МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

УТВЕРЖДАЮ

Зав.кафедрой,

к. ф.-м. н.

_____ М. В. Огнева

ОТЧЕТ О ПРАКТИКЕ

студента 2 курса 273 группы КНиИТ

Пронина Антона Алексеевича

вид практики: производственная распределенная (научно-исследовательская работа)

кафедра: Кафедра информатики и программирования

курс: 2

семестр: 3

продолжительность: 18 нед., с 01.09.2023 г. по 15.01.2024 г.

Руководитель практики от университета,

старший преподаватель _____ Е. Е. Лапшева

Тема практики:«Разработка системы программирования “КуМир роботы“»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одним из наиболее интересных вопросов, требующих особого внимания в обучении информатике, является вопрос системы обучения программированию. Это связано с тем, что профессия специалистов в области информатики и информационных технологий в какой-то мере начинается со школы. Одним из прямых приложений программирования является робототехника.

Ввиду растущего интереса в сфере образования к обучению робототехнике в школе и широкой распространенности и глубокой степени интеграции современной образовательной системы с этим языком, появилась идея о разработке транслятора с языка КуМир в язык C++. Актуальность идеи заключается в снижении входного порога в область робототехники как со стороны ученика, предоставляя возможность, используя полученные навыки программирования в системе КуМир, заниматься разработкой роботов, так и со стороны преподавателя, уменьшая затраты на приобретение программных и аппаратных средств разработки. В качестве аппаратной платформы данной задачи был выбран электронный конструктор и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов — платы Arduino, ввиду низкой стоимости устройств, периферийных модулей, простоты разработки аппаратных устройств на базе этих плат, высокой модульности систем и их высокой распространенности среди робототехников.

Транслятор с языка КуМир в язык C++ стал отправной точкой процесса создания комплекса аппаратно-программных и методических средств изучения робототехники в школе с использованием языка программирования КуМир. Важным аспектом комплекса является доступность и удобство изучения робототехники. Для повышения доступности изучения, было принято решение разработать отдельный клиент системы программирования КуМир “КуМир Ро-

боты”, направленный на изучение робототехники на основе плат Arduino.

Цель работы — разработка системы программирования роботов на основе плат Arduino на базе исходных кодов системы программирования КуМир. Для выполнения поставленной цели, требуется выполнить следующие задачи:

- проанализировать исходные коды системы программирования КуМир;
- проанализировать историю развития технологий, используемых для разработки системы программирования;
- доработка и устранение ошибок в исходном коде системы программирования КуМир;
- проанализировать системы программирования плат Arduino и их аналоги;
- разработать отдельный клиент для разработки роботов;
- обновить используемые библиотеки и фреймворки для разработки среды программирования;
- разработать плагин для взаимодействия с платами Arduino.

1 Предметная область. Подходы, понятия, средства

1.1 Среда исполнения и язык программирования КуМир

КуМир (Комплект Учебных МИРов) — система программирования, предназначенная для поддержки начальных курсов информатики и программирования в средней и высшей школе с открытым исходным кодом [1]. Спектр возможностей применения данной системы программирования ограничен, ряд стандартных команд позволяет разрабатывать небольшие приложения с целью изучения основ программирования на процедурных языках.

На данный момент язык КуМир — это язык, с которого хорошо начать, чтобы освоить основы алгоритмического подхода и процедурный стиль программирования. Система КуМир [2]в современном ее состоянии (с подсистемой ПиктоМир) состоит из расширяемого набора исполнителей (или роботов), набора систем программирования и вспомогательных утилит и программ. Расширяемый набор исполнителей (роботов) представляет собой отдельные самостоятельные программы и (или) электронно-механические устройства, имеющие собственное пультовое управление (интерфейс), а также возможность локального или сетевого управления из выполняющей системы. Набор систем программирования:

- система программирования КуМир на школьном алгоритмическом языке, состоящая из редактора-компилятора алгоритмического языка с многооконным интерфейсом, интегрированная с выполняющей системой. Система не является полноценным интерпретатором или компилятором школьного алгоритмического языка. Зачастую, программа на школьном алгоритмическом языке компилируется в некий промежуточный код (подобный подход был использован в языке Java), который затем интерпретируется и исполняется с автоматической генерацией точек останова по событи-

ям и шагам (при пошаговом выполнении). В системе программирования КуМир также отсутствует отладчик в его классическом понимании. Это имеет свои причины, так как сильно упрощает процесс освоения и написания системы программирования КуМир, а также отладку программ. Все изменения используемых в школьной программе величин автоматически визуализируются на полях программы. Визуализируются и результаты логических операций. Кроме того, к системе можно подключать любого исполнителя (робота) из набора и программно управлять им;

- система бестекстового, пиктографического программирования ПиктоМир. В ней учащийся собирает программу из команд исполнителя (робота). Управляющие конструкции языка представлены определенной параметрической организацией алгоритмов. Созданная учащимися программа при выполнении передает команды робота и получает его состояние;
- система программирования на производственном языке программирования C++, Python и т.п [3];

1.2 Архитектура системы программирования КуМир

Помимо инструментов разработки, среда обладает рядом функциональных возможностей — просмотр документации, создание и выполнение практикумов, работа с окном в свернутом режиме и пр. Стартовое окно системы программирования КуМир представлено на рисунке 1.

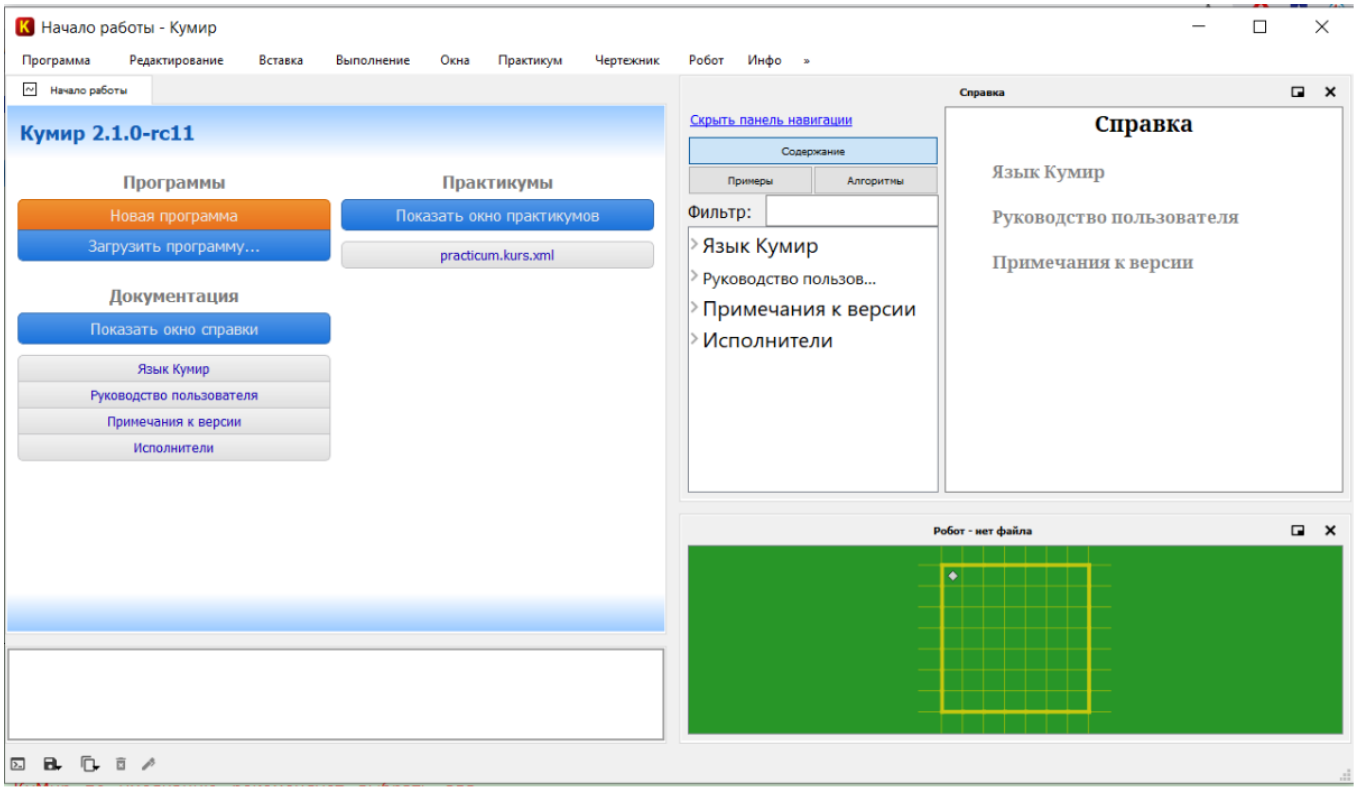

Рисунок 1 — Внешний вид среды программирования КуМир

С точки зрения кода, система программирования КуМир представляет собой монолитное приложение. Исходные коды разделены на 2 основные части:

- библиотеки;
- плагины.

Существует ряд клиентов среды программирования КуМир — для учителя, для ученика старших классов, стандартный клиент. Помимо клиентов, существуют отдельные приложения для компиляции и исполнения кода на языке программирования КуМир. Описанные выше приложения создаются при помощи разделяемого программного кода и вспомогательных инструментов.

К вспомогательным инструментам стоит отнести и ряд функций, упрощающих работу с системой сборки CMake [4]. В исходных кодах можно найти функции упрощающие поиск библиотек для фреймворка QT, определения и установки файлов ресурсов при сборке, а также функции для сборки артефактов приложения(клиентов) из исходных кодов.

Для создания нового артефакта - клиента или приложения командной строки используется CMake-функция $kumir2\_add\_launcher$. По созданному конфигурационному файлу CmakeLists.txt с описанием нового клиента по аналогии с имеющимися, во время сборки приложения, программа-сборщик обнаружит файл конфигурации и с его помощью создаст клиент. Пример конфигурационного файла CmakeLists.txt приведен в листинге 1:

```
project(kumir2-robots)
cmake_minimum_required(VERSION 3.0)

find_package(Kumir2 REQUIRED)

kumir2_add_launcher(
   NAME            kumir2-robots
   SPLASHSCREEN    "splashscreen-classic.png"
  WINDOW_ICON     "window-icon-classic.png"
    APP_ICON_NAME   "kumir2-classic"
   X_ICONS_DIR     "../../../app_icons/linux/hicolor"
    WIN_ICONS_DIR   "../../../app_icons/win32"
   X_NAME          "Kumir Robots Edition"
    X_NAME_ru       "Кумир для роботов"
    X_CATEGORIES    "Education,X-KDE-Edu-Misc"
    CONFIGURATION
  "CourseManager,Editor,ActorArduino,
 ArduinoCodeGenerator\(preload=Files\),
 KumirAnalizer\(preload=Files\),*CodeGenerator,
 KumirCodeRun(nobreakpoints),!CoreGUI\(nosessions\)")
```

Листинг 1 — Пример содержимого файла CMakeLists.txt для нового клиента системы программирования КуМир

Функция $kumir2\_add\_launcher$ позволяет декларативно настроить результат

сборки — указать его название, иконку, а также ряд зависимостей для сборки приложения.

Клиент среды программирования состоит из плагинов и исполнителей. Плагины ссылаются на библиотеки КуМир-а, предоставляя конкретные реализации на основе контрактов [5], описываемых моделями библиотек.

Среди инструментов разработчика существуют скрипты для кодогенерации оснастки исполнителей, развертывания приложения в разных операционных системах и генерации CMake-скриптов.

Библиотеки представляют собой набор базовых сущностей, реализующих определенную часть функционала. В подавляющем большинстве, классы, описывающие область знания не содержат в себе логики работы с данной областью и представляют лишь анемичные модели [6], [7]для хранения состояния. Для работы с моделями используются генераторы или фабрики. Среди библиотек существует собственная реализация AST-дерева [8]. Библиотека содержит модели, описывающие выражения, типы, модули, алгоритмы, переменные и лексемы. Программный код, использующий данные модели представляет собой плагины для кодогенерации.

Кроме AST-деревьев, при помощи библиотек описаны:

- модели уведомлений об ошибках;
- модели для работы с XML-документацией;
- модели для работы с плагинами(базовый функционал, использующийся для создания конкретных реализаций);
- модели для работы с кодировками;
- промежуточные модели для работы с LLVM.

Описанные выше библиотеки используются для создания плагинов — основных блоков приложений. Система плагинов позволяет гибко конфигурировать и изменять набор пользовательских функций. Плагин может инкапсулировать

как функции по трансляции или компиляции языка КуМир в другой язык, так и графическую оболочку приложения, отдельные части его интерфейса.

Список функций, разделенных при помощи плагинов приведен ниже:

- графический интерфейс;
- режим учителя;
- статический анализ программного кода;
- генерация программного кода с языка программирования КуМир в платформо-ориентированный код;
- компиляция программного кода на языке программирования КуМир;
- запуск кода на языке программирования КуМир для выбранной архитектуры процессора и операционной системы;
- генерация программного кода с языка программирования КуМир в инструкции LLVM;
- поддержка языка программирования Python 3.

В ходе ВКР бакалавра был разработан плагин для генерации программного кода с языка программирования КуМир в язык программирования C++, а также исполнитель “Ардуино” [9], предоставляющий набор базовых операций для работы с платами Arduino. Эти доработки являются ключевыми составляющими для разработки отдельного клиента “КуМир Роботы”, предназначенного для программирования роботов на базе плат Arduino.

Анализ архитектуры проекта и имеющихся инструментов позволил определить необходимый функционал для разработки нового приложения. Первым шагом на пути разработки стал анализ зависимостей приложения и обновление основной зависимости — фреймворка QT.

1.3 Обновление зависимостей проекта

Обновление зависимостей проекта является важной частью его поддержки и сопровождения. Вот несколько причин, почему это необходимо:

1. исправление ошибок и уязвимостей: Одна из основных причин обновления зависимостей — исправление ошибок и уязвимостей в используемых библиотеках. Разработчики постоянно работают над улучшением своего программного обеспечения и выпускают исправления проблем, которые были обнаружены. Если вы не обновляете зависимости, ваш проект может содержать уязвимости, которые могут быть использованы злоумышленниками;
2. улучшение производительности и функционала: Обновление зависимостей может также привести к улучшению производительности вашего проекта и добавлению новых функций. Разработчики могут оптимизировать код, исправлять узкие места и добавлять новые возможности в новых версиях библиотек. Если вы не обновляете зависимости, вы можете упустить эти улучшения и новые возможности;
3. совместимость с другими зависимостями: Когда вы обновляете одну зависимость, это может потребовать обновления других зависимостей, чтобы сохранить совместимость между ними. Когда вы используете устаревшие версии библиотек, вы можете столкнуться с проблемами совместимости, которые могут привести к ошибкам или неожиданному поведению вашего проекта;
4. поддержка сообщества и документация: Обновление зависимостей помогает поддерживать активное сообщество вокруг вашего проекта. Разработчики библиотек постоянно работают над улучшением своего программного обеспечения и предоставляют обновления с новыми возможностями или исправлениями. Если вы не обновляете зависимости, вы можете

упустить эти улучшения и использовать менее поддерживаемые версии библиотек.

В целом, обновление зависимостей — важный элемент поддержки проекта. Оно помогает сохранить безопасность, улучшить производительность, добавить новые функции и поддерживать совместимость со сторонними библиотеками [10].

Система программирования КуМир имеет ряд зависимостей — библиотек и фреймворков, используемых для разработки. Основными являются:

- фреймворк QT;
- система сборки CMake;
- интерпретатор языка программирования Python;
- библиотека Zlib;
- библиотека boost;
- библиотека LLVM.

Фаза активной разработки среды и языка программирования КуМир была прекращена в 2014 году. Судя по состоянию репозитория с исходными кодами [11], последние изменения, вносимые в код лишь исправляют небольшие недочеты и недоработки. На данный момент, развитие языка программирования КуМир приостановлено, изменения, в корне меняющие функционал системы программирования не планируются [12]. Большинство исходного кода написано с использованием фреймворка QT, четвертой версии. Поддержка данной версии была прекращена в 2015 году и признана устаревшей [13].

Установка данной версии фреймворка проблематична ввиду отсутствия пакетированных образов. Альтернативой поиску необходимой зависимости является сборка пакета из исходных кодов. Оба рассмотренных выше варианта позволят безопасно разрабатывать приложение, с минимальным риском возникновения ошибок в результате сборки. Однако, использование неактуальных

версий зависимостей предполагает наличие потенциальных уязвимостей, проблем с развертыванием приложений на устройствах потребителей, а также ряд других уязвимостей связанных с прекращением поддержки [14].

Для разработки отдельного клиента с новым функционалом для программирования роботов на базе исходных кодов системы программирования КуМир, потребовалось произвести миграцию исходных кодов на поддерживаемую версию фреймворка $QT-QT5.15$. Среди основных сложностей при миграции стоит выделить следующие:

- изменение программного интерфейса фреймворка;
- несовместимость части библиотек с новой версией;
- несовместимость с системой сборки.

После изменения версии используемого фреймворка на более современную, в первую очередь пришлось решать проблемы с системой сборки. В новой версии фреймворка разработчики приняли решение об изменении подхода к именованию основных строительных блоков и изменению пакета, содержащего данный функционал.

Далее возник ряд предупреждений об использовании устаревших инструкций, например, в версии фреймворка отказались от использования отдельных перечислений, решив использовать одно общее перечисление [15], описывающее широкий спектр свойств. Иными словами, вместо $QT :: Color :: Blue$, теперь стоит использовать $QT :: Blue$. Подобные недочеты составили 70% всех изменений, внесенных при обновлении зависимости проекта.

Серьезной проблемой стало обновление модуля для работы с пользовательской документацией, отображенной на рисунке 2.

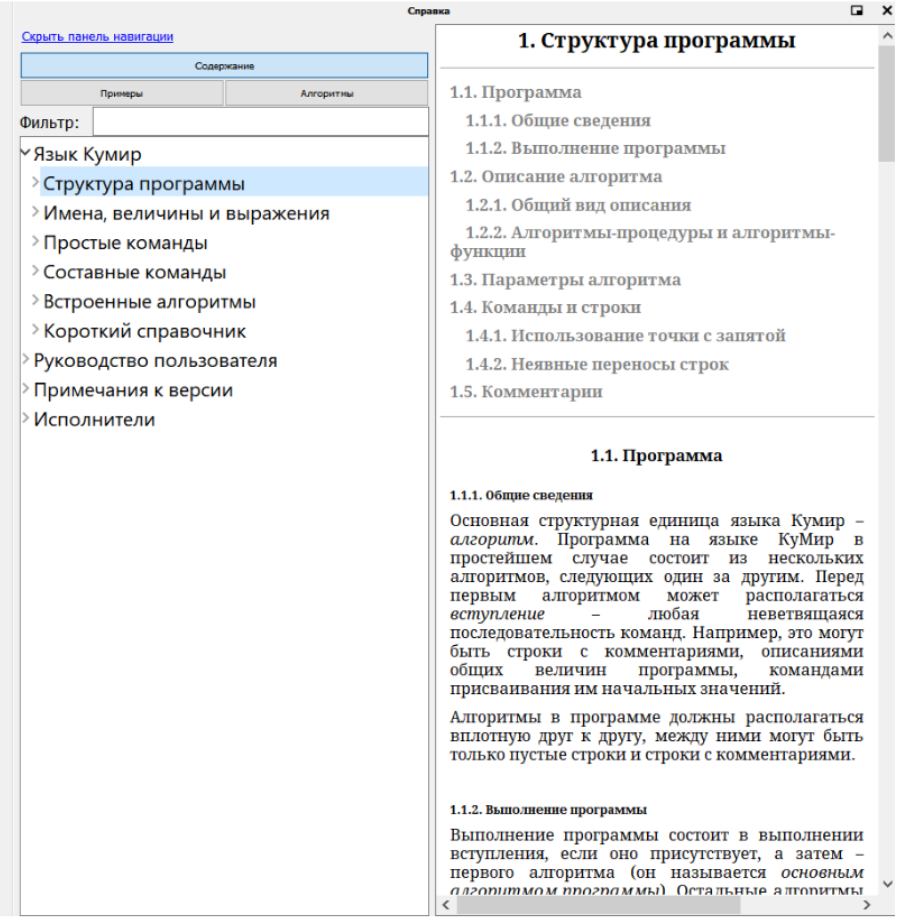

Рисунок 2 — Внешний вид плагина для работы с документацией

Файлы с документацией представляют собой xml-файлы. Основная сущность, используемая для разбора этих файлов была помечена устаревшей в новой версии фреймворка. Ввиду кроссплатформенности приложения, код парсинга xml-документов различался для различных операционных систем. Последней сложностью, возникшей при миграции стало изменение кода обработки пользовательского ввода с клавиатуры. При изучении функционала системы, собранной из исходных кодов, периодически возникала ошибка, приводящая к закрытию приложения. Профилирование приложения не позволило определить причину ошибки. Проблемным местом оказался модуль обработки пользовательского ввода с клавиатуры. Код модуля использовал библиотеку X11 [16].

В результате обновления миграции приложения удалось повысить доступность приложения для современных систем, повысить его надежность, а также

упростить дальнейшую разработку решив проблему совместимости версий зависимостей проекта.

Миграция приложения на новую версию фреймворка стала отправной точкой в процессе разработки отдельного клиента для программирования роботов. Была создана отдельная ветка в репозитории, где была произведена миграция и началась разработка приложения “КуМир Роботы”. Помимо создания конфигурации и определения необходимых зависимостей для приложения, был разработан плагин для работы с платами Arduino. Перед разработкой данного расширения был произведен анализ существующих средств аппаратной обработки плат Arduino.

2 Аппаратно-программный комплекс Arduino

2.1 Робототехника в школе

Arduino — комплекс аппаратно-программных средств с открытым исходным кодом, обладающий низким порогом вхождения как со стороны программирования, так и со стороны электроники. Электронные платы способны считывать и генерировать входные и выходные сигналы. Для передачи последовательности команд управления плате используется язык программирования Arduino и соответствующая среда исполнения. Многолетняя история развития платформы Arduino включает в себя тысячи проектов — от небольших устройств до крупных научных инструментов. Сообщество разработчиков данной платформы состоит из программистов различного уровня — от любителей до профессионалов, собравших и систематизировавших свой опыт взаимодействия с Arduino для помощи в вопросах разработки проектов, полезный как новичкам, так и профессионалам.

Использование платформы Arduino в педагогической деятельности открывает новые возможности для студента и школьника. Проекты, реализуемые в учебном процессе в образовательных учреждениях, могут выполняться и развиваться дома.

В последние годы во многих школах активно организуется основная и дополнительная образовательная деятельность учащихся, связанная с освоением элементов робототехники [17]. Использование платформы Arduino для образовательных учреждений позволяет получить возможность развить навыки программирования на практике, а также освоить азы схемотехники, дает возможность обучающимся освоить основные приемы разработки аппаратной и программной части автономных автоматизированных комплексов. Исследователи вопроса актуальности изучения робототехники на базе платформы Arduino

в образовательном процессе отмечают повышение креативности учащихся [18], активное формирование и оттачивание профессиональных навыков и умений, а также развиваются и осваиваются компетенции естественно-научной направленности, связанные с физико-техническими дисциплинами [19].

2.2 Анализ инструкций и плат Arduino

Семейство плат Arduino включает в себя несколько моделей, которые отличаются по характеристикам, возможностям и цене:

- Arduino Uno: Это самая популярная и распространенная модель платы Arduino. Она оснащена микроконтроллером ATmega328P и имеет несколько цифровых и аналоговых входов/выходов. На плате есть разъемы для подключения сенсоров, дисплеев, моторов и других компонентов. Arduino Uno имеет простой интерфейс, что делает его идеальным для начинающих. Платы полностью совместимы с наборами плат расширения, позволяющими увеличить спектр возможностей применения платы — от добавления ethernet до портативного зарядного устройства с аккумулятором;
- Arduino Mega: Этот микроконтроллер имеет более мощный процессор ATmega2560, обладает большим количеством входов/выходов и памяти. Arduino Mega обычно используется для более сложных проектов, требующих большего количества подключенных компонентов;
- Arduino Leonardo/Arduino Micro: Эти модели оснащены микроконтроллерами ATmega32U4 и обладают возможностью эмуляции устройств USB, таких, как клавиатура или мышь. Arduino Leonardo/Arduino Micro обычно используются для создания интерактивных устройств, таких как игровые контроллеры;
- Arduino Nano: Эта небольшая и компактная модель имеет те же характеристики, что и Arduino Uno, но в более удобном форм-факторе. Arduino

Nano обычно используется в проектах, где необходимо сохранить место на плате. Пригодятся, если вам нужно собрать очень компактное устройство;

• Arduino Due: Эта модель оснащена микроконтроллером ARM Cortex-M3, который позволяет разработчикам выполнять сложные вычисления и работать с более высокой производительностью. Arduino Due часто используется для создания проектов, требующих большой вычислительной мощности.

Кроме вышеперечисленных моделей, существуют и другие варианты, такие как Arduino Y?n, Arduino Nano Every, Arduino MKR и другие. Каждая из них имеет свою спецификацию и предназначение, что позволяет выбрать подходящую модель в зависимости от конкретных потребностей и требований вашего проекта.

Семейства плат отличаются друг от друга рядом технических характеристик, а именно:

• микроконтроллер: Каждая плата Arduino основана на определенном микроконтроллере, таком как ATmega328P, ATmega2560, ATmega32U4 и других. Разные микроконтроллеры имеют разные характеристики, такие как частота работы, объем памяти и количество входов/выходов;

• форм фактор: Существует несколько различных форм-факторов плат Arduino, таких, как Arduino Uno, Arduino Mega, Arduino Nano и другие. Каждая плата имеет свои уникальные размеры и разъемы для подключения различных модулей и дополнительных компонентов;

• функциональность: В разных семействах плат Arduino могут быть различные функциональные возможности. Например, некоторые платы могут иметь встроенный WiFi или Bluetooth модуль, возможность работы с сенсорами или поддержку расширений для специфических задач;

• напряжение питания: Некоторые платы могут работать от батарейного питания или низкого напряжения, в то время как другие могут требовать

более высокого напряжения;

• цена: Разные платы Arduino имеют разные цены в зависимости от их характеристик и функциональности. Некоторые платы могут быть более дорогими из-за дополнительных функций или встроенных модулей.

ля процессоров с различной архитектурой процесс прошивки будет отличаться — для каждой архитектуры имеется своя программа, осуществляющая прошивку платы. Несмотря на это, программирования плат Arduino осуществляется при помощи скетчей — программ на специализированном языке программирования Arduino. Язык является подмножеством языка C++ и содержит набор базовых, кроссплатформенных функций.

В качестве целевой платформы были выбраны платы Arduino Uno ввиду широкой распространенности в сообществе, низкой стоимости и простоты использования.

2.3 Программные средства взаимодействия с платами Arduino

При работе с платами Arduino вопросы, касающиеся работы с платами решаются при помощи Arduino IDE. Интерфейс приложения показан на рисунке 3.

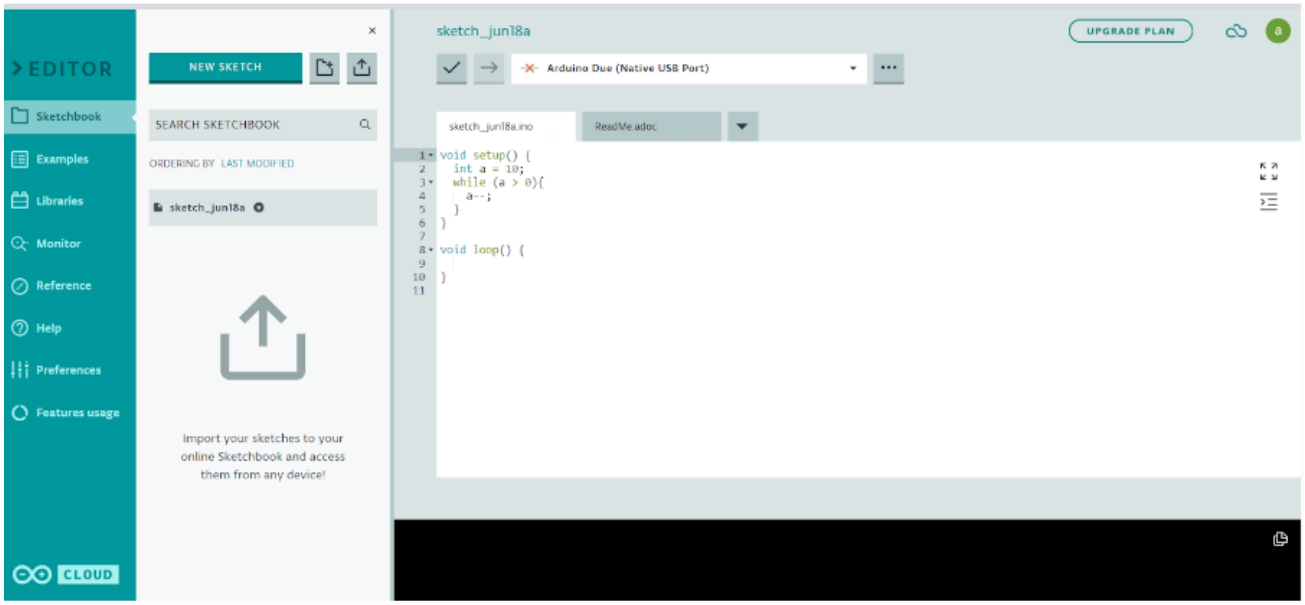

Рисунок 3 — Интерфейс программы Arduino IDE

Для передачи последовательности команд управления плате используется язык программирования Arduino и соответствующая среда исполнения. Программное обеспечение может быть легко использовано на любой операционной системе [20].

На текущей момент существует две версии среды программирования Arduino IDE. Среда программирования позволяет автоматически обнаруживать подключенные к устройству платы, производить их прошивку, а также обрабатывать поступающие с подключенной платы сигналы в реальном времени. Приложение портировано на все операционной системы, программный код для работы с платами имеет общую реализацию, учитывающую различия операционных систем, имеющие значительное влияние на работу с платами.

Исходный код системы программирования Arduino IDE как первой [21], так и второй [22]версии открыт для просмотра и редактирования. Для анализа исходных кодов было решено выбрать более современную версию приложения и рассмотреть алгоритмы взаимодействия приложения с платами.

Вторая версия приложения разработана при помощи языка программиро-

вания Typescript [23]и библиотеки react-js [24]. Приложение может работать как в браузере, так и локально на устройстве пользователя.

При авторизации в приложении у пользователя появляется возможность установить расширение для браузера, как показано на рисунке 4 и начать работу.

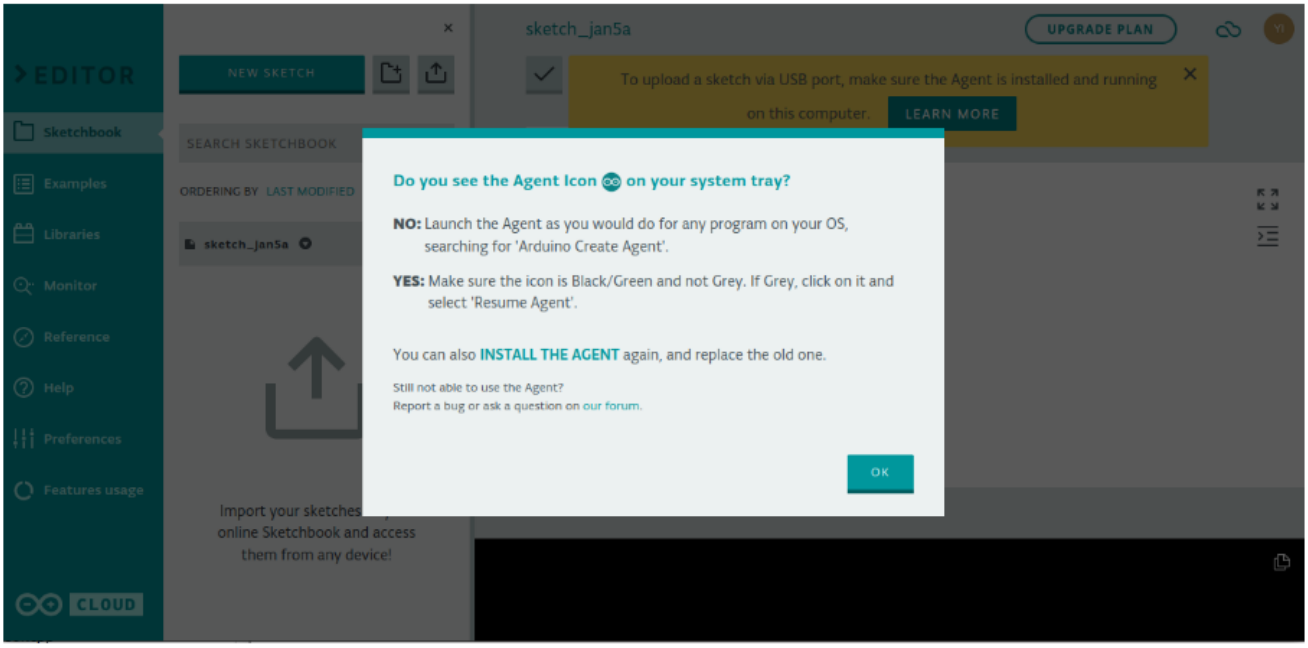

Рисунок 4 — Предложение установить дополнительное ПО для старта работы с Arduino

В репозитории среды программирования можно найти информацию об использовании приложения Arduino-cli [25], [26]для взаимодействия с платами. Arduino-cli — это инструмент командной строки для управления платами Arduino без использования среды разработки Arduino IDE. Он позволяет компилировать, загружать и управлять скетчами Arduino с помощью командной строки, что делает его удобным инструментом для автоматизации процесса разработки и интеграции Arduino в различные среды разработки и рабочие процессы. Arduino-cli поддерживает широкий спектр плат Arduino и различных плат расширения, а также предоставляет возможность установки дополнительных библиотек и платформ.

Приложение обладает набором удобных команд при помощи которых можно организовать сообщение и передачу данных. Среди доступных функций требовалось внедрить в собственный клиент следующие:

- компиляция скетча;
- поиск плат;
- прошивка выбранной платы;
- поиск плат в реальном времени.

Приложение предоставляет два способа для интеграции. Первый предполагает использование приложения по прямому назначению(в качестве приложения командной строки). Общение между потребителем и приложением осуществляется при помощи вызова команд с определенным набором аргументов. Arduino-cli может работать в фоновом режиме — имеется возможность запуска приложения как системную службу или демона в зависимости от операционной системы.

Второй способ обмена информации между приложением и потребителями является открытие grpc-канала [27], [28]. Часть сообщений передается через дуплексный канал связи. Примером такого взаимодействия является опрос подключенных к устройству пользователя плат Arduino. Остальные сообщения передаются по симплексному каналу связи.

В результате анализа было решено разработать плагин для интеграции приложения Arduino-cli как приложения командной строки и предоставления описанных выше возможностей пользователям. С точки зрения имеющейся архитектуры была добавлена библиотека, содержащая контракты для встраивания приложений командной строки, а также был разработан плагин, отвечающий за поиск, соединение и работу с Arduino-cli.

2.4 Система программирования “КуМир роботы“

По окончанию анализа исходных кодов среды и языка программирования КуМир, была составлена схема доработки исходного кода проекта для реализации отдельного клиента среды программирования для изучения робототехники. Схема дорабатываемых библиотек в разрезе архитектуры проекта приведена на рисунке 5. Схема добавленных плагинов в разрезе архитектуры проекта приведена на рисунке 6.

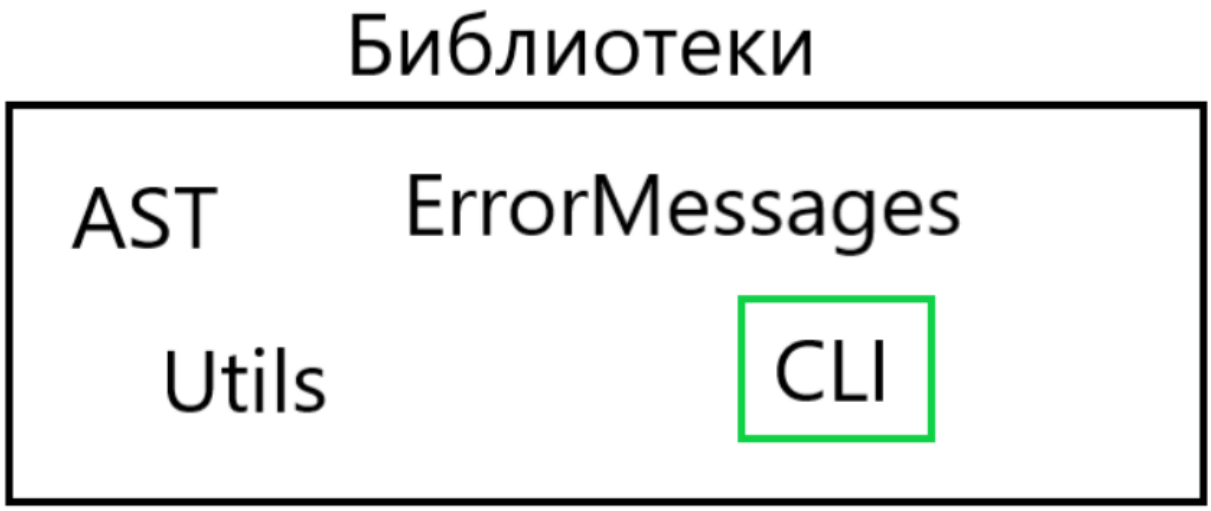

Рисунок 5 — Разработанная библиотека(выделена рамкой)

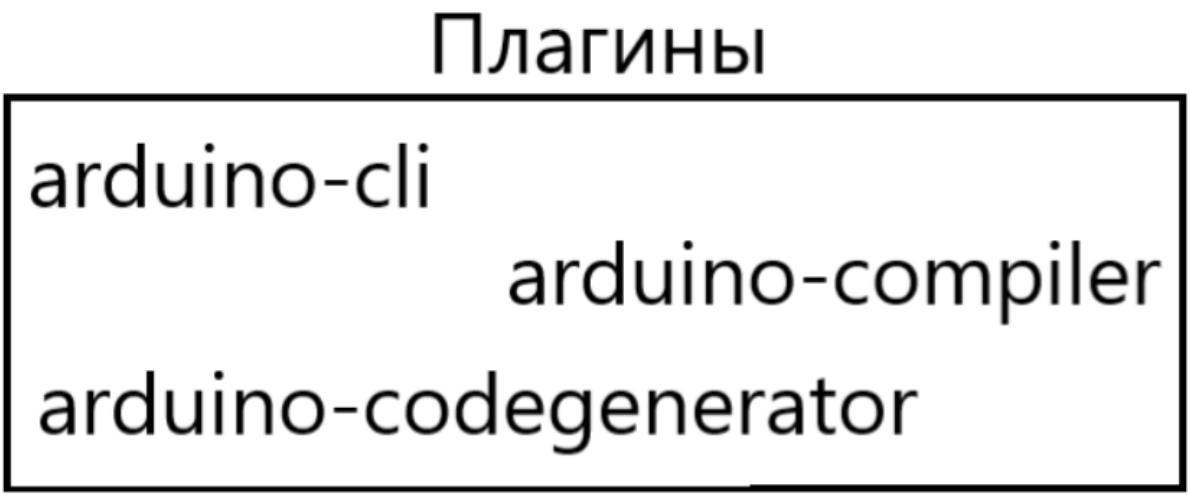

Рисунок 6 — Список разработанных плагинов для работы с платами Arduino

При создании файла конфигурации нового приложения было решено не включать в него стандартный набор исполнителей - Робот, Кузнечик и пр. При

работе в данном клиенте единственным доступным исполнителем является исполнитель “Ардуино”, предоставляющий следующие функции:

- функции аналогового и дискретного ввода/вывода сигналов платы;
- функции для соединения ПК с платами;
- набор констант для работы с портами платы.

Плагины arduino-compiler и arduino-codegenerator работают по аналогии с уже существующими плагинами для КуМира. Использование этих плагинов при разработке нового приложения не составляет сложностей. Плагина Arduino-cli накладывает ряд дополнительных ограничений на пользователя, связанных с установкой дополнительного ПО, а именно самой программы Arduino-cli.

В связи с добавлением внешней зависимости для приложения плагин для работы с платами Arduino содержит виджет проверки состояния доступности программы Arduino-cli. Внешний вид виджета представлен на рисунке 7.

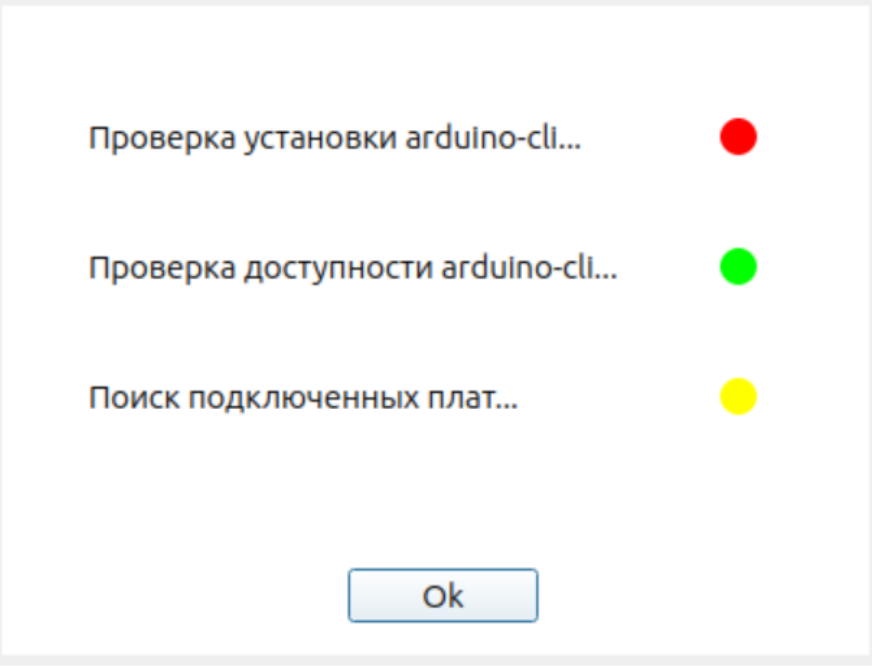

Рисунок 7 — Внешний вид виджета для возможности разрабатывать платы Arduino

Виджет открывается при запуске приложения. Проверка возможности работы включает в себя 3 этапа:

• проверка установки arduino-cli — поиск установленного на устройстве пользователя приложения;

• проверка доступности arduino-cli — проверка наличия прав и корректности установки приложения;

• поиск подключенных плат — опрос портов устройства пользователя на предмет наличия подключенных плат.

Первые 2 пункта проверки являются обязательными и должны быть успешно пройдены для старта работы. Результат работы виджета сохраняется в настройках приложения. Проверка возможности работы может быть произведена не только при запуске клиента “КуМир Роботы“. Кнопка с соответствующим названием находится в меню “Настройки“.

В случае успешного прохождения проверок функционал по работе с платами доступен. Иначе часть возможностей становится недоступной. При попытке воспользоваться недоступной частью функционала, система уведомляет пользователя о невозможности действия. Виджет работает по принципу конечного автомата. Ряд состояний, определяющих работу виджета приведен ниже:

1. проверка установки;
2. проверка доступности;
3. arduino-cli не доступна;
4. поиск плат;
5. платы не найдены;
6. одна или несколько плат найдены;
7. проверки успешно пройдены.

В случае завершения работы виджета в состояниях 1-3, пользователю предлагается перейти по ссылке на официальный сайт приложения и установить его. Для проверки корректности установки Arduino-cli виджет пытается обратиться по 1 из стандартных путей установки приложения в системе. В случае установ-

ки приложения по пути, отличающемуся от стандартного, пользователь может добавить путь в файловой системе к установленному приложению.

Для отслеживания списка подключенных плат используется команда $boardlist$ с флагом $-w$. Список подключенных плат доступен во вкладке “Платы“. При наведении пользователь видит выпадающий список с набором подключенных к устройству плат. Для различия плат с одинаковыми именами используется параметр $FQBN$. Представляет собой строку, которая содержит информацию о модели платы, архитектуре микроконтроллера и версии платформы Arduino. $FQBN$ используется в Arduino IDE и других инструментах для определения параметров и компиляции программного кода для конкретной платы Arduino. Примером $FQBN$ может быть "$arduino : avr : uno$", где "$arduino$" указывает на производителя, "avr" — на архитектуру микроконтроллера и "uno" — на модель платы.

Разработанная система программирования значительно упрощает процесс изучения робототехники. Предыдущая итерация проекта представляла собой консольное приложение-транслятор для генерации скетча для платы Arduino по программному коду на языке программирования КуМир. Отдельный клиент предоставляет возможность интерактивного изучения робототехники используя один интерфейс как для отслеживания состояния плат, так и для работы с платами напрямую — их прошивки, компиляции и настройки скетча.

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены материалы по системе программирования КуМир, аппартно-программной платформе Arduino и фреймворку QT. Были решены следующие задачи:

- проанализированы исходные коды системы программирования КуМир;
- проанализирована история развития технологий, используемых для разработки системы программирования;
- были произведены доработки и устранение ошибок в исходном коде системы программирования КуМир;
- проанализировано семейство плат Arduino и их аналоги;
- разработан отдельный клиент для разработки роботов;
- обновлены используемые библиотеки и фреймворки для разработки среды программирования;
- разработан плагин для взаимодействия с платами Arduino.

Разработка приложения “КуМир роботы“ прошла начальную стадию развития — произошло логическое и физическое разделение исходного кода от основной массы, был разработан ряд отдельных плагинов, формуриующих наполнение приложения. В дальнейшем, планируется ряд доработок для повышения удобства пользования приложения. Список возможных доработок приведен ниже:

- профилирование приложения с целью поиска и устранения утечек памяти;
- создание виджета, демонстрирующего устройство подключенной платы;
- создание плагина-шаблонизатора плат для выбора .h-файла, содержащего набор предзаданных портов и констант;

- динамическое изменение статического анализатора при помощи внедренного шаблона платы;
- создание библиотеки шаблонов, позволяющей пользователям делиться шабонами роботов;
- интеграция с API github для повышения доступности библиотек шаблонов — добавление возможности подключения в клиент репозитория с шаблонами;
- рефакторинг компилятора языка программирования КуМир — отказ от использования кода фреймворка QT.

По окончанию реализации и тестирования среды программирования “КуМир роботы“ для разработки роботов, планируется спроектировать и разработать одного или нескольких роботов на базе аппаратного комплекса Arduino для опробации разработки в школах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Оффициальный сайт КуМир [Электронный ресурс]. — 2022. — URL: https://www.niisi.ru/kumir/. Загл. с экр. Яз. рус.

2 Леонов, А. Г. Методика преподавания основ алгоритмизации на базе системы «КуМир» [Электронный ресурс]. — 2009. — URL: https://inf.1sept.ru/view_article.php?ID=200901701. Загл. с экр. Яз. рус.

3 Кушниренко, А. Г. докл. ПиктоМир: пропедевтика алгоритмического языка (опыт обучения программированию старших дошкольников) // Большой московский семинар по методике раннего обуч. информатике. — М.: ИТОРОИ, 2012.

4 Манаев Р.Г. ТЕХНОЛОГИЯ ВНЕДРЕНИЯ НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ В КРУПНЫХ ВЫСОКОНАГРУЖЕННЫХ СИСТЕМАХ С МИНИМИЗАЦИЕЙ ОШИБОК И ВРЕМЕННЫХ ПОТЕРЬ СО СТОРОНЫ РАЗРАБОТЧИКОВ // Инновации и инвестиции. 2020. №12.

5 Маклафлин Б., Поллайс Г., Уэст Д. Объектно-ориентированнный анализ и проектирование. - СПб.: Питер, 2013. - 608с.: ил.

6 Фаулер, Мартин. Шаблоны корпоративных приложений. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2016. - 544с.: ил. - Парал. тит. англ.

7 Эванс, Эрик. Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем.: Пер. с англ. - СПб.: ООО “Диалектика”, 2020 - 448с.: ил. - Парал. тит. англ.

8 Ахо, Альфред В., лам, Моника С., Сети, Рави, Ульман, Джеффри Д. Компиляторы: принципы, технологии и инструментарий. 2-е изд.: Пер. с англ. - М.: ООО “И.Д. Вильямс”, 2018 - 1184с.: ил. - Парал. тит. англ.

9 Пронин А.А., Синельников Е.А. Модули в языке программирования КуМир 2.0 // Информационные технологии в образовании: сборник / редакционная коллегия: С. Г. Григорьев [и др.]. – Саратов: Саратовский университет [издание], 2022. – Вып. 5: материалы XIV Всероссийской научно-практической конференции «Информационные технологии в образовании» (ИТО-Саратов-2022), Саратов, 28-29 октября 2022 г. – 290 с. : ил. (9,19Мб). – URL: https://sgu.ru/node/197426. – Режим доступа: Свободный. Продолжающиеся издания СГУ на сайте www.sgu.ru. [207-211]

10 Смирнов Максим докл. Модернизация унаследованных приложений // конференция ArchDays 2023

11 Исходный код среды исполнения КуМир [Электронный ресурс]. — URL: https://github.com/a-a-maly/kumir2 Загл. с экр. Яз. рус

12 Кушниренко, А. Г. Опыт интеграции цифровой образовательной среды КуМир в платформу Мирера // Объединенная конференция "СПО: от обучения до разработки": материалы конференции / Под ред. В. Л. Чёрный. — МАКС Пресс, 2022. — С. 24–30.

13 Список версий фреймворка QT [Электронный ресурс]. — URL: https://wiki.qt.io/Portal:Quick_Access

14 Вареница Виталий Викторович, Марков Алексей Сергеевич, Савченко Владислав Вадимович, Цирлов Валентин Леонидович ПРАКТИЧЕСКИЕ АСПЕКТЫ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ ПРИ ПРОВЕДЕНИИ СЕРТИФИКАЦИОННЫХ ИСПЫТАНИЙ ПРОГРАММНЫХ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ // Вопросы кибербезопасности. 2021. №5 (45).

15 Амини Камран. Экстремальный Си. Параллелизм, ООП и продвинутые возможности. — СПб.: Питер, 2021. — 752 с.: ил. — (Серия «Для профессионалов»).

16 Бобков В.А., Черкашин А.С. Обработка и визуализация пространственных данных на гибридном вычислительном кластере // Прикладная информатика. 2014. №4 (52).

17 Шабалин, К. В. Формирование креативных способностей школьников при выполнении проектов на базе платформы Arduino / К. В. Шабалин // Педагогическое образование в России. — 2022. — No 2. — С. 135–140.

18 Глазов Сергей Юрьевич, Сергеев Алексей Николаевич, Усольцев Вадим Леонидович ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ ПЛАТФОРМЫ ARDUINO В УЧЕБНОМ ПРОЦЕССЕ ПЕДАГОГИЧЕСКОГО ВУЗА И ОБЩЕОБРАЗОВАТЕЛЬНЫХ ШКОЛ // Известия ВГПУ. 2021. №10 (163).

19 Серёгин, М. С. Использование платформы Arduino в образовательной деятельности / М. С. Серёгин // Инновационная наука. — 2019. — No 6. —С. 62–64.

20 Серёгин, М. С. Использование платформы Arduino в образовательной деятельности / М. С. Серёгин // Инновационная наука. — 2019. — No 6. — С. 62–64.

21 Исходный код приложения Arduino IDE первой версии [Электронный ресурс]. — URL: https://github.com/arduino/Arduino

22 Исходный код приложения Arduino IDE современной версии [Электронный ресурс]. — URL: https://github.com/arduino/arduino-ide

23 Борис Черный. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений. - СПб.: Питер, 2021. - 352 с.: ил. - (Серия “Бестселлеры O’Reily”)

24 Мардан Азат. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. — СПб.: Питер, 2019. — 560 с.: ил. — (Серия «Библиотека про-

граммиста»).

25 Исходный код приложения Arduino CLI [Электронный ресурс]. — URL: https://github.com/arduino/arduino-cli

26 Документация к приложению Arduino CLI [Электронный ресурс]. — URL: https://arduino.github.io/arduino-cli/0.35/

27 Оффициальный сайт протокола grpc [Электронный ресурс]. — URL: https://grpc.io/

28 Индрасири Касун, Курупу Данеш. gRPC: запуск и эксплуатация облачных приложений. Go и Java для Docker и Kubernetes. - СПб.: Питер, 2021. - 224с.: ил. - (Серия “Бестселлеры O’Reily”)

ПРИЛОЖЕНИЕ А

Исходный код плагина, интегрирующего Arduino-cli в проект

```
#include <QPainter>
#include <QPen>
#include <QDebug>

#include "check_item.h"
#include "ui_check_item.h"

CheckItemWidget::CheckItemWidget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::check_item)
{
    ui->setupUi(this);
}

void CheckItemWidget::setLabelText(QString labelText) {
    ui ->label->setText(labelText);
}

CheckItemWidget::~CheckItemWidget()
{
    delete ui;
}

void CheckItemWidget::setState(ArduinoPlugin::CheckItemStates state){
    switch(state){
        case ArduinoPlugin::Error:
            drawCircle(Qt::red);
            return;
        case ArduinoPlugin::Success:
            drawCircle(Qt::green);
```

```
31              return;
32          case ArduinoPlugin::Warning:
33              drawCircle(Qt::yellow);
34              return;
35      }
36  }
37
38  void CheckItemWidget::drawCircle(Qt::GlobalColor color) {
39      QPixmap pm(20, 20);
40      pm.fill();
41
42      QPainter p(&pm);
43      p.setRenderHint(QPainter::Antialiasing, true);
44      QPen pen(color, 2);
45      p.setPen(pen);
46      QBrush brush(color);
47      p.setBrush(brush);
48      p.drawEllipse(3, 3, 15, 15);
49
50      this->ui->statusImage->setPixmap(pm);
51  }
52
53  #ifndef CHECK_ITEM_H
54  #define CHECK_ITEM_H
55
56  #include <QWidget>
57  #include "check_item_states.h"
58
59  namespace Ui {
60  class check_item;
61  }
62
63  class CheckItemWidget : public QWidget
```

```
64 {
65     Q_OBJECT
66
67 public:
68     explicit CheckItemWidget(QWidget *parent = nullptr);
69     ~CheckItemWidget();
70
71     void setState(ArduinoPlugin::CheckItemStates state);
72     void setLabelText(QString labelText);
73
74 private:
75     Ui::check_item *ui;
76
77     void drawCircle(Qt::GlobalColor color);
78 };
79
80 #endif // CHECK_ITEM_H
81
82 #ifndef CHECK_ITEM_STATES_H
83 #define CHECK_ITEM_STATES_H
84 namespace ArduinoPlugin{
85     enum CheckItemStates {
86         Warning,
87         Error,
88         Success
89     };
90 }
91 #endif // CHECK_ITEM_STATES_H
92
93 #ifndef CHECK_STEPS_H
94 #define CHECK_STEPS_H
95 namespace ArduinoPlugin {
96     enum ArduinoPluginCheckState {
```

```
97          InstallationCheck,
98          AvailabilityCheck,
99          CliUnavailable,
100         SearchForBoads,
101         NoBoardsFound,
102         AnyBoardsFound,
103         CheckCompleted
104     };
105 }
106 #endif // CHECK_STEPS_H
107
108 #include "checkarduinopluginitialization.h"
109 #include "ui_checkarduinopluginitialization.h"
110 #include "constants.h"
111
112 CheckArduinoPluginInitialization::CheckArduinoPluginInitialization
113 (QWidget *parent) :
114     QDialog(parent),
115     ui(new Ui::CheckArduinoPluginInitialization)
116 {
117     ui->setupUi(this);
118
119     ui->arduinoCliInstallation->setLabelText("Проверка установки arduino-cli...");
120     ui->arduinoCliAvailability->setLabelText("Проверка доступности arduino-cli...");
121     ui->boardsConnectivity->setLabelText("Поиск подключенных плат...");
122     this->setPalette(QPalette(Qt::white));
123     initCheck();
124 }
125
126 void CheckArduinoPluginInitialization::initCheck(){
127     ui->arduinoCliInstallation->setState(ArduinoPlugin::Error);
128     ui->arduinoCliAvailability->setState(ArduinoPlugin::Success);
129     ui->boardsConnectivity->setState(ArduinoPlugin::Warning);
```

```
130 }
131
132 ArduinoPluginCheckState CheckArduinoPluginInitialization::changeState
133 (ArduinoPluginCheckState currentState){
134
135 }
136
137 CheckArduinoPluginInitialization::~CheckArduinoPluginInitialization()
138 {
139     delete ui;
140 }
141
142 #ifndef CHECKARDUINOPLUGININITIALIZATION_H
143 #define CHECKARDUINOPLUGININITIALIZATION_H
144
145 #include <QDialog>
146 #include <QVBoxLayout>
147
148 #include "check_steps.h"
149 #include "check_item.h"
150 using ArduinoPlugin::ArduinoPluginCheckState;
151
152 namespace Ui {
153 class CheckArduinoPluginInitialization;
154 }
155
156 class CheckArduinoPluginInitialization : public QDialog
157 {
158     Q_OBJECT
159
160 public:
161     explicit CheckArduinoPluginInitialization(QWidget *parent = nullptr);
162     ~CheckArduinoPluginInitialization();
```

```
163
164 private:
165     Ui::CheckArduinoPluginInitialization *ui;
166
167     void initCheck();
168     ArduinoPluginCheckState changeState(ArduinoPluginCheckState currentState);
169 };
170
171 #endif // CHECKARDUINOPLUGININITIALIZATION_H
172
173 #ifndef CONSTANTS_H
174 #define CONSTANTS_H
175 #include <QString>
176
177 #define CHECK_ARDUINOCLI_INSTALLATION (QString("Проверка установки arduino-cli..."))
178 #define CHECK_ARDUINOCLI_AVAILABILITY (QString
179 ("Проверка доступности arduino-cli..."))
180 #define SEARCH_FOR_CONNECTED_DOARDS (QString("Поиск подключенных плат..."))
181 #endif // CONSTANTS_H
182
183 #include "mainwindow.h"
184
185 #include <QApplication>
186 #include <QProcess>
187 #include <QDebug>
188
189 void readTest() {
190     QProcess process;
191 //    process.start("type", QStringList{"whereis"});
192     process.startDetached( "/home/linuxbrew/.linuxbrew/bin/arduino-cli" );
193 //    process.start("find /home -type f -name file-to-search");
194     if( !process.waitForStarted() || !process.waitForFinished() ) {
195         return;
```

```
196     }
197
198     qDebug() << process.readAllStandardError();
199     qDebug() << process.readAllStandardOutput();
200 }
201
202 void qprocTest(QApplication a){
203     auto proc = new QProcess(a.instance());
204
205     auto stdOutReadyCallBack = [&proc](){
206         auto output = QString(proc->readAllStandardOutput());
207         qDebug() << output;
208
209         proc->close();
210         delete proc;
211     };
212
213     auto errorOccuredCallback = [&proc](QProcess::ProcessError error){
214         switch (error) {
215         case QProcess::FailedToStart:
216             qDebug() << "The process failed to start.";
217             break;
218         case QProcess::Crashed:
219             qDebug() <<
220  "The process crashed some time after starting successfully.";
221             break;
222         case QProcess::Timedout:
223             qDebug() << "The last waitFor...() function timed out";
224             break;
225         case QProcess::WriteError:
226             qDebug() <<
227  "An error occurred when attempting to write to the process.";
228             break;
```

```
229         case QProcess::ReadError:
230             qDebug() <<
231 "An error occurred when attempting to read from the process.";
232             break;
233         case QProcess::UnknownError:
234             qDebug() << "An unknown error occurred. ";
235             break;
236         default:
237             break;
238         }
239     };
240 
241     auto stateChangedCallback = [&proc](QProcess::ProcessState state){
242         switch (state) {
243         case QProcess::NotRunning:
244             qDebug() << "The process is not running.";
245             break;
246         case QProcess::Starting:
247             qDebug() << "The process is starting, but the program has not
248 yet been invoked.";
249             break;
250         case QProcess::Running:
251             qDebug() << "The process is running and is ready for
252 reading and writing.";
253             break;
254         default:
255             break;
256         }
257 
258         proc->close();
259     };
260     QObject::connect(proc,
261                      &QProcess::readyReadStandardOutput,
```

```
262                     a.instance(),
263                     stdOutReadyCallBack);
264
265     QObject::connect(proc,
266                     &QProcess::errorOccurred,
267                     a.instance(),
268                     errorOccuredCallback);
269
270     QObject::connect(proc,
271                     &QProcess::stateChanged,
272                     a.instance(),
273                     stateChangedCallback);
274
275     proc->open(QProcess::ReadOnly);
276     proc->setProgram("sh");
277     proc->setArguments(QStringList{"-c", "arduino-cli"});
278 //    proc->startDetached();
279     proc->startDetached("/bin/bash",
280 QStringList{"-c", "whereis arduino-cli", "\n"});
281     proc->waitForStarted();
282     proc->waitForFinished();
283
284     proc->close();
285 }
286
287 int main(int argc, char *argv[])
288 {
289     QApplication a(argc, argv);
290     MainWindow w;
291
292 //    readTest();
293
294     w.show();
```

```
295     return a.exec();
296 }
297
298 #include "mainwindow.h"
299 #include "./ui_mainwindow.h"
300
301 MainWindow::MainWindow(QWidget *parent)
302     : QMainWindow(parent)
303     , ui(new Ui::MainWindow)
304 {
305     ui->setupUi(this);
306
307     m_plugin_dialog = new CheckArduinoPluginInitialization(this);
308     m_plugin_dialog->show();
309 }
310
311 MainWindow::~MainWindow()
312 {
313     delete ui;
314     delete m_plugin_dialog;
315 }
316
317 #ifndef MAINWINDOW_H
318 #define MAINWINDOW_H
319
320 #include <QMainWindow>
321 #include "checkarduinopluginitialization.h"
322
323 QT_BEGIN_NAMESPACE
324 namespace Ui { class MainWindow; }
325 QT_END_NAMESPACE
326
327 class MainWindow : public QMainWindow
```

```
328 {
329     Q_OBJECT
330 
331 public:
332     MainWindow(QWidget *parent = nullptr);
333     ~MainWindow();
334 
335 private:
336     Ui::MainWindow *ui;
337 
338     CheckArduinoPluginInitialization *m_plugin_dialog;
339 };
340 #endif // MAINWINDOW_H
341 
342 
```