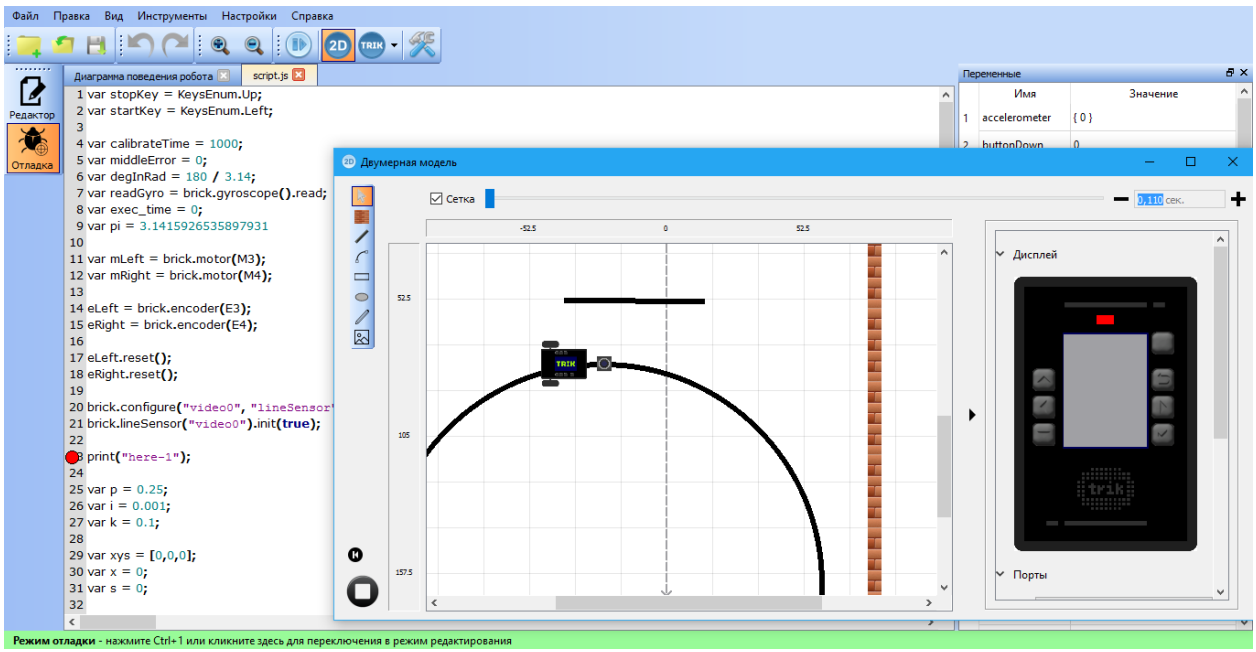

Рис. 10: Точка останова в программе на языке JavaScript в TRIK Studio

можно поставить на любой номер строки в боковом поле редактора. В класс, реализующий вкладки с текстовым редактором, был добавлен метод, позволяющий получить номера всех строк, на которые был поставлен данный маркер. Этот список точек останова затем передаётся в TrikQtsInterpreter, который уже оповещает TrikQtsDebugger.

Как это выглядит в среде TRIK Studio можно увидеть на рис. 10. На данном снимке экрана показана вкладка с текстом программы для робота на языке JavaScript, в которой на двадцать третью строку поставлена точка останова (красный круг). Была запущена интерпретация этой программы, которая дошла до данной строки и перешла в режим ожидания. Теперь можно проверить в каком состоянии находится робот, просмотреть показания его сенсоров, может быть, поменять его местоположение, и продолжить интерпретацию.

5. Апробация

5.1. На примерах из поставки TRIK Studio

Вместе с TRIK Studio поставляется набор программ-примеров для роботов, в которых задействованы различные возможности робототехнических контроллеров и используемых ими сенсоров. Например, этот набор включает в себя программы для следования по линии, используя сенсор света, и для езды вокруг препятствия, используя датчик расстояния. По программам на языке диаграмм из этого набора был сгенерирован код JavaScript и затем запущен на интерпретацию в том же окружении. Во всех случаях поведение робота при интерпретации JavaScript соответствовало его поведению при интерпретации диаграмм. На рис. 11 запущена интерпретация программы на языке JavaScript, сгенерированной для примера alongTheLine, в котором робот едет по нарисованной линии, используя два датчика света.

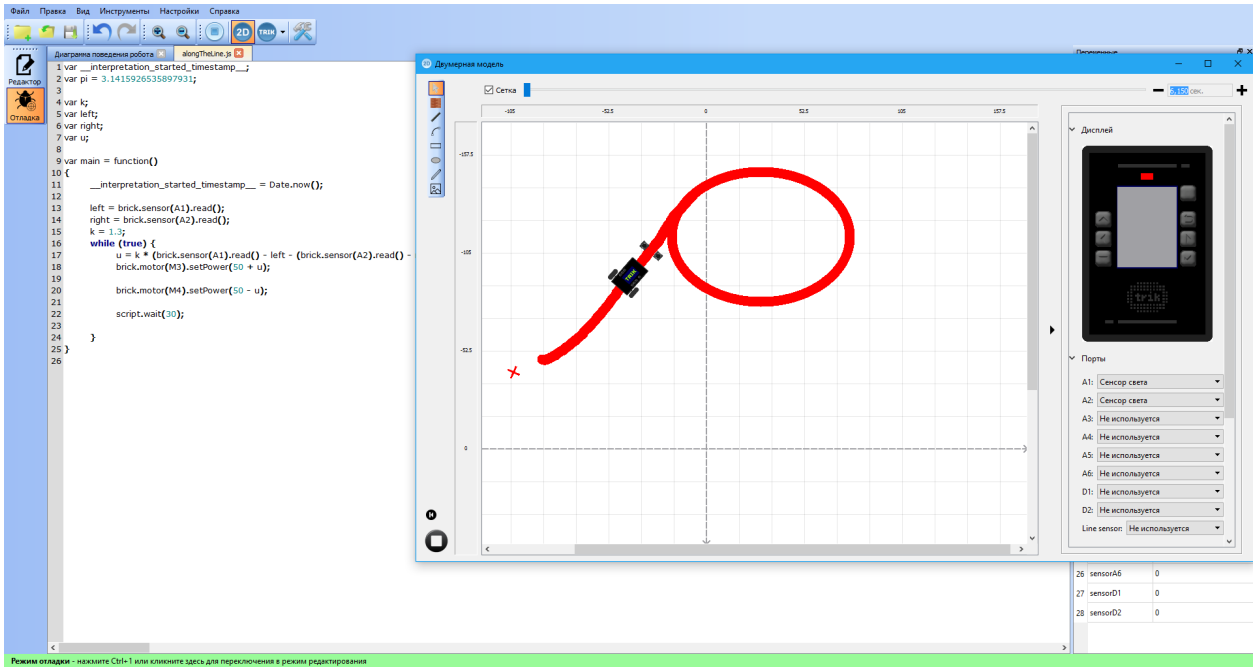

Рис. 11: Интерпретация JavaScript запущенная для примера alongTheLine

5.2. На примере онлайн-курса

На образовательной платформе Stepik был создан краткий курс с задачами на написание программы на языке JavaScript в TRIK Studio. Он использовался для проверки того, что система TRIK Studio, проверяющая задания, будет работать и с программами на языке JavaScript. За основу был взят уже существующий курс про программированию роботов в TRIK Studio. Так как интерпретатор JavaScript управляет 2D-моделью через тот же интерфейс, что и интерпретатор диаграмм, оказалось возможным использовать те же файлы с задачами, что несравненно является большим плюсом данного подхода.

5.3. Через проведение олимпиады на базе TRIK Studio

В марте 2017 года в Сочи прошла олимпиада Национальной Технологической Инициативы[10], где активно использовалось текстовое программирование в TRIK Studio, в том числе и отладка. В ходе подготовки к данной олимпиаде были выявлены и исправлены различные недоработки в реализации интерпретатора JavaScript.

[10]Интеллектуальные робототехнические системы — Олимпиада НТИ, URL: http://nti-contest.ru/profiles/irs/ (дата обращения: 30.03.2017)

Заключение

В результате работы было выполнено следующее:

- разработана архитектура средств отладки текстовых языков программирования в TRIK Studio;
- реализована поддержка интерпретации программ на языке JavaScript В TRIK Studio;
- реализован отладчик языка JavaScript с возможностью ставить точки останова (breakpoints);
- выполнена апробация решения на практике посредством тестирования на стандартных примерах задач, создания краткого онлайн-курса и проведения олимпиады на базе TRIK Studio среди школьников.

Также, данная работа легла в основу выступления на конференции Современные технологии в теории и практике программирования, основное содержание которого было изложено в опубликованных тезисах [11].

Список литературы

[1] Debugging with GDB / Richard Stallman, Roland Pesch, Stan Shebs, et al. The GNU Source-Level Debugger. — Ninth edition. — Free Software Foundation, 2010.

[2] Mordvinov Dmitry, Litvinov Yurii, Bryksin Timofey. TRIK Studio: Technical Introduction // Proceedings of the FRUCT'20. — FRUCT Oy, 2017. — P. 296–308.

[3] Roland H. Pesch. GDB QUICK REFERENCE. — 1999. — URL: `https://web.stanford.edu/class/cs107/gdb_refcard.pdf` (дата обращения: 07.05.2017).

[4] Scintilla and SciTE. Projects using Scintilla. — 2015. — URL: `http://www.scintilla.org/ScintillaRelated.html` (дата обращения: 02.02.2017).

[5] Terekhov Andrey, Luchin Roman, Filippov Sergey. Educational Cybernetical Construction Set for schools and universities // IFAC Proceedings Volumes, Volume 45, Issue 11. — Elsevier Ltd., 2012. — P. 430–435.

[6] The Qt Company Ltd. Qt Script // Qt Documentation. — 2016. — URL: `http://doc.qt.io/qt-5/qtscript-index.html` (дата обращения: 18.09.2016).

[7] Архитектура среды визуального моделирования QReal / А.Н. Терехов, Т.А. Брыксин, Ю.В. Литвинов и др. Системное программирование. Т. 4. — СПбГУ, 2000.

[8] Когутич Д.А., Смирнов М.А., Литвинов Ю.В. Поддержка конструктора EV3 в TRIK Studio // Материалы научно-практической конференции студентов, аспирантов и молодых учёных "Современные технологии в теории и практике программирования". — Изд-во Политехн. ун-та, 2015. — С. 40–41.

[9] Литвинов Ю.В., Кириленко Я.А. TRIK Studio: среда обучения программированию с применением роботов // V Всероссийская конференция «Современное технологическое обучение: от компьютера к роботу» (сборник тезисов). — ЗАО «Полиграфическое предприятие № 3», 2015. — С. 5–7.

[10] Литвинов Ю.В., Мордвинов Д.А. Тестирование среды программирования роботов // ИНСТРУМЕНТЫ И МЕТОДЫ АНАЛИЗА ПРОГРАММ. — Изд-во Политехн. ун-та, 2015. — С. 176–186.

[11] Малютин Д.П., Литвинов Ю.В. Поддержка отладки текстовых языков в TRIK Studio // Материалы научно-практической конференции студентов, аспирантов и молодых учёных "Современные технологии в теории и практике программирования". — Изд-во Политехн. ун-та, 2017. — С. 62–62.