**Василина Слинкина,**
***средняя общеобразовательная школа № 20, г. Шадринск, Курганская область,***

**Д. А. Слинкин,**
***Шадринский государственный педагогический университет, Курганская область***

# ОБРАЗОВАТЕЛЬНАЯ РОБОТОТЕХНИКА: ОСНОВЫ ВЗАИМОДЕЙСТВИЯ МЕЖДУ НАСТАВНИКОМ И КОМАНДОЙ

***Аннотация***
В статье рассматриваются эффективные механизмы взаимодействия между наставником и командой при подготовке к робототехническим соревнованиям. Анализируются теоретические аспекты такого взаимодействия, приводится конкретный пример подготовки команды-призера к одному из робототехнических соревнований и взаимодействия автора данной статьи в роли наставника с соавтором — капитаном команды.
***Ключевые слова:*** LEGO MINDSTORMS EV3, образовательная робототехника, алгоритм, программирование, Free Pascal, наставник команды.
***DOI:*** 10.32517/2221-1993-2019-18-4-8-16

Современные робототехнические соревнования для школьников и студентов [2, 3, 4, 7, 8, 12] характеризуются заранее (часто — очень заранее, за несколько месяцев) сформулированными конкурсными заданиями, сложность которых ранжируется в широких пределах. Это могут быть как простейшие гонки по линии, так и манипуляционные задачи, требующие от робота нетривиальных действий с множеством предметов в трех измерениях. Значительно усложнять решение даже простых задач могут ограничения по времени или распределение призовых мест в зависимости от скорости выполнения задания. Залог успеха в подготовке к подобным соревнованиям — не только высокий уровень компетентности команды, но и ее эффективное взаимодействие с учителем, тренером или наставником, превосходящим в знаниях и опыте любого участника команды. И жесткое регулирование подготовки к соревнованиям, и минимальное вмешательство наставника в работу команды являются одинаково неприемлемыми. Первое резко понизит автономность команды в отсутствие помощи от наставника, не позволит команде действовать на соревновании самостоятельно, заблокирует ее работу при возникновении незапланированных изменений в условии задач, в конструкции полигона и т. д. Второе не позволит выйти в лидеры соревнований из-за неверной интерпретации правил состязания или конкурсного задания, неэффективной конструкции робота, неверной реакции на изменение внешних условий и т. д. *Только эффективное взаимодействие между наставником и командой, поддержка баланса интересов между участниками процесса позволят команде стать призером*. Рассмотрим данный вопрос более предметно.

## 1. Характеристика этапов решения конкурсных робототехнических задач

**Решение большинства конкурсных робототехнических задач реализуется в три этапа:**

1) этап моделирования;
2) этап конструирования;
3) этап программирования.

***Контактная информация***

**Слинкин Дмитрий Анатольевич**, канд. пед. наук, доцент кафедры программирования и автоматизации бизнес-процессов, Шадринский государственный педагогический университет, Курганская область; *адрес:* 641870, Курганская область, г. Шадринск, ул. К. Либкнехта, д. 3; *e-mail*: xdsl@list.ru

**Vasilina Slinkina,**
School 20, Shadrinsk, Kurgan Region,
**D. A. Slinkin,**
Shadrinsk State Pedagogical University, Kurgan Region

## EDUCATIONAL ROBOTICS: THE BASICS OF INTERACTION BETWEEN THE COACH AND THE TEAM

***Abstract***
The article discusses the effective mechanisms of interaction between the coach and the team in preparing for robotic competitions. The theoretical aspects of such interaction are analyzed, a specific example of the preparation of the winning team to one of the robotic competitions and the interaction of the author of this article as a coach with a co-author — the team captain is given.
***Keywords:*** LEGO MINDSTORMS EV3, educational robotics, algorithm, programming, Free Pascal, team coach.

***На первом этапе — этапе моделирования*** — разрабатывается общий алгоритм решения задачи, минимально формализованный, часто — в словесно-графическом исполнении. Формируется представление о функциональных возможностях робота и основных алгоритмических конструкциях, необходимых для его программирования.

Данный этап является основополагающим, и допущенные на нем ошибки приводят к серьезным проблемам в реализации проекта, к отбрасыванию готовых конструкций и программ по причине их несоответствия условию исходной задачи, неоптимальности решения и т. п.

Таким образом, при построении модели решения задачи крайне важной является роль наставника команды. Он, в силу своего опыта и знаний, способен оценить эффективность предлагаемых детьми решений, подвергнуть аргументированной критике неудачные предложения, выдвинуть собственные варианты решения задачи, в которые крайне важно включать элементы решений, предлагаемых членами команд, что позволит избежать психологического отторжения от «навязываемого» решения.

***На втором этапе — этапе конструирования*** — создается конструкция робота в соответствии с разработанной на первом этапе моделью. Наставник обсуждает с детьми конструкционные особенности робота и помогает в создании особо сложных или критичных по нагрузке элементов.

Как и на первом этапе, наставник указывает ребятам на ошибки конструкции с натурной демонстрацией возможных проблем.

При использовании сборных конструкций, соединяемых штифтами, важно обращать внимание детей на недопустимость нагрузок вдоль штифтов, так как это часто приводит к разрушению конструкции робота во время движения или выполнения других операций.

Также следует концентрировать внимание участников команд на минимизации количества конструкционных элементов без снижения эффективности конструкции.

При использовании моторов следует отдавать предпочтение прямому вращению, без использования разного рода передач либо при минимизации их количества. Например, при наличии мощного низкоскоростного большого мотора и маломощного высокоскоростного компактного мотора следует, при прочих равных условиях, выбирать большой мотор с прямой передачей вращения вместо компактного мотора с понижающей передачей.

При использовании датчиков важно обратить внимание детей на граничные условия корректности их функционирования. Например, ультразвуковой датчик расстояния LEGO не может оценить расстояние менее трех сантиметров, а датчик гироскопа той же компании перестает корректно функционировать при движении робота более чем в одной плоскости (например, при преодолении горок, наклонных препятствий и т. д.).

Многие из указанных ограничений не декларируются в сопроводительной документации, информация о них может быть получена только по неофициальным каналам или на основе личного опыта наставника.

Разработанная конструкция робота должна быть протестирована на формальное соответствие условиям задачи (например, по линейным размерам) и подвержена механическим и программным испытаниям с целью определения ее принципиальной пригодности для решения задачи.

Например, если одна из подзадач конкурсного задания заключается в преодолении горки, то следует в обязательном порядке провести испытание такого преодоления, в результате которого могут быть обнаружены невозможность подъема на горку из-за неправильно расположенного центра тяжести или выступающих частей конструкции, невозможность спуска по причине зацепов отдельных элементов конструкции за вершину горки, невозможность движения по линии на горке из-за меняющегося расстояния между датчиками и поверхностью полигона и т. д.

Другой типичный пример — тестирование поворотов робота вблизи препятствий, бортов полигона и т. д.

Принципиальные ошибки конструкции робота лучше выявить и исправить на данном этапе, что позволит во время программирования робота не отвлекаться на модификацию его механической части и не вносить исправления в уже созданную программу, учитывая эти модификации. Ребята часто игнорируют процесс тестирования конструкции робота, стремясь как можно быстрее перейти к его программированию, и задача наставника — аргументированно доказать важность данного процесса, помочь выявить основные проблемные места конструкции и внести необходимые коррективы.

На данном этапе могут быть также разработаны программные элементы (процедуры, функции, блоки, скетчи и т. д.), решающие простейшие подзадачи основной задачи, из которых на следующем этапе и будет собираться программа.

Сложность ***третьего, заключительного, этапа — этапа программирования*** — напрямую зависит от алгоритмической сложности решения исходной задачи, при этом нет прямой зависимости между сложностью конструирования робота и сложностью его программирования.

Основным требованием к началу третьего этапа является техническая завершенность конструкции робота.

В общем виде алгоритм решения задачи уже разработан на первом этапе, здесь же он конкретизируется в конструкциях выбранного языка программирования.

Как показывает опыт, максимальный эффект на данном этапе приносит комбинация стратегических решений наставника и тактических решений членов команды. Вмешательство наставника в процесс разработки программы должно быть минимальным, оно оправдано только при возникновении тупиковых ситуаций, а также ошибок, не связанных напрямую с создаваемой детьми программой. Наставник может подсказывать, указывать на недочеты, формулировать проблемные ситуации, которые возникнут из-за некорректного решения, однако ядро программы должно быть разработано детьми *самостоятельно*. Это важнейшее требование, исполнение которого позволит ребятам эффективно (несмотря на цейтнот) решать возникающие на соревнованиях проблемы, модифицировать программу в соответствии с изменившимися внешними условиями или формулировками задач.

## 2. Пример решения конкурсной задачи «Царевна-лягушка»

Рассмотрим комплексный пример применения вышеизложенного подхода при решении задачи «Царевна-лягушка» — конкурсного задания в категории «Мастер» Открытого областного турнира по робототехнике «Сказочный турнир» [5]. Проанализируем подготовку к данному испытанию команды «Рубин», золотого призера данного соревнования [10]. Команда «Рубин» в переменном составе в течение двух последних лет является участником и призером многих робототехнических соревнований регионального и российского уровней, ее возглавляет бессменный капитан — соавтор данной статьи, ученица шестого класса, занимающаяся в робототехническом кружке при факультете информатики, математики и физики Шадринского государственного педагогического университета.

По условию задачи робот-ткач должен соткать ковер размером 4×4 кубика по заданному образцу.

Полигон представляет собой поле размером 1200×2400 мм с разметкой черного цвета шириной 30 мм (рис. 1).

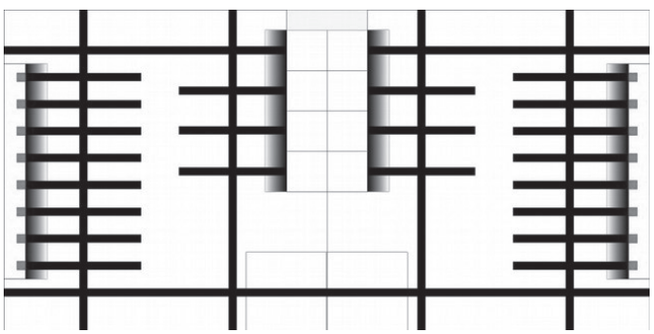

*Рис. 1. Полигон*

В левой и правой частях полигона находится по восемь лего-кубиков размером 4×4×2 единицы, по одному у каждого подъездного отрезка. Кубики могут быть зеленого, синего, желтого и красного цветов, их расположение задается случайным образом перед каждой зачетной попыткой (рис. 2). Следует заметить, что термин «кубик» является в данном случае несколько некорректным, однако ради удобства именно его мы будем в дальнейшем использовать.

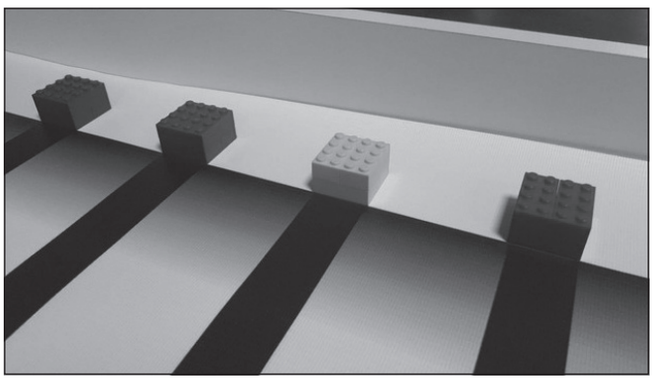

*Рис. 2. Расположение кубиков*

В прямоугольной зоне выгрузки в центре поля (восемь белых квадратов) размером 300×600 мм требуется создать мозаику из кубиков так, чтобы при просмотре с одной из четырех сторон изображение совпало с шаблоном, который выдается командам перед началом турнира и не меняется на протяжении всего турнира. Кубики, составляющие ковер, должны стоять настолько близко друг к другу, насколько это возможно, при этом расстояние между ними не должно превышать 3 мм. В турнирном испытании возможны изменения, которые не затронут размер кубиков, их количество и первоначальное расположение.

Полное условие задачи доступно в Положении об открытом областном турнире по робототехнике «Сказочный турнир» [6].

### 2.1. Этап 1. Анализ задачи и разработка модели решения

Сначала, конкретизируя условие задачи, **мы сформулировали ряд вопросов, определяющих будущую конструкцию робота:**

1) Каково количество переносимых за раз кубиков?
2) Каковы способ подъезда к кубикам и способ их захвата?
3) Каков способ определения цвета кубика?
4) Каков способ выгрузки кубика(ов)?

***1. Каково количество переносимых за раз кубиков?***

Для данного испытания отсутствуют формальные ограничения по времени, поэтому потенциально возможен перенос любого количества кубиков за раз — от 1 до 16. Однако захват и перенос в целевое местоположение единичного кубика обладает рядом недостатков, связанных со сложностью позиционирования в целевом местоположении, а захват и перенос одновременно 16 кубиков увеличит громоздкость конструкции, что с учетом небольшого расстояния между бортами полигона и зоной расположения кубиков создаст серьезные проблемы в маневрировании робота.

Анализируя данную ситуацию, мы пришли к выводу, что наиболее эффективным вариантом будет сбор одного ряда (четырех) кубиков за раз. Это позволит как решить проблему с позиционированием в целевом местоположении, так и уменьшить громоздкость конструкции до приемлемых значений.

***2. Каковы способ подъезда к кубикам и способ их захвата?***

Анализируя способы подъезда к кубикам и их захвата, мы рассмотрели два варианта. Первый вариант подразумевал индивидуальный подъезд к каждому кубику и размещение его в контейнере под роботом. Второй вариант предусматривал перемещение вдоль линии расположения кубиков и перемещение кубиков с помощью манипулятора в контейнер перед роботом. Автор первого варианта — капитан команды «Рубин». Анализ данного варианта вскрыл его недостатки, после чего наставником был предложен второй вариант, в котором также присутствовали некоторые проблемы, однако предложения по их устранению оказались значительно менее громоздки, чем в первом варианте.

Рассмотрим преимущества и недостатки указанных вариантов.

Первый вариант позволяет обеспечить точный подъезд к каждому кубику, ориентируясь по направляющей

черной линии. Недостатков такого варианта было отмечено два:

- во-первых, индивидуальный подъезд значительно замедляет решение;
- во-вторых, сборка ковра в целевом местоположении превращается в нетривиальную задачу.

Второй вариант усложняет позиционирование перед каждым кубиком, однако значительно упрощает и ускоряет маневрирование при сборе кубиков и делает примитивным позиционирование в целевом местоположении при сборке ковра. Именно он был выбран нами за основу.

***3. Каков способ определения цвета кубика?***

Определение цвета кубика можно реализовать как сверху, так и с любой из сторон. Вариант «сверху» мы признали предпочтительным, так как при этом гарантируется постоянное расстояние между датчиком и кубиком. Другие варианты не могут этого гарантировать. Большой опыт в работе с датчиками цвета позволил капитану команды сразу предложить данный вариант и аргументировать его эффективность. Вмешательства наставника не потребовалось.

***4. Каков способ выгрузки кубика(ов)?***

Благодаря выбранному расположению контейнера для сбора кубиков — «перед роботом», сборка одного ряда ковра, как предложил капитан команды, будет заключаться в подъезде к целевой зоне, разблокировке контейнера, перемещении вперед на константное расстояние и отъезде назад. Последующие ряды собираются аналогично, с автоматическим сдвигом вперед всех предыдущих рядов ковра. В данном случае наставник лишь указал на необходимость в обязательном выравнивании на линии сборки и отъезде назад после сборки на расстояние, достаточное для гарантированного разворота без касания собранного ковра. Капитан команды признал логичность данных аргументов.

Определив принципиальную конструкцию робота, **мы разработали общий алгоритм решения задачи.** Выбранная траектория движения робота, основные зоны и ключевые точки принятия решения обозначены на рисунке 3.

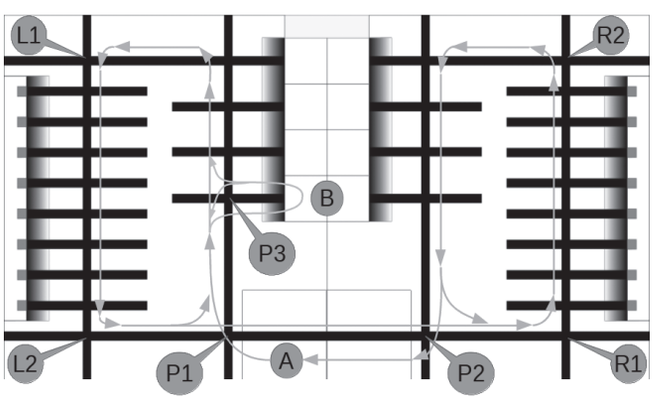


*Рис. 3. Траектория движения робота, ключевые точки*

Зона A — точка старта и финиша робота.

Зона B — целевая точка сборки ковра.

P1 и P2 — первая и вторая точки принятия решения. В этих точках принимается решение о дальнейшем движении робота:

- в зону B, если робот собрал полный ряд;
- в левую часть полигона, если там остались кубики;
- в правую часть полигона, если там остались кубики.

P3 — третья точка принятия решения. В этой точке принимается решение о дальнейшем движении робота:

- в зону A, если ковер полностью собран;
- в левую часть полигона, если там остались кубики;
- в правую часть полигона, если там остались кубики.

L1 и L2 — начальная и конечная точки сбора кубиков в левой части полигона.

R1 и R2 — начальная и конечная точки сбора кубиков в правой части полигона.

Вариант, предложенный капитаном команды изначально, предполагал наличие только одной контрольной точки принятия решения — P1. Однако наставник указал большое количество дополнительных манипуляций, которые придется проделывать для смещения в данную точку после завершения отдельных частей алгоритма. Например, если в левой зоне сбора закончились кубики, то роботу придется после обработки правой зоны возвращаться в точку P1, затем разворачиваться и снова двигаться в правую зону. В результате совместного анализа траекторий движения было принято решение о необходимости трех контрольных точек.

Блок-схема разработанного алгоритма приведена на рисунке 4.

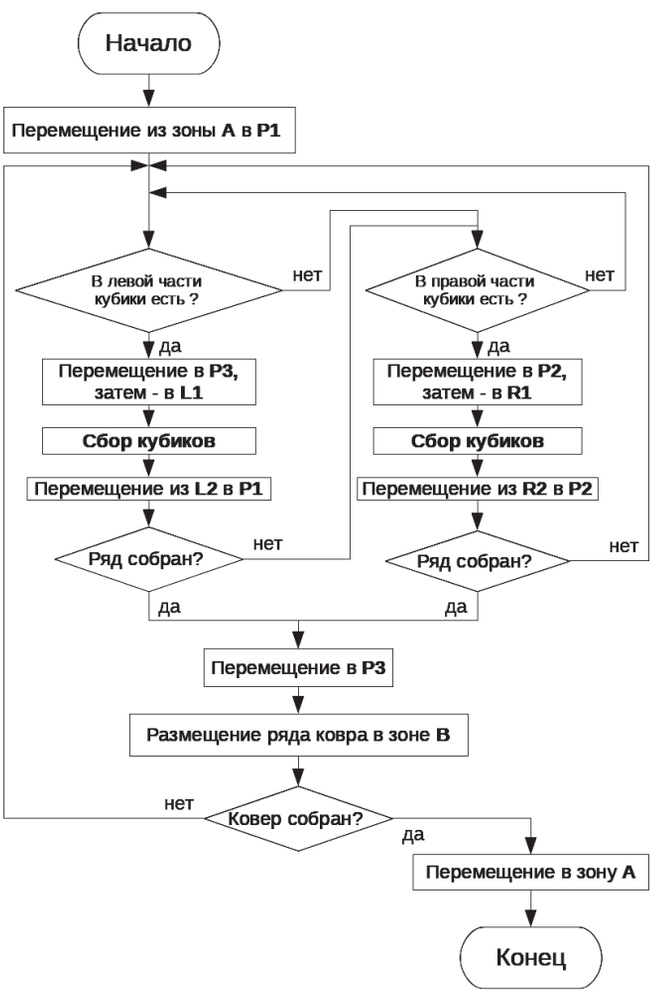


*Рис. 4. Общий алгоритм решения задачи*

Алгоритм сбора кубиков един для левой и правой частей (рис. 5).

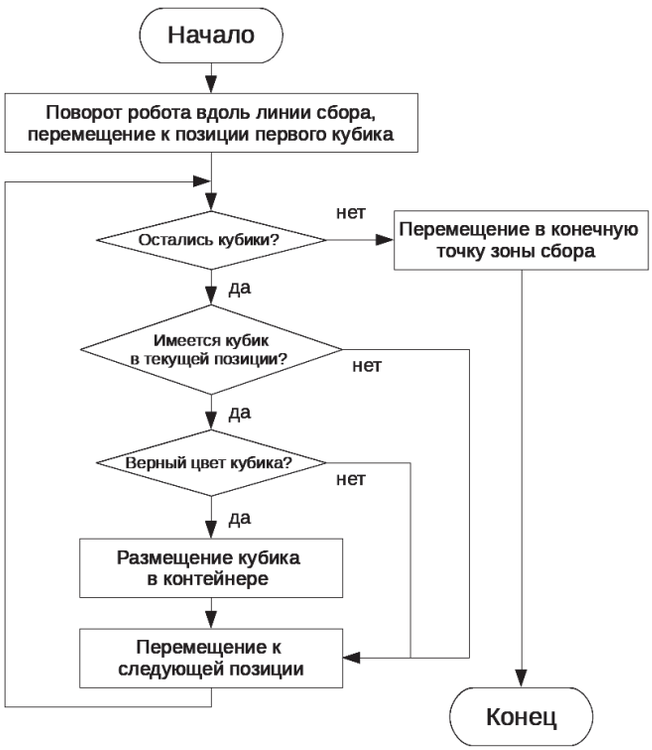


*Рис. 5. Алгоритм сбора кубиков*

Уже во время создания алгоритма сбора кубиков наставник и капитан команды разошлись во мнениях по нескольким вопросам, связанным с будущей реализацией алгоритма, но решили отложить спор до момента испытания конструкции робота, так как разрешить возникшие противоречия мог только натурный эксперимент.

## 2.2. Этап 2. Конструирование робота

Ответив на основные вопросы, связанные с конструкцией робота, мы приступили к его сборке, результат которой показан на рисунках 6 и 7.

В качестве конструкторов мы использовали базовый набор LEGO MINDSTORMS Education EV3 45544 и некоторые элементы из LEGO Technic Motorbike 8051. Также были задействованы дополнительный средний мотор и датчики цвета. Основными концепциями при сборке робота были *простота и надежность*.

Робот состоит из колесной базы, на которую впереди крепится контейнер с фиксатором на базе среднего мотора и тремя датчиками цвета, два из которых предназначены для штатного движения по линии, а третий — для позиционирования над кубиками в процессе их сбора.

Также на робота крепятся манипулятор на основе среднего мотора, перемещающий кубики в контейнер, и четвертый датчик, определяющий цвет очередного кубика. Важным является расположение третьего датчика точно над границей линии, вдоль которой перемещается робот при сборе кубиков, а четвертого датчика — столь же точно над кубиком (рис. 8). Углы поворота движущихся частей манипулятора и фиксатора строго ограничены.

Полный набор изображений, иллюстрирующих конструкцию робота и отдельных его частей, см.: http://rubirobot.ru/files/contests/cover-2019-01-21/cover-2019-01-21-img.7z

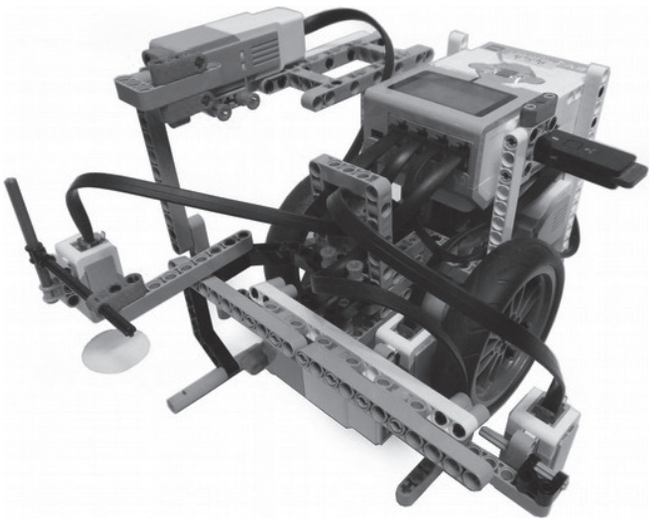

*Рис. 6. Робот-ткач: вид сбоку*

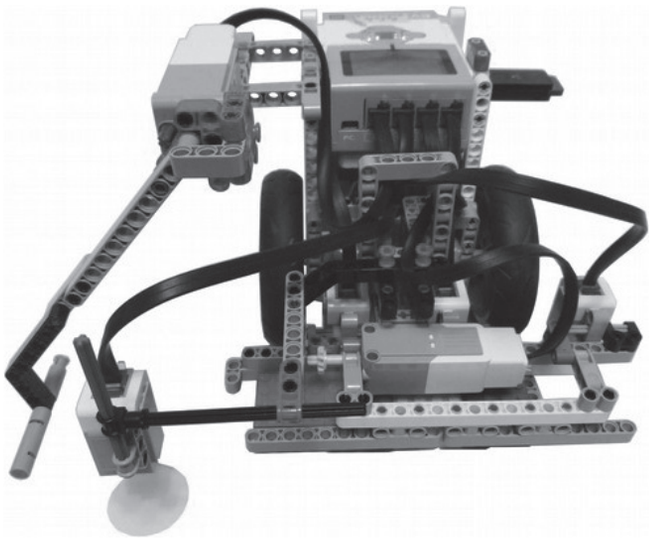

*Рис. 7. Робот-ткач: ряд ковра собран и зафиксирован*

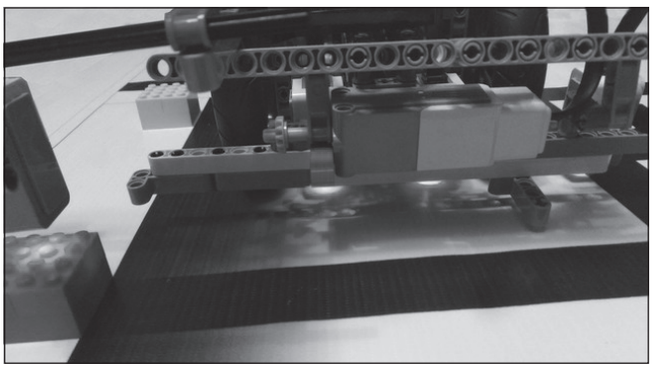

*Рис. 8. Позиционирование робота-ткача в зоне сбора кубиков*

Завершив разработку конструкции робота, мы приступили к ее тестированию. Для программирования робота мы воспользовались библиотекой RubiRobotLib [9, 11].

Конструкция была протестирована на алгоритмах:

- движения по линии до перекрестка с помощью П-регулятора;

- движения по линии до перекрестка с помощью одного (третьего) датчика;
- поворотов влево и вправо вокруг оси и вокруг колеса;
- разворота вокруг оси;
- выравнивания на черной линии;
- движений манипулятора и фиксатора;
- гарантированного определения цвета кубика с помощью многократного чтения значений четвертого датчика.

Более 80 % указанных алгоритмов были реализованы командой самостоятельно, остальные — с помощью наставника команды.

Например, наставник указал на возможность прохождения нескольких перекрестков подряд без торможения робота, а также на необходимость в одних случаях останавливаться на перекрестке, а в других — после перекрестка. В результате командой были реализованы, а затем применены в основном алгоритме три варианта П-регулятора с учетом указанных замечаний.

Также наставник требовал читаемости программных элементов, в результате чего, например, процедуры движения по линии с помощью П-регулятора были переименованы из малоговорящих run, run1 и run2 соответственно в runToCross, runToCrossPlus, runToCrossStop.

На этом же этапе был реализован прототип вспомогательного алгоритма сбора кубиков в любой из зон сбора (см. рис. 5). Как было сказано выше, наставник и капитан команды разошлись во мнениях по нескольким вопросам и только после оживленной дискуссии и набора натурных экспериментов на готовой конструкции робота смогли достичь консенсуса.

Во-первых, по мнению капитана команды, наличие или отсутствие кубика в определенной позиции можно фиксировать по отсутствию цвета на датчике. Наставник же утверждал, что отсутствие кубика в определенной позиции можно гарантировать только после захвата его роботом, а надеяться на датчик нельзя, так как расстояние между полигоном и датчиком цвета невелико и даже в случае отсутствия кубика датчик может возвратить тот или иной цвет. Проведенный эксперимент показал наличие такой вероятности, и капитан команды согласился с мнением наставника.

Во-вторых, по мнению наставника, определив один раз корректный цвет кубика, можно было запомнить его и в дальнейшем пользоваться этой информацией при повторном прохождении зоны сбора. Однако капитан команды указал, что в вероятной ситуации неточного позиционирования цветового датчика над кубиком можно получить неверный цвет (например, при установке датчика на границе красного кубика может быть получен желтый цвет), поэтому цвет надо проверять каждый раз при позиционировании над кубиком. В ответ наставник рассказал капитану команды о принципах вероятности возникновения независимых событий, с чем капитан согласился, но привел довод, связанный с постепенным уменьшением количества кубиков в зонах сбора и возможным увеличением точности позиционирования благодаря этому факту. Проведя эксперимент, мы обнаружили, что точность позиционирования над кубиками действительно повысилась в случае безостановочного прохождения нескольких предварительных пустых участков, поэтому наставник согласился с мнением капитана команды.

Результатом экспериментов стал набор программного кода, с помощью которого наставник и капитан команды отстаивали свое мнение. Данный код был положен капитаном команды в основу реализации прототипа алгоритма сбора кубиков.

Испытания конструкции робота с использованием реализованных алгоритмов показали ее работоспособность, механическую прочность, корректность прохождения поворотов вблизи препятствий (бортов полигона), достаточную точность позиционирования на кубиках.

## 2.3. Этап 3. Разработка программы решения задачи

Процесс реализации основного алгоритма проходил под руководством наставника, но без прямого его участия. Программный код был создан практически в полном объеме командой, за исключением некоторых вспомогательных процедур и функций, разработанных наставником на предыдущем этапе, а также совместной реализации прототипа алгоритма сбора кубиков.

**Основными задачами наставника на текущем этапе являлись:**

- в совместной работе с командой определить структуру наборов данных, заголовков процедур и функций и настоять на их читаемом наименовании;
- помочь команде в реализации алгоритма сбора кубиков на основании прототипа и с учетом ранее определенных структур данных;
- обеспечить контроль читаемости решения, указывать команде на сложные конструкции в целях замены их на более простые и эффективные;
- обеспечить проверку работоспособности отдельных частей алгоритма и всего алгоритма в целом.

Результат совместной работы наставника и команды размещен по адресу: http://rubirobot.ru/files/contests/cover-2019-01-21/cover.pas и доступен всем желающим для изучения и применения.

**Рассмотрим отдельные части данной программы, наиболее важные для понимания сущности решения.**

К данному моменту уже разработаны отдельные процедуры и функции, которые использовались для тестирования конструкции робота. Этот набор можно внедрить в программу без изменений либо создать отдельный модуль для возможности использования его в дальнейшем. Командой осознанно был выбран безмодульный вариант, что со стратегической точки зрения не обеспечивает максимальную эффективность, однако, в ближайшей перспективе ускоряет разработку и отладку программы.

Блок-схема основного алгоритма была разработана ранее (см. рис. 4), и на текущем этапе главными задачами были корректное определение необходимых структур данных и эффективная реализация всех описанных алгоритмов.

Очевидно, что **в качестве общих (глобальных) наборов данных нам понадобятся**:

1. Структура для хранения эталонного ковра, аналог которого необходимо собрать в зоне B:

```
const strok = 4; // строк в ковре
      stolb = 4; // столбцов в ковре

  // сам ковер
  var m: array [1..strok,1..stolb] of Integer =
     (
     (2,3,4,5),
     (3,4,5,2),
     (4,5,2,3),
     (5,2,3,4)
     );
```

Структуру мы определяем в виде двумерного массива с автозаполнением при старте программы. При изменении конкурсного задания подобная структура легко правится как в области контента, так и при изменении размеров массива. Как показало соревнование, и то и другое оказалось востребованным, размер ковра был изменен с $4 \times 4$ на $5 \times 3$, при этом один из 16 кубиков должен был остаться на поле в исходном местоположении.

2. Структура для хранения информации о состоянии левой и правой зон сбора кубиков:

```
// 0 - кубик на месте, 1 - кубика нет (уже взят)
// kub[1] - массив левого набора кубиков
// kub[2] - массив правого набора кубиков
var kub: array [1..2,1..8] of integer = (
 (0,0,0,0,0,0,0,0),
 (0,0,0,0,0,0,0,0));
```

В процессе сбора кубиков ячейки массива постепенно заполняются единицами. Структура используется как во время сбора кубиков, так и во время принятия решения о направлении робота в соответствующую зону сбора. Наличие такого хранилища позволяет быстро проходить точки, из которых кубики были раньше изъяты, а также игнорировать пустые зоны сбора, что в сумме значительно ускоряет алгоритм.

3. Переменная для хранения текущего ряда ковра:

```
var cur: integer=1;
```

Переменная увеличивается каждый раз по окончании сбора одного ряда ковра. С ее помощью определяется факт завершения сбора ковра.

4. Переменная для хранения всех возможных направлений робота в точках P1, P2 и P3:

```
//  p1Left - ключевая точка P1,
    направление робота - влево
//  p1Right - ключевая точка P1,
    направление робота - вправо
//  p2Down - ключевая точка P2,
    направление робота - вниз
//  p3Left - ключевая точка P2,
    направление робота - влево
var Point: (p1Left, p1Right, p2Down, p3Left);
```

В ключевых точках P1, P2 и P3 принимается решение о возможном дальнейшем движении робота, поэтому требуется знать, в какую сторону он в настоящий момент повернут.

Для точки P1 возможны два направления:

- влево, когда робот прибывает из зоны A или по окончании сбора в правой зоне ковра;
- вправо, когда робот прибывает в данную точку после сбора в левой зоне ковра.

Для точки P2 важен момент, когда робот прибывает в нее по окончании сбора в правой зоне ковра (направлен вниз).

В точке P3 решение о дальнейших действиях принимается только после расположения очередного ряда ковра в зоне B и возврата в данную точку (робот при этом будет направлен влево).

Первоначально для идентификации расположения робота командой был использован числовой тип данных (значения 0, 1, 2 или 3 для переменной Point). Для повышения читаемости алгоритма наставник предложил изучить понятие перечислимого типа и применить его для достижения указанной цели, что и было сделано командой по прошествии некоторого времени (значения p1Left, p1Right, p2Down или p3Left для переменной Point).

**Рассмотрим основную программу,** созданную командой «Рубин» в соответствии с алгоритмом, представленным на рисунке 4, проанализируем ее достоинства и недостатки.

В реализации алгоритма используется большое количество ранее определенных в программе функций и процедур. Некоторые из них объединяют часто используемые операции на роботе, другие обрабатывают используемые в программе структуры данных, третьи решают вспомогательные задачи. Перечислим эти задачи.

Перемещение вперед по черной линии до перекрестка с использованием П-регулятора. Выход из процедуры — без остановки моторов:

```
procedure runToCross(speed: double=15; koef:
double=0.2);
```

Перемещение вперед по черной линии до перекрестка с использованием П-регулятора. Затем — перемещение вперед датчиками за перекресток. Выход из процедуры — без остановки моторов:

```
procedure runToCrossPlus(speed: double=15; koef:
double=0.2);
```

Перемещение вперед по черной линии до перекрестка с использованием П-регулятора. Затем — перемещение вперед датчиками за перекресток и остановка моторов:

```
procedure runToCrossStop(maxSpeed: double=15;
koef: double=0.5);
```

Возвращает истину, если кубики имеются в nkub-части полигона (1 — для левой части полигона и 2 — для правой части):

```
function kubikiEst(nkub: integer): boolean;
```

Повороты вокруг оси вправо и влево на 90 градусов:

```
procedure pravo(speed: double=10);
procedure levo(speed: double=10);
```

Поворот влево «змейкой» для решения проблемы близкого бордюра:

```
procedure levoToBorder(speed: double=10);
```

Выравнивание на черной линии:

```
procedure rovno();
```

Перемещение к зоне сбора кубиков, выравнивание перед местоположением первого кубика:

```
procedure podjezd();
```

Сбор кубиков в nkub-части полигона (1 — для левой части полигона и 2 — для правой части):

```
function sbor(nkub: integer):boolean;
```

Код указанных функций находится в программе, адрес местоположения которой в сети Интернет указан выше.

**Реализация основного алгоритма:**

```
1. var fts:boolean;
2. begin
3. // Обнуление интервалов
4. getinterval:=0; setinterval:=0;
5. // Инициализация библиотеки с указанием
6. // набора обязательных датчиков и моторов
7. ev3init([ 1,'color', 2,'цвет', 3,'ev3Color', 4,'color',
8.           'A','mmotor', 'B','largemotor', 'outC','большой', 8,'средний' ]);
9. // Установка способов остановки больших моторов
10.ev3motor1.setHowStop(saBrake); ev3motor2.setHowStop(saBrake);
11.// Установка способов остановки средних моторов
12.ev3Mmotor1.setHowStop(saHold); ev3Mmotor1.setHowStop(saHold);
13.
14.runToCrossStop();
15.point:=p1Left;
16.
17.while cur<=strok do begin
18. while true do Begin
19.
20.  if kubikiest(1) then begin
21.   case point of
22.    p1Left: begin pravo; runToCrossPlus(); end;
23.    p1Right: begin levo; runToCrossPlus(); end;
24.    p2Down: begin pravo; runToCrossStop();
25.     pravo; runToCrossPlus(); end;
26.    p3Left: begin pravo; end;
27.   end;
28.   runToCrossPlus(); runToCrossPlus(); runToCross();
29.   levoToBorder();
30.   runToCross();
31.   podjezd();
32.   fts:=sbor(1);
33.   runToCrossStop();
34.   point:=p1Right;
35.   if fts then break;
36.  end;
37.
38.  if kubikiest(2) then begin
39.   case point of
40.    p1Left: begin pravo; pravo; runToCrossPlus(); end;
41.    p1Right: begin runToCrossPlus(); end;
42.    p2Down: begin pravo; end;
43.    p3Left: begin levo; runToCrossPlus();
44.     levo; runToCrossPlus(); end;
45.   end;
46.   runToCross();
47.   podjezd();
48.   fts:=sbor(2);
49.   runToCrossStop();
50.   levo();
51.   runToCrossPlus(); runToCrossPlus(); runToCrossPlus(); runToCrossStop();
52.   point:=p2Down;
53.   if fts then break;
54.  end;
55. end;
56.
57. // собрали ряд, поехали ткать
58. case point of
59.  p1Left: begin pravo; runToCrossStop(); pravo; end;
60.  p1Right: begin levo; runToCrossStop(); pravo; end;
61.  p2Down: begin pravo; runToCrossStop(); pravo;
62.  runToCrossStop(); pravo; end;
63.  p3Left: begin pravo; pravo; end;
64. end;
65.
66. runToCrossStop();
67. rovno();
68. ev3MMotor1.RunDeg(-100,10,saBrake); ev3MMotor2.RunDeg(70,10).wait;
69. ev3rule.RunDegWait(0,100,5); ev3rule.RunDegWait(0,150,-15);
70. ev3MMotor1.RunDeg(100,10); ev3MMotor2.RunDeg(70,-10);
71.
72. // соткали ряд, возвращаемся
73. levo();
```

```
74. levo();
75. runToCrossStop();
76. point:=p3Left;
77. end;
78.
79. case point of
80. p1Left: begin pravo; pravo; end;
81. p1Right: begin end;
82. p2Down: begin pravo; end;
83. p3Left: begin levo; runToCrossStop(); levo; end;
84. end;
85.
86. ev3rule.RunDegWait(0,250,10);
87. ev3Sound.beep(signal4);
88.
89. ev3.sleep(2000);
90. end.
```

В строках 4–12 производится инициализация библиотеки RubiRobotLib и настройка способов остановки моторов по умолчанию. Инициализация производится в режиме принудительного определения и — в случае необходимости — переподключения внешних устройств, что позволяет эффективно решить основную проблему соревнований по робототехнике: запрет на какие-либо манипуляции с роботом, взятым из карантина, кроме запуска самого робота и запуска единственной программы. Специфика лего-робота такова, что достаточно часто внешние устройства не определяются автоматически при загрузке робота, несмотря на корректное аппаратное подключение. Переопределить устройства можно переподключением проводов после загрузки робота, однако на многих соревнованиях такие действия запрещены. Нам неизвестны программные способы решения данной проблемы с использованием штатной прошивки и ПО от LEGO. К счастью, библиотека RubiRobotLib содержит в себе все необходимое для реинициализации внешних устройств.

Основной цикл алгоритма (строки 17–77) обеспечивает формирование ковра. Он требует первоначального расположения робота в одной из трех ключевых точек. По окончании работы цикла робот также должен находиться в одной из таких точек. По факту, при входе в цикл робот будет находиться в P1, а по выходе из него — в P3. Однако на момент разработки алгоритма было неизвестно, какие изменения будут внесены в условие задачи на соревновании, поэтому наставник настоял на универсализации подхода к расположению робота с помощью CASE-блоков (строки 21–27, 39–45, 58–64 и 79–84), которые позволяют переместить робота в требуемое на данный момент местоположение из любой ключевой точки.

Цикл в строках 18–55 обеспечивает сбор одного ряда ковра. Робот циклически перемещается между зонами сбора, формируя текущий ряд, либо обрабатывает только одну зону сбора, если во второй закончились кубики. Выход из цикла обеспечивается в случае сформированности ряда ковра и срабатывает по завершении функции сбора и перемещении в одну из ключевых точек.

В строках 58–76 обеспечивается перенос сформированого ряда ковра из любой ключевой точки в зону B и возврат в P3.

В строках 79–89 производится возврат из любой ключевой точки (по факту — из точки P3) в зону A, после чего робот издает звуковой сигнал и программа завершает свою работу.

Разработанная программа позволила команде «Рубин» получить максимальное количество баллов за данную задачу и выйти в безусловные лидеры соревнования в категории «Мастер». Большую роль в этом сыграла библиотека RubiRobotLib, решившая как языковую проблему, так и некоторые проблемы в штатном программном обеспечении LEGO. Механизм взаимодействия между наставником и командой при конструировании и программировании робота также показал свою эффективность, прежде всего, в высоком уровне автономности команды на соревновании, умении самостоятельно решать возникающие проблемы.

### Список использованных источников

1. Веб-портал ФГБОУ ВО Шадринского государственного педагогического университета. http://shgpi.edu.ru/

2. Всероссийская робототехническая олимпиада. http://robolymp.ru/

3. Всероссийский робототехнический фестиваль «Робофест». http://www.russianrobofest.ru/

4. Ежегодный Международный фестиваль робототехники «РобоФинист». https://robofinist.ru/

5. Открытый областной турнир по робототехнике «Сказочный турнир». http://shgpi.edu.ru/struktura-universiteta/f11/news/2018/otkrytyi-oblastnoi-turnir-po-robototekhnike-skazochnyi-turnir/

6. Положение об открытом областном турнире по робототехнике «Сказочный турнир». http://shgpi.edu.ru/files/faculties/f11/news/2018/polojenie_skazochnii_turnir.pdf

7. Программа ранней профессиональной подготовки и профориентации школьников. http://юниор-профи.рф/

8. Программа JuniorMasters, конкурсная документация к Открытому корпоративному чемпионату JuniorMasters. https://juniors.ru/competition-tasks/

9. Проект RubiRobot. http://rubirobot.ru

10. Результаты Открытого областного турнира по робототехнике «Сказочный турнир». https://vk.com/robot_shgpu?w=wall-123732266_645

11. *Слинкин Д. А.* Использование языка программирования Free Pascal и программной библиотеки RubiRobotLib для управления роботами на платформе LEGO MINDSTORMS EV3 // Информатика в школе. 2018. № 7. С. 8–12.

12. World Robot Olympiad. https://wro-association.org/