Рассмотрено
На заседании Управляющего
Совета Протокол № 1
От 30.08 2017
Председатель
_____ Ю.В. Владимиров

Утверждено
Приказом № 3
От 04.09 2017
Директор
_____ Т.Ю. Щипкова

Согласовано
С профсоюзным комитетом
Протокол № 1
От 01.09 2017
Председатель
_____ Н.Б. Стуловская

Государственное бюджетное общеобразовательное учреждение города Москвы «Романовская школа»

Дополнительная общеобразовательная (общеразвивающая) программа

Направленность программы – техническая

Форма организации образовательной деятельности – кружок

Название кружка – Робототехника, 7-8 классы

Уровень программы – ознакомительный

Возраст учащихся – 12-14 лет

Срок реализации программы – 1 год

Автор-составитель программы:
Новосельский Алексей Кириллович,
педагог дополнительного образования

Москва
2017

Пояснительная записка

Робототехника — прикладная наука, занимающаяся разработкой автоматизированных технических систем. Робототехника опирается на такие дисциплины, как электроника, механика, программирование.

Робототехника является одним из важнейших направлений научно-технического прогресса, в котором проблемы механики и новых технологий соприкасаются с проблемами искусственного интеллекта. На современном этапе в условиях введения ФГОС возникает необходимость в организации урочной и внеурочной деятельности, направленной на удовлетворение потребностей ребенка, требований социума в тех направлениях, которые способствуют реализации основных задач научно-технического прогресса. К таким современным направлениям в школе можно отнести робототехнику робототехническое конструирование. В настоящий момент во многих образовательных учреждениях России осуществляется внедрение в учебный процесс робототехники. Проводятся соревнования по робототехнике, учащиеся участвуют в различных конкурсах, в основе которых использование новых научно-технических идей, обмен технической информацией и инженерными знаниями.

Программа работы в 7 классе предлагает использование образовательных конструкторов LEGO Mindstorms RCX, программного обеспечения "Brix Command Center" для программирования моделей из Lego на языке NQC (модификация языка Си), среды виртуального моделирования "LDraw" для проектирования конструкций.

Обоснование курса

Работа с образовательными конструкторами LEGO позволяет школьникам в форме познавательной игры узнать многие важные идеи и развить необходимые в дальнейшей жизни навыки. При построении модели затрагивается множество проблем из разных областей знания – от теории механики до психологии, – что является вполне естественным.

Очень важным представляется тренировка работы в коллективе и развитие самостоятельного технического творчества. Простота в построении модели в сочетании с большими конструктивными возможностями конструктора позволяют детям в конце урока увидеть сделанную своими руками модель, которая выполняет поставленную ими же самими задачу.

Изучая простые механизмы, ребята учатся работать руками (развитие мелких и точных движений), развивают элементарное конструкторское мышление, фантазию, изучают принципы работы многих механизмов. *Цель* первой части курса заключается в том, чтобы перевести уровень общения ребят с техникой «на ты», познакомить с профессией инженера: изучение понятий конструкции и ее основных свойствах (жесткости, прочности и устойчивости), элементов черчения.

Вторая часть курса предполагает использование компьютеров и специальных интерфейсных блоков совместно с конструкторами. Важно отметить, что компьютер используется как средство управления моделью; его использование направлено на

составление управляющих алгоритмов для собранных моделей. Учащиеся получают представление об особенностях составления программ управления, автоматизации механизмов, моделировании работы систем. *Цель* второй половины курса состоит в том, чтобы научить ребят грамотно выразить свою идею, спроектировать ее техническое и программное решение, реализовать ее в виде модели, способной к функционированию.

Предлагаемый курс – это интегрированный курс, в котором помимо информационных технологий задействованы:

- ✓ материальная технология (конструктор Лего),
- ✓ физика (механизм, оптика),
- ✓ биология,
- ✓ ОБЖ и многое другое.

Конструктор Лего и программное обеспечение к нему предоставляет прекрасную возможность учиться ребенку на собственном опыте. Такие знания вызывают у детей желание двигаться по пути открытий и исследований, а любой признанный и оцененный успех добавляет уверенности в себе. Учение происходит особенно успешно, когда ребенок вовлечен в процесс создания значимого и осмысленного продукта, который представляет для него интерес.

Важно, что при этом ребенок сам ***строит свои знания***, а учитель лишь консультирует работу.

В окружающем нас мире очень много роботов: от лифта в вашем доме до производства автомобилей, они повсюду. Конструктор Перворобот приглашает ребят войти в увлекательный мир роботов, погрузиться в сложную среду информационных технологий. Программное обеспечение ***Robolab*** отличается дружественным интерфейсом, позволяющим ребенку постепенно превращаться из новичка в опытного пользователя.

Lego позволяет учащимся:

1. Совместно обучаться школьникам в рамках одной бригады;
2. Распределять обязанности в своей бригаде;
3. Проявлять повышенное внимание культуре и этике общения;
4. Проявлять творческий подход к решению поставленной задачи;
5. Создавать модели реальных объектов и процессов;

Лего-технологии позволяют выйти на новые образовательные результаты

- ✓ Умение работать в группе;
- ✓ Решать задачи практического содержания
- ✓ Моделировать и исследовать процессы;
- ✓ Переходить от обучения к учению
- ✓ Роль учителя меняется от наставника-тренера, к союзнику-помощнику.

Требования по аппаратно-программному обеспечению

Из расчёта на 15 человек в кружке.

- Конструктор LEGO Mindstorms – 8 шт.
- Компьютеры – 15 шт.

Программное обеспечение на каждом ПК:

- ОС Windows версии XP SP3 и выше.
- Brix Command Center.

- LDraw.
- MS Office версии 2007 и выше.

Учебно-тематический план занятий.

№ занятия	Название темы	Количество часов (теория/практика)
1	**Техника безопасности** **Роботы вокруг нас.**	1 час(1/0)
2	**От графического программирования к текстовому.** Языки программирования в робототехнике. Язык Си. Язык для программирования роботов NQC.	1 час(1/0)
3-5	**Модель 1. Светофор с датчиком освещенности.** Среда программирования "Brix Command Center". Команды языка NQC для включения ламп, чтения датчика освещенности. Циклы. Переменные.	3 часа(1/2)
6-9	**Виртуальное моделирование в среде пакета ПО «LDraw»** • Обзор пакета ПО «LDraw». • Сборка механических конструкций. • Создание инструкций по сборке.	4 часа(2/2)
10-13	**Модель 2. Телеуправляемый робот-манипулятор.** • Демонстрация готовой модели в виртуальной среде "LDRaw". • Управление 3 моторами по данным с 3 датчиков. • Знакомство с многозадачностью,	4 часа(2/2)
14-17	**Модель 3. Автомобиль для соревнований.** • Эскиз. • Подготовка алгоритма работы. • Отладка. • Соревнования.	4 часа(1/3)
18-23	**Модель 4. Автономный робот-манипулятор.** • Модификация готовой модели в виртуальной среде "LDRaw" • Обработка данных датчика поворота. Контейнеры. Счётчики.	6 часов(2/4)
24-29	**Модель 5. Принтер или сортировщик деталей (на выбор).** • Подготовка модели в виртуальной среде "LDRaw" • Обработка данных датчика освещенности.	6 часов(2/4)
30-34	**Индивидуальная творческая работа.** Моделирование в «LDraw», сборка, подготовка презентации работы.	5 часов(1/4)
Всего за год	**34 часа**	

Литература

1. Индустрия развлечений: ПервоРобот. Книга для учителя и сборник проектов. int.
2. Автоматизированные устройства: ПервоРобот. Книга для учителя. int.
3. MindStorms for schools. Educational division.
4. Наука. Энциклопедия. – М., «РОСМЭН», 2001. – 125 с.
5. Энциклопедический словарь юного техника. – М., «Педагогика», 1988.
6. www.school.edu.ru/int
7. http://www.int-edu.ru

Приложение 1. Памятка по языку NQC.

Конструкции операторов, команд, констант

Операторы

Оператор	Описание
while (cond) body	Выполнить body ноль или более раз, пока условие верно
do body **while** (cond)	Выполните body один или более раз, пока условие верно
until (cond) body	Выполнить body ноль или более раз, пока условие не верно
break	Выход из тела while/do/until
continue	Пропустить до следующего повторения тела while/do/until
repeat (expression) body	Повторить body указанное количество раз
if (cond) stmt1 **if** (cond) stmt1 **else** stmt2	Выполнить stmt1, если условие верно. Выполнить stmt2 (если присутствует), если условие ложно.
start task_name	Начать указанную задачу
stop task_name	Остановить указанную задачу
function(args)	Вызвать функцию, используя поставляемые аргументы
var = expression	Оценить выражение и присвоить переменной
var += expression	Оценить выражение и добавить к переменной
var -= expression	Оценить выражение и вычесть из переменной
var *= expression	Оценить выражение и умножить на переменную
var /= expression	Оценить выражение и разделить на переменную
var |= expression	Оценить выражение и выполнить поразрядное ИЛИ с переменной
var &= expression	Оценить выражение и выполнить поразрядное И с переменной
return	Возврат из функции в место вызова
expression	Оценить выражение

Условия

Условия используются с операторами управления, чтобы принять решения. В большинстве случаев, условие влечет сравнение выражений.

Условие	Значение

true	всегда истинно
false	всегда ложно
`expr1 == expr2`	проверить, равны ли выражения
`expr1 != expr2`	проверить, не равны ли выражения
`expr1 < expr2`	проверить, что одно выражение - меньше чем другое
`expr1 <= expr2`	проверить, что одно выражение меньше чем или равно другому
`expr1 > expr2`	проверить, что одно выражение больше чем другое
`expr1 >= expr2`	проверить, что одно выражение больше чем или равно другому
`! condition`	логическое отрицание условия
`cond1 && cond2`	логическое И двух условий (истинно, если и только если оба условия верны),
`cond1 || cond2`	логическое ИЛИ двух условий (истинно, если и только если, по крайней мере, одно из условий верно).

Выражения

Есть множество различных величин, которые могут использоваться в выражениях, включая константы, переменные и значения сенсора. Отметим, что SENSOR_1, SENSOR_2 и SENSOR_3 - макроопределения, которые расширяются соответственно до SensorValue (0), SensorValue (1) и SensorValue (2).

Величина	**Описание**
`number`	Постоянное значение (например "123")
`variable`	Именованная переменная (например "x")
`Timer(n)`	Значение таймера n, где n - между 0 и 3
`Random(n)`	Случайное число между 0 и n
`SensorValue(n)`	Текущее значение сенсора n, где n - между 0 и 2
`Watch()`	Значение часов системы
`Message()`	Значение последних полученных ИК сообщений

Значения могут быть объединены, используя операторы. Некоторые операторы могут использоваться только в оценке постоянных выражений, что означает, что их операнды должны либо быть константами, либо выражениями, включающими только константы. Операторы перечислены ниже в порядке предшествования (от наивысшей степени к самой низкой).

Оператор	**Описание**	**Ассоциативность**	**Ограничение**	**Пример**

`abs()`
 	
`sign()`
	Абсолютное значение
	Значение операнда
		не применяется
		
		не применяется
				`abs(x)`
				
				`sign(x)`

`++`
 	
`--`
	Инкремент
	
	Декремент
		слева
		
		слева
			только переменные
			только переменные
				`x++ or ++x`
				
				`x-- or --x`

`-`
 	
`~`
	Унарный минус
	Поразрядное отрицание (унарное)
		справа
		справа
			только константа	`-x`
				`~123`

* / %	Умножение Деление Модуль	слева слева слева	только константа	x * y x / y 123 % 4
+ -	Сложение Вычитание	слева слева		x + y x - y
<< >>	Левый сдвиг Правый сдвиг	слева слева	только константа только константа	123 << 4 123 >> 4
&	Поразрядное И	слева		x & y
^	Поразрядное XOR	слева	только константа	123 ^ 4
|	Поразрядное ИЛИ	слева x		x | y
&&	Логическое И	слева	только константа	123 && 4
||	Логическое ИЛИ	слева	только константа	123 || 4

Функции контроллера RCX

Большинство функций требует, чтобы все аргументы были постоянными выражениями (числами или операциями, включающими другие постоянные выражения). Исключения - функции, которые используют сенсор как аргумент, и те, которые могут использовать любое выражение. В случае сенсоров, аргументом должно быть имя сенсора: SENSOR_1, SENSOR_2, или SENSOR_3. В некоторых случаях имеются предопределенные имена (например, SENSOR_TOUCH) для соответствующих констант.

Функции	Описание	Пример
SetSensor(sensor, config)	Сконфигурировать сенсор.	SetSensor(SENSOR_1, SENSOR_TOUCH)
SetSensorMode(sensor, mode)	Установить режим сенсора	SetSensor(SENSOR_2, SENSOR_MODE_PERCENT)
SetSensorType(sensor ,type)	Установить тип сенсора	SetSensor(SENSOR_2, SENSOR_TYPE_LIGHT)
ClearSensor(sensor)	Очистить значение сенсора	ClearSensor(SENSOR_3)
On(outputs)	Включить один или более выходов	On(OUT_A + OUT_B)
Off(outputs)	Выключить один или более выходов	Off(OUT_C)
Float(outputs)	Разрешить плавный останов	Float(OUT_B)
Fwd(outputs)	Установить выходы на курс ВПЕРЕД	Fwd(OUT_A)
Rev(outputs)	Установить выходы на курс НАЗАД	Rev(OUT_B)
Toggle(outputs)	Переключить направление выходов	Toggle(OUT_C)
OnFwd(outputs)	Включить в прямом направлении	OnFwd(OUT_A)

OnRev(outputs)	Включить в обратном направлении	OnRev(OUT_B)
OnFor(outputs, time)	Включить на указанное число сотых секунды. Время может быть выражением.	OnFor(OUT_A, 200)
SetOutput(outputs, mode)	Установить режим выхода	SetOutput(OUT_A, OUT_ON)
SetDirection(outputs, dir)	Установить направление выхода	SetDirection(OUT_A, OUT_FWD)
SetPower(outputs, power)	Установить уровень выходной мощности (0-7). Мощность может быть выражением.	SetPower(OUT_A, 6)
Wait(time)	Ожидать указанное время в сотых секунды. Время может быть выражением.	Wait(x)
PlaySound(sound)	Воспроизвести указанный звук (0-5).	PlaySound(SOUND_CLICK)
PlayTone(freq, duration)	Воспризвести тон указанной частоты на указанное время (в десятых от секунды)	PlayTone(440, 5)
ClearTimer(timer)	Сбросить таймер (0-3), на значение 0	ClearTimer(0)
StopAllTasks()	Остановить все в настоящее время запущенные задачи	StopAllTasks()
SelectDisplay(mode)	Выбрать один из 7 режимов дисплея: 0: часы системы, 1-3: значение сенсора, 4-6: установка выхода. Режим может быть выражением.	SelectDisplay(1)
SendMessage(message)	Послать ИК-сообщение (1-255). Сообщение может быть выражением.	SendMessage(x)
ClearMessage()	Очистить буфер ИК-сообщения	ClearMessage()
CreateDatalog(size)	Создать новый datalog данного размера	CreateDatalog(100)
AddToDatalog(value)	Добавить значение к datalog. Значение может быть выражением.	AddToDatalog(Timer(0))
SetWatch(hours, minutes)	Установить значение часов системы	SetWatch(1,30)
SetTxPower(hi_lo)	Установить уровень мощности ИК-передатчика на низкое или высокое значение	SetTxPower(TX_POWER_LO)

Константы RCX

Многие из значений для функций RCX были названы константами, которые могут помочь сделать код более удобочитаемым. где только возможно Используйте именованные константы, а не простое числовое значение.

Для конфигурации сенсора в SetSensor()	SENSOR_TOUCH, SENSOR_LIGHT, SENSOR_ROTATION, SENSOR_CELSIUS, SENSOR_FAHRENHEIT, SENSOR_PULSE, SENSOR_EDGE

Для режимов в SetSensorMode()	SENSOR_MODE_RAW, SENSOR_MODE_BOOL, SENSOR_MODE_EDGE, SENSOR_MODE_PULSE, SENSOR_MODE_PERCENT, SENSOR_MODE_CELSIUS, SENSOR_MODE_FAHRENHEIT, SENSOR_MODE_ROTATION
Для типов в SetSensorType()	SENSOR_TYPE_TOUCH, SENSOR_TYPE_TEMPERATURE, SENSOR_TYPE_LIGHT, SENSOR_TYPE_ROTATION
Для выходов в On(), Off() и т.д..	OUT_A, OUT_B, OUT_C
Для режимов в SetOutput()	OUT_ON, OUT_OFF, OUT_FLOAT
Для направления в SetDirection()	OUT_FWD, OUT_REV, OUT_TOGGLE
Для выходной мощности в SetPower()	OUT_LOW, OUT_HALF, OUT_FULL
Для звуков в PlaySound()	SOUND_CLICK, SOUND_DOUBLE_BEEP, SOUND_DOWN, SOUND_UP, SOUND_LOW_BEEP, SOUND_FAST_UP
Для режимов в SelectDisplay()	DISPLAY_WATCH, DISPLAY_SENSOR_1, DISPLAY_SENSOR_2, DISPLAY_SENSOR_3, DISPLAY_OUT_A, DISPLAY_OUT_B, DISPLAY_OUT_C
Уровень мощности передатч. в SetTxPower()	TX_POWER_LO, TX_POWER_HI

Ключевые слова

Ключевые слова - это слова, зарезервированные компилятором NQC для собственно языка программирования. Ошибкой является использование любого из них как имени функции, задачи или переменной. Существуют следующие ключевые слова: `__sensor`, `abs`, `asm`, `break`, `const`, `continue`, `do`, `else`, `false`, `if`, `inline`, `int`, `repeat`, `return`, `sign`, `start`, `stop`, `sub`, `task`, `true`, `void`, `while`.

Задачи, подпрограммы, встроенные функции и макросы

Программы NQC могут иметь многозадачность. Кроме того, части кода возможно поместить в подпрограммах, чтобы затем использовать в различных местах программы. Использование задач и подпрограмм делает программы более понятными и компактными.

Задачи

Программа NQC состоит максимум из 10 задач. Каждая задача имеет имя. Одна задача должна иметь имя `main`, и эта задача будет выполнена. Другие задачи будут выполнены только, когда работающие задачи выдают команды для их исполнения, используя команду `start`. С этого момента обе задачи работают одновременно (т.е. и первая задача продолжает работать). Одна работающая задача может также остановить другую работающую задачу при использовании команды `stop`. Позже эта задача может снова быть запущена, но она будет стартовать с начала, а не с места, где была остановлена.

Подпрограммы

Иногда во многих местах программы необходима одинаковая часть кода. В этом случае можно поместить эту часть кода в подпрограмме и дать ей имя. Теперь можно выполнить эту часть кода, просто вызывая ее имя изнутри задачи. NQC (а фактически - контроллер RCX) учитывает максимум 8 подпрограмм.

```
sub turn_around()
{
    ...
}
task main()
{
    ...
    turn_around();
    ...
    turn_around();
    ...
    turn_around();
    ...
}
```

Подпрограмма вызывается путем записи ее имени с последующими круглыми скобками. Таким образом, этот вызов выглядит одинаково со многими из команд. Однако никаких параметров здесь не записывают, т.е. между круглыми скобками нет ничего.

Следует сделать некоторые предупреждения. Дело в том, что в NQC подпрограммы являются несколько необычными. Например, их нельзя вызвать из других подпрограмм. Подпрограммы можно вызывать из различных задач, но это не поощряется, т.к. возникает проблема возможного запуска одной и той же подпрограммы фактически дважды одновременно различными задачами. Все это может приводить к нежелательным эффектам. Кроме того, вызывая подпрограмму из различных задач, из-за ограничений в программируемом оборудовании RCX, становится невозможным использовать усложненные выражения. Итак, если точно не известно, что она делает - ***не вызывайте подпрограмму из различных задач!***

Встроенные функции

Как показано выше, подпрограммы создают определенные проблемы. Их хорошая черта - то, что они сохраняются в памяти RCX только один раз. Это экономит память и, поскольку RCX не имеет достаточно много доступной памяти, это полезно. Но когда подпрограммы короткие, лучше использовать вместо них встроенные функции. Они не сохраняются отдельно, а копируются в каждом месте, где используются. Это занимает больше памяти, но проблемы подобные использованию сложных выражений, больше не возникают. Здесь также и нет никакого ограничения на количество встроенных функций.

Определение и вызов встроенных функций происходит точно так же, что и в подпрограммах. Только используется ключевое слово **void**, а не **sub** (слово **void** используется, потому что такое же слово имеется в других языках подобных C). Так вышеупомянутый пример при использовании встроенных функций, выглядит следующим образом:

```
void turn_around()
{
    ...
}
task main()
{
    ...
    turn_around();
    ...
    turn_around();
    ...
    turn_around();
    ...
}
```

Встроенные функции имеют преимущество перед подпрограммами. Они могут иметь аргументы. Аргументы могут использоваться, чтобы передать значение определенных переменных во встроенную функцию.

Заметим, что в круглых скобках позади имени встроенной функции определяются аргумент(ы) функции. Когда есть несколько аргументов, их необходимо отделить запятыми.

Определение макроса

Есть еще один режим дать маленьким частям кода имя. Можно определить макрос в NQC (не путать с макросом в RCX Command Center). Ранее мы видели, что можем определить константы, используя `#define` и задавая им имя. Но фактически таким же образом можно определить любую часть кода. Запишем снова ту же программу, но теперь с использованием макроса для разворотов.

```
#define turn_around OnRev(OUT_C);Wait(340);OnFwd(OUT_A+OUT_C);
task main()
{
      ...
      turn_around;
      ...
      turn_around;
      ...
      turn_around;
      ...
}
```

После оператора `#define` слово `turn_around` поддерживает последующий текст. Теперь везде, где в программе написано `turn_around`, оно заменяется кодом этого текста. Заметим, что текст должен быть в одной строке. (Существуют режимы установить оператор `#define` для многих строк, но это не рекомендуется).

Операторы `#define` фактически намного более мощны. Они могут также иметь аргументы. Например, для движения вперед имеется два аргумента: скорость и время.

```
#define forwards(s,t)SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
task main()
{
      forwards(3,200);
}
```

Очень полезно определить такой макрос. Он делает код более компактным и удобочитаемым. Кроме того, можно проще изменить код, когда, например, изменяются связи с двигателями.

Встроенные звуки

Имеется шесть встроенных в RCX звуков, пронумерованных от 0 до 5. Они звучат следующим образом:

0 - Звук клавиши

1 - Бип-бип

2 - Звук убывающей частоты
3 - Звук нарастающей частоты
4 - Звук ошибки 'Бахх'

5 - Звук быстро нарастающей частоты

Можно воспроизводить их используюя команды PlaySound (). Вот - маленькая программа, которая делает их.

```
task main()
{
PlaySound(0); Wait(100);
PlaySound(1); Wait(100);
PlaySound(2); Wait(100);
PlaySound(3); Wait(100);
PlaySound(4); Wait(100);
PlaySound(5); Wait(100);
}
```

Может возникнуть вопрос, для чего нужны команды ожидания. Причина в том, что команда, которая играет звук, не ожидает его окончания. Она немедленно выполняет следующую команду. RCX имеет небольшой буфер, в котором он может хранить некоторые звуки, но через некоторое время этот буфер становится полным, и звуки теряются. Это не настолько серьезно для звуков, но очень важно для музыки, как мы увидим ниже.

Отметим, что аргументом PlaySound () должна быть константа. Здесь не возможно поместить переменную!

Для более интересной музыки NQC имеет команду PlayTone (). Она имеет два аргумента. Первый - частота, и второй - продолжительность (в тиках по 1/100 секунды, как в команде ожидания). Вот - таблица полезных частот:

Звук	1	2	3	4	5	6	7	8
G#	52	104	208	415	831	1661	3322	
G	49	98	196	392	784	1568	3136	
F#	46	92	185	370	740	1480	2960	
F	44	87	175	349	698	1397	2794	
E	41	82	165	330	659	1319	2637	
D#	39	78	156	311	622	1245	2489	
D	37	73	147	294	587	1175	2349	
C#	35	69	139	277	554	1109	2217	
C	33	65	131	262	523	1047	2093	4186
B	31	62	123	247	494	988	1976	3951
A#	29	58	117	233	466	932	1865	3729
A	28	55	110	220	440	880	1760	3520

Как мы отметили выше для звуков, здесь RCX также не ждет завершения ноты. Так, если их используется много подряд лучше добавлять (немного дольше), команды ожидания между ними. Вот - пример:

```
task main()
{
PlayTone(262,40); Wait(50);
PlayTone(294,40); Wait(50);
PlayTone(330,40); Wait(50);
PlayTone(294,40); Wait(50);
PlayTone(262,160); Wait(200);
}
```

Можно создать музыкальные пьесы очень легко используя Фортепьяно RCX (меню Tools), которое

является частью RCX Command Center .

Если необходимо получить воспроизведение музыки RCX при движении робота, используйте лучше отдельную задачу для этого. Ниже дан пример довольно глупой программы, в которой RCX двигается назад и вперед, постоянно выдавая музыку.

```
task music()
{
while (true)
{
PlayTone(262,40); Wait(50);
PlayTone(294,40); Wait(50);
PlayTone(330,40); Wait(50);
PlayTone(294,40); Wait(50);
}
}
task main()
{
start music;
while (true)
{
OnFwd(OUT_A+OUT_C); Wait(300);
OnRev(OUT_A+OUT_C); Wait(300);
}
}
```

Приложение 2. Перечень деталей

балка 1x2 4x

балка 1x2 4x

балка 1x4 4x

балка 1x6 4x

балка 1x12 2x

балка 1x16 голубая 2x

балка 1x16 черная 2x

пластина 1x2 8x

пластина 1x8 2x

опора скользящая черная 2x2 2x

пластина 2x4 с отверстиями 4x

пластина 2x6 с отверстиями 4x

пластина 2x8 с отверстиями 4x

пластина 2x10 с отверстиями 2x

пластина 6x14 2x

пластина угловая 2x2 2x

минишкив/блок 8x

универсальная втулка 20x

короткий штифт с кнопкой 4x

укороченный штифт 1,5 мм 8x

штифт гладкий 12x

штифт-полуось 8x

черный штифт с выступами 12x

фиксатор 2x

захват с одним промежуточным отверстием 2x

захват 4x

втулка-удлинитель оси 2x

кирпич 2x2 желтый 6x

кирпич 2x4 красный 6x

угловая балка со скруглением 2x

8-зубое зубчатое колесо 3x

24-зубое зубчатое колесо 2x

40-зубое зубчатое колесо 2x

ось 2-кнопочная 4x

ось 3-кнопочная 2x

ось 4-кнопочная 2x

ось 5-кнопочная 2x

ось 6-кнопочная 2x

ось 8-кнопочная 2x

ось 10-кнопочная 2x

ось 12-кнопочная 2x

полуось 2x

ремень синий 26 мм 4x

полусфера с зеркальной поверхностью 2x

черный парик 1x

коричневый парик 1x

фигурка человека 1x

витая трубка 2x

провод 12,8 см с соединительными пластинами 2x2 3x

провод 52 см с соединительными пластинами 2x2 3x

колесный диск 4x

средний шкив 2x

колесный диск большой 2x

гладкая шина 2x

малая шина с протектором 2x

средняя шина с протектором 2x

шина 2x

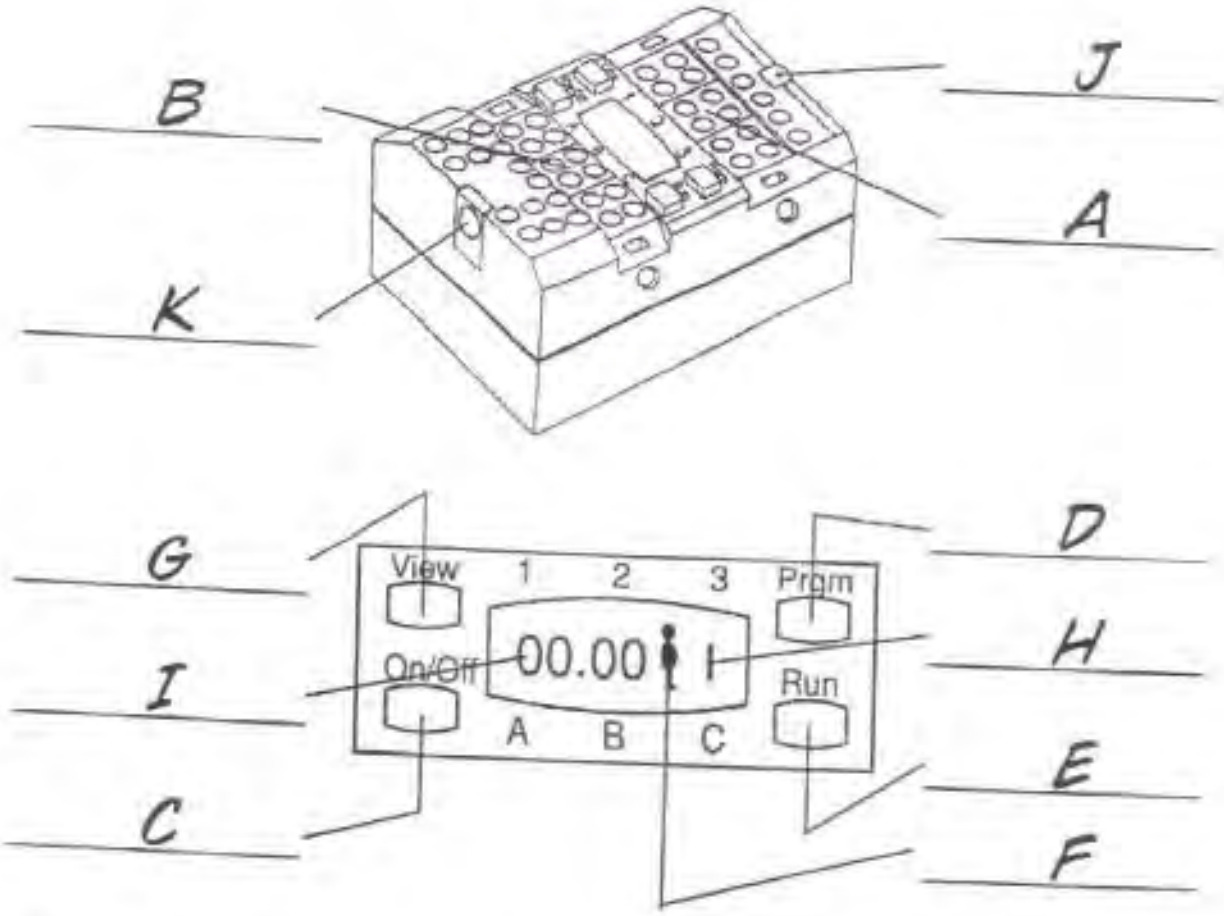
лампа 1x2 1x

Датчик Освещенности 1x

Датчик Касания 2x

большой мотор 9 В 2x

для соединения с USB-портом к набору 9786 1x



A. Порты ввода — к ним подключают датчики касания и освещенности

B. Порты вывода — к ним подключают моторы и лампочки

C. Кнопка включения/выключения блока RCX

D. Кнопка выбора программного отсека, в который будет загружена программа

E. Кнопка начала/остановки выполнения программы

F. Индикатор выполнения программы (показывает, выполняется ли программа в текущий момент)

G. Кнопка просмотра текущего значения показаний датчика, подключенного к порту

H. Индикатор номера текущего программного отсека

I. Индикатор загрузки программного обеспечения

J. Приемник/передатчик инфракрасных сигналов

K. Гнездо сетевого адаптера

Рисунок 1. Устройство RCX