Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

**Л. Е. Стрюк, О. А. Дахин, А. А. Шалыто**

# Решение задачи движения робота по линии с применением автоматного подхода (проект *RoboChuck*)

Проектная документация

Санкт-Петербург
2008

## Оглавление

Введение 3
1. Постановка задачи 4
2. Особенности робота исполнителя 4
2.1. Структура робота 4
2.1.1. Датчики 5
2.1.2. Шасси 5
2.1.3. Модуль движения 6
2.1.4. Модуль управления 6
2.2. Общее описание электронных компонентов робота 6
3. Алгоритм движения 7
4. Автоматы, управляющие роботом 7
4.1. Описание управляющей системы 7
4.2. Автомат A0 8
4.3. Автомат A1 8
5. Описание пользовательского интерфейса 10
5.1. Управляемый режим 10
5.2. Автономный режим 10
5.3. Обмен сообщениями между пользовательским интерфейсом и модулем управления 11
5.4. Протокол общения модуля управления с модулем движения 11
Экспериментальное исследование 12
Заключение 12
Источники 13

# Введение

Задача следования маршруту является одной из наиболее распространенных задач в робототехнике. Ее распространенность связана с ее простотой и с огромным числом способов реализовать ее решение. В данном проекте маршрут представляет собой кривую, напечатанную лазерным принтером на белой бумаге (рис. 1).

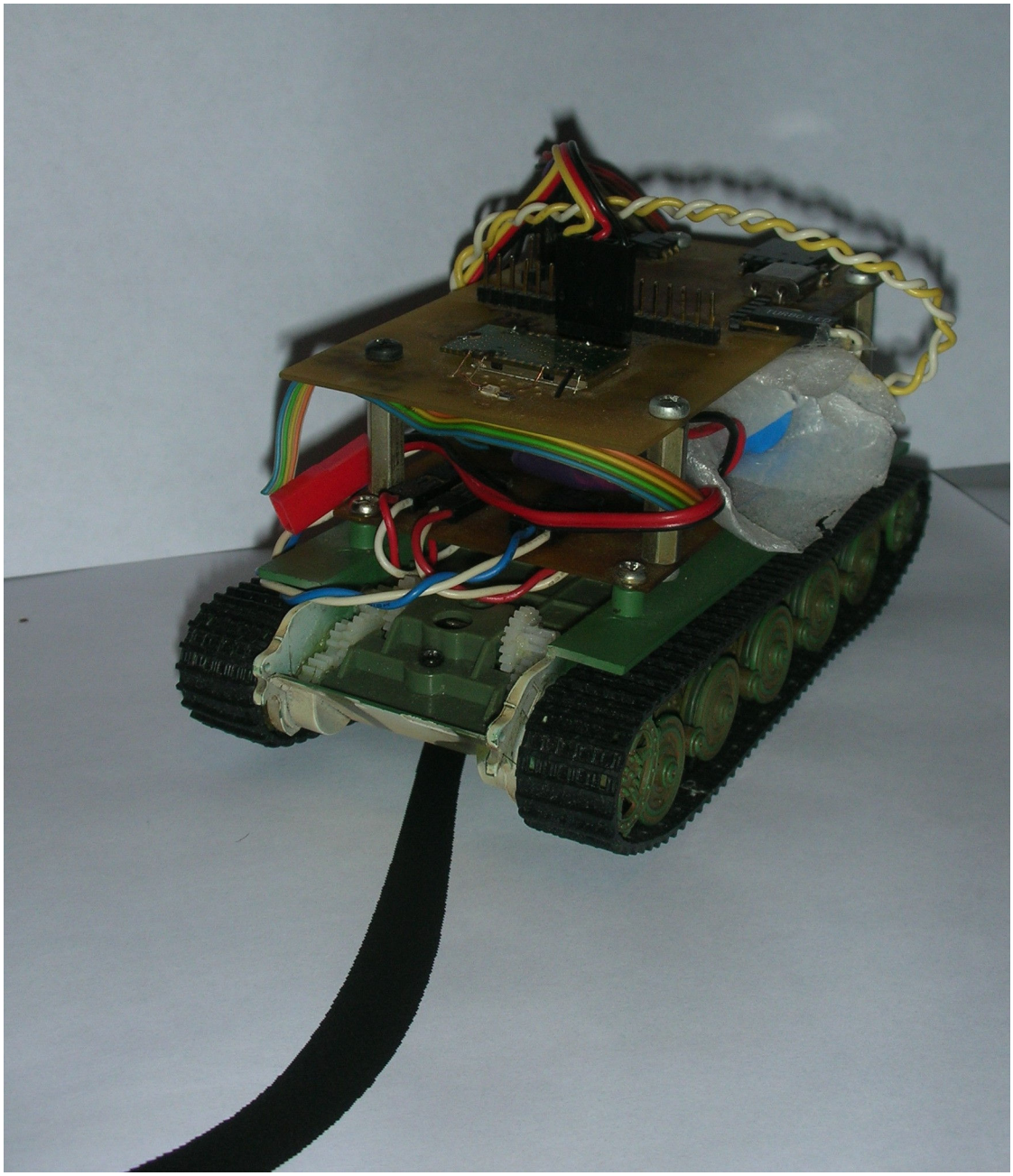

**Рис. 1. Движение по линии**

# 1. Постановка задачи

Создать систему, обеспечивающую следование робота по линии, с применением автоматного подхода.

Система состоит из двух частей: робота-исполнителя и пользовательского интерфейса для персонального компьютера (ПК).

Робот-исполнитель и интерфейс общаются между собой с помощью *Bluetooth*.

Робот-исполнитель работает в двух режимах: автономное движение по линии и движение, контролируемое пользователем ПК.

Когда пользователь переводит робота в автономный режим, интерфейс визуализирует состояния автомата, управляющего роботом. Соответственно робот отправляет на ПК данные о своем состоянии.

В любом режиме движения интерфейс должен отображать состояния датчиков, получая их с робота-исполнителя.

# 2. Особенности робота исполнителя

## 2.1. Структура робота

На рис. 2 представлен робот *«Chucky»*.

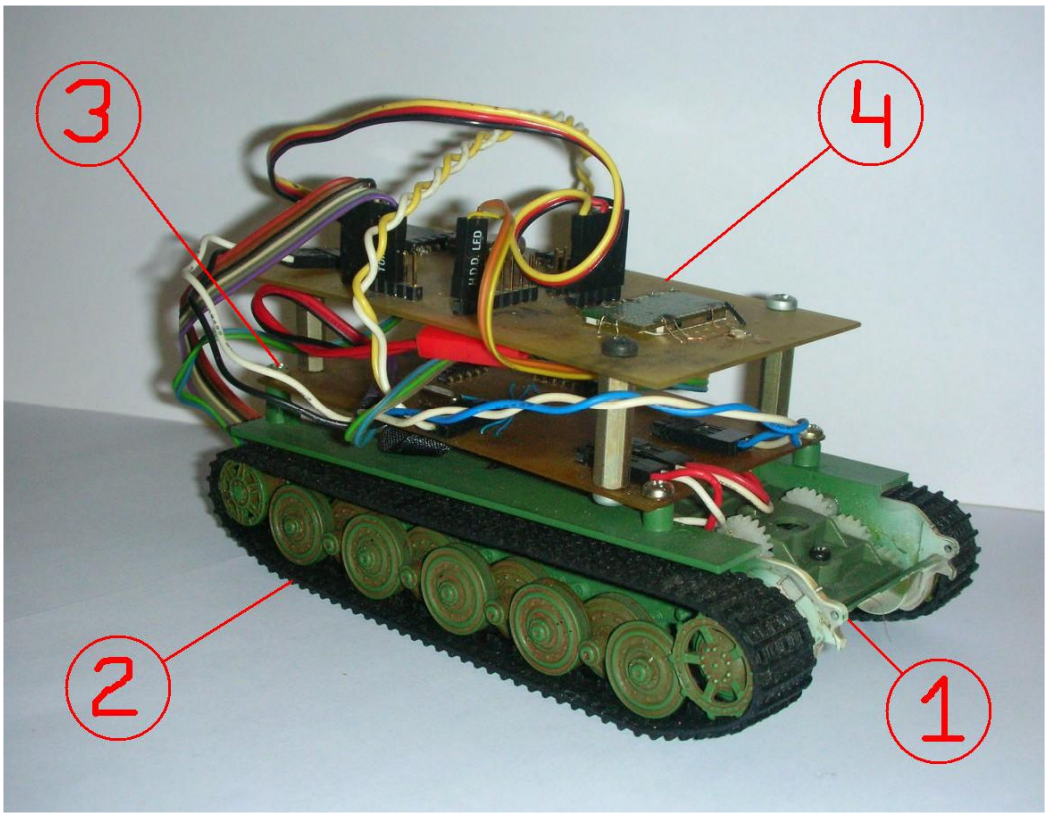


**Рис. 2. Подробная фотография робота**

На этом рисунке:

1. Датчики.
2. Шасси.
3. Модуль движения.
4. Модуль управления.

### 2.1.1. Датчики

В качестве датчиков применяются три инфракрасных (ИК) фоторезистора со встроенными светодиодами. В предыдущей версии датчика были использованы шесть фоторезисторов и шесть ИК светодиодов, но эта версия оказалась неработоспособной, так как светодиоды «засвечивали» фотодатчик и в результате на выходе всегда был сигнал, соответствующий яркому свету.

Фоторезисторы подключены на аналогово-цифровой преобразователь (АЦП) контроллера модуля управления. Плата с фоторезисторами расположена под днищем робота (рис.3).

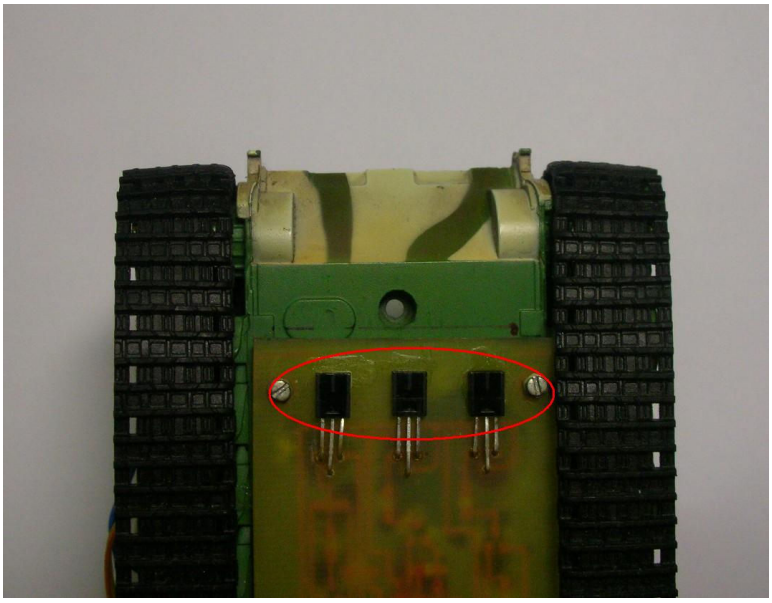

**Рис. 3. Фотодатчики**

За счет того, что освещенность измеряется в ИК-диапазоне, существенным является то, что линия должна быть **напечатана именно лазерным принтером**, так как линия, нарисованная обычным черным маркером или тем же струйным принтером, для ИК-излучения прозрачна.

Линия обнаруживается за счет того, что свет от белой бумаги свет отражается значительно лучше, чем от черной линии. Измерения происходят следующим образом: выбирается средняя из 16 измерений разности между освещенностью при включенном и выключенном светодиоде. Если эта разность меньше порогового значения (которое пользователь может задать), то считается, что датчик находится над черной линией.

Одно измерение освещенности происходит приблизительно за 60–100 микросекунд. На получение данных со всех трех датчиков уйдет не более чем 100*16*3 = 4.8 миллисекунды. Это позволяет достаточно часто опрашивать датчики.

### 2.1.2. Шасси

Подвижная часть представляет собой гусеничное шасси от модели танка в масштабе 1:48. Каждая гусеница подключена к своему двигателю. Таким образом, задавая различные скорости вращения этих двигателей, реализуется движение робота в различных направлениях.

В автономном режиме осуществляется движение вперед, назад, а также выполняются повороты на месте (по и против часовой стрелки).

Выбранное шасси не может гарантировать соответствие между ожидаемым и реальным направлением движения из-за ненадежной передачи вращательного момента от мотора к гусенице. Данная особенность была учтена при разработке алгоритма управления движения по линии – алгоритм устраняет неустойчивость движения (разд. 3).

### 2.1.3. Модуль движения

Этот модуль состоит из процессора *ATtiny2313* и драйвера двигателей.

Из ресурсов процессора использованы следующие:

- восьмибитный таймер;
- порт *UART*;
- порт ввода-вывода.

Таймер использован для создания широтно-импульсная модуляция (ШИМ) сигналов для каждого из двигателей.

Интерфейс *UART* использован для общения с модулем управления.

Драйвер двигателей представляет собой микросхему, в которую интегрированы два моста (восемь транзисторов). Контроллер подключен к этой микросхеме с помощью порта ввода-вывода. При этом управление каждым двигателем состоит из двух частей:

- логический «0» или «1» для задания направления работы двигателя;
- ШИМ–сигнал для задания скорости вращения двигателя.

Также на этой плате содержится драйвер для общения с ПК по интерфейсу *RS232*. Драйвер использовался для вывода отладочной информации. Подключить контроллер напрямую к *COM*-порту ПК нельзя, так как на компьютере логическому нулю и единице соответствуют напряжения, от которых контроллер может элементарно сгореть. Эта микросхема обеспечивает увеличение напряжения сигнала, который приходит от контроллера к ПК, и, соответственно, уменьшение напряжения, идущего от ПК к контроллеру.

### 2.1.4. Модуль управления

Этот модуль состоит из контроллера *Atmega128L* и *iWrap*.

*Atmega128L* является самым мощным восьмибитным микроконтроллером семейства *AVR*.

Из многочисленных ресурсов контроллера *Atmega128L* использованы:

- три канала АЦП;
- порт ввода-вывода;
- два *UART* порта.

В этом модуле выполняется управление модулем движения, обработка команд, полученных от пользователя и обработка данных с датчиков.

Для связи робота с ПК используется микросхема *iWrap* компании *BLUEGIGA*. Она реализует протокол *Bluetooth*, используемый для общения робота с ПК. К ее достоинствам можно отнести подробную документацию, удобный корпус и то, что данные, полученные с радиоканала, микросхема выдает по протоколу *RS232*, который реализован в контроллерах, использованных для этого проекта.

## 2.2. Общее описание электронных компонентов робота

Для управления роботом используются микроконтроллеры *Atmel* семейства *AVR*. Они были выбраны по следующим причинам:

1. Даже самый дешевый контроллер может быть перепрошит до 1000 раз, а это очень важно в условии написания экспериментальной программы, которая очень часто изменяется.

2. Эти микроконтроллеры оптимизированы под язык *Cu*, что улучшает читаемость кода и избавляет от необходимости изучать их ассемблер. Это делает разницу в производительности программ, написанных на *Cu* и ассемблере, минимальной.

3. У любого процессора этого семейства достаточно богатая периферия. Это позволяет сильно упростить электронную схему, так как не требуются лишние детали. Это, в свою очередь, ведет к уменьшению времени, необходимого для расстановки деталей и трассировки печатных плат.

4. Компания *Atmel* выпускает очень удобную среду разработки: *CodeVision AVR*, обладающая следующими преимуществами перед другими средами:

- В среду встроен компилятор, редактор кода, и имеется терминал. Это дает возможность редактировать, компилировать, «прошивать» и отлаживать приложения.
- В среде содержится менеджер проектов, позволяющий сэкономить время на чтения документации контроллеров.

5. Контроллеры этого семейства можно программировать без программатора и источника питания, не вынимая из платы, а просто подключив контроллер к специальному устройству.

# 3. Алгоритм движения

Для решения поставленной задачи движения по линии был разработан соответствующий алгоритм. Он довольно простой, но без знания его проблематично понять принцип работы автомата *A1*. При нахождении на линии робот видит ее центральным датчиком. При потере контакта с линией центрального датчика, робот поворачивается в сторону бокового датчика, который последним видел линию. Боковой датчик успеет увидеть линию при потере, из-за маленького расстояния между датчиками (или отклонение будет не существенным).

Данный алгоритм позволяет нивелировать неточности, вызванные шасси, возвращая робот на линию. Например, если робот ехал прямо по линии, и левую гусеницу заклинило. Из-за различного вращения гусениц робот начнет поворачиваться влево, что приведет к потере линии. Центральный датчик потеряет линию. При этом последним ее увидит правый боковой датчик. Произойдет поворот робота в другую сторону, что вернет его на линию.

Данный алгоритм позволяет ездить по незамкнутой линии (при достижении конца – разворот). При реализации алгоритма был применен автоматный подход (разд. 4). Алгоритм реализован на языке *Cu*.

# 4. Автоматы, управляющие роботом

## 4.1. Описание управляющей системы

Система управления роботом состоит из двух автоматов *A0* и *A1*. *A0* – автомат, выполняющий общение с ПК. *A1* – автомат, управляющий автономным движением. Конфигурация движения управляется автоматом *A0*. При переходе в состояние «Автономное движение» запускается автомат *A1*. При переходе из этого состояния автомат *A1* прекращает свою работу. Таким образом, автомат *A1* является вложенным для одного из состояний автомата *A0*.

Входные переменные и выходные воздействия записываются в виде *x_n.*. или *z_n.*., где вместо символа *n* стоит номер автомата.

Иногда на ребре перехода в графе неудобно отображать все значения всех переменных, поэтому у переменных есть приоритеты. Приоритет переменной суть номер переменной: чем меньше номер, тем больше приоритет.

Входные и выходные воздействия выполняются при переходе от одного состояния к другому, причем в следующем порядке:

1. Выходное воздействие состояния, которое покинули.
2. Воздействие, происходящее на ребре перехода.
3. Входное воздействие нового состояния.

## 4.2. Автомат *A0*

**Краткое описание**

Этот автомат (рис. 4) обеспечивает переключение управления робота из контролируемого режима в автономный и наоборот. При этом в обоих режимах он позволяет получить информацию о датчиках и других параметрах.

**Перечень состояний автомата *A0***

*0* — управляемое движение.
*1* — автономное движение.

**Перечень входных переменных автомата *A0***

*x0_0* — пришло сообщение с ПК.
*x0_2* — с ПК пришла команда на смену состояния.

**Перечень выходных воздействий автомата *A0***

*z0_1* — остановить двигатели.
*z0_2* — выполнить команду, пришедшую с ПК, доступную в состоянии *0*.
*z0_3* — произвести действия для смены состояния автомата.
*z0_4* — выполнить команду, пришедшую с ПК, доступную в состоянии *1*.

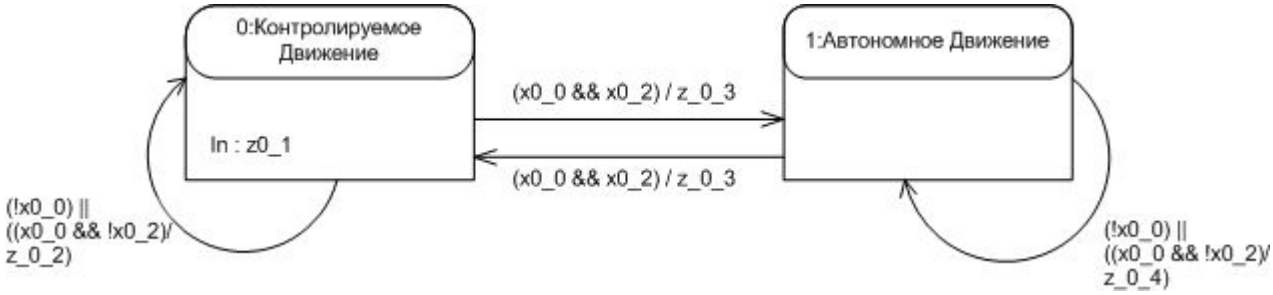


**Рис. 4. Граф перехода автомата *A0***

## 4.3. Автомат *A1*

**Краткое описание**

Этот автомат (рис. 5) выполняет управление роботом в автономном режиме. При старте робот движется вперед до срабатывания какого-нибудь из датчиков. После нахождения линии двигается по ней, при потере старается вернуться на нее.

**Перечень состояний автомата *A1***

*0* — начальный поиск.
*1* — центр.
*2* — левый центр.

*3* — правый центр.
*4* — левый.
*5* — пусто.
*6* — правый.

**Перечень входных переменных автомата *A1***
*x1_0* — ни один датчик не сработал.
*x1_1* — сработал левый датчик.
*x1_2* — сработал левый и центральный датчики.
*x1_3* — сработал центральный датчик.
*x1_4* — сработал правый датчик.
*x1_5* — сработал правый и центральный датчики.
*x1_6* — сработали все датчики.

**Перечень выходных воздействий автомата *A1***
*z1_1* — отправить команду на модуль движения «ехать вперед».
*z1_2* — отправить команду на модуль движения «ехать влево».
*z1_3* — отправить команду на модуль движения «ехать вправо».
*z1_4* — установить последний сработавший датчик как левый.
*z1_5* — установить последний сработавший датчик как правый.
*z1_6* — отправить на модуль движения команду поворота в направлении последнего сработавшего датчика.

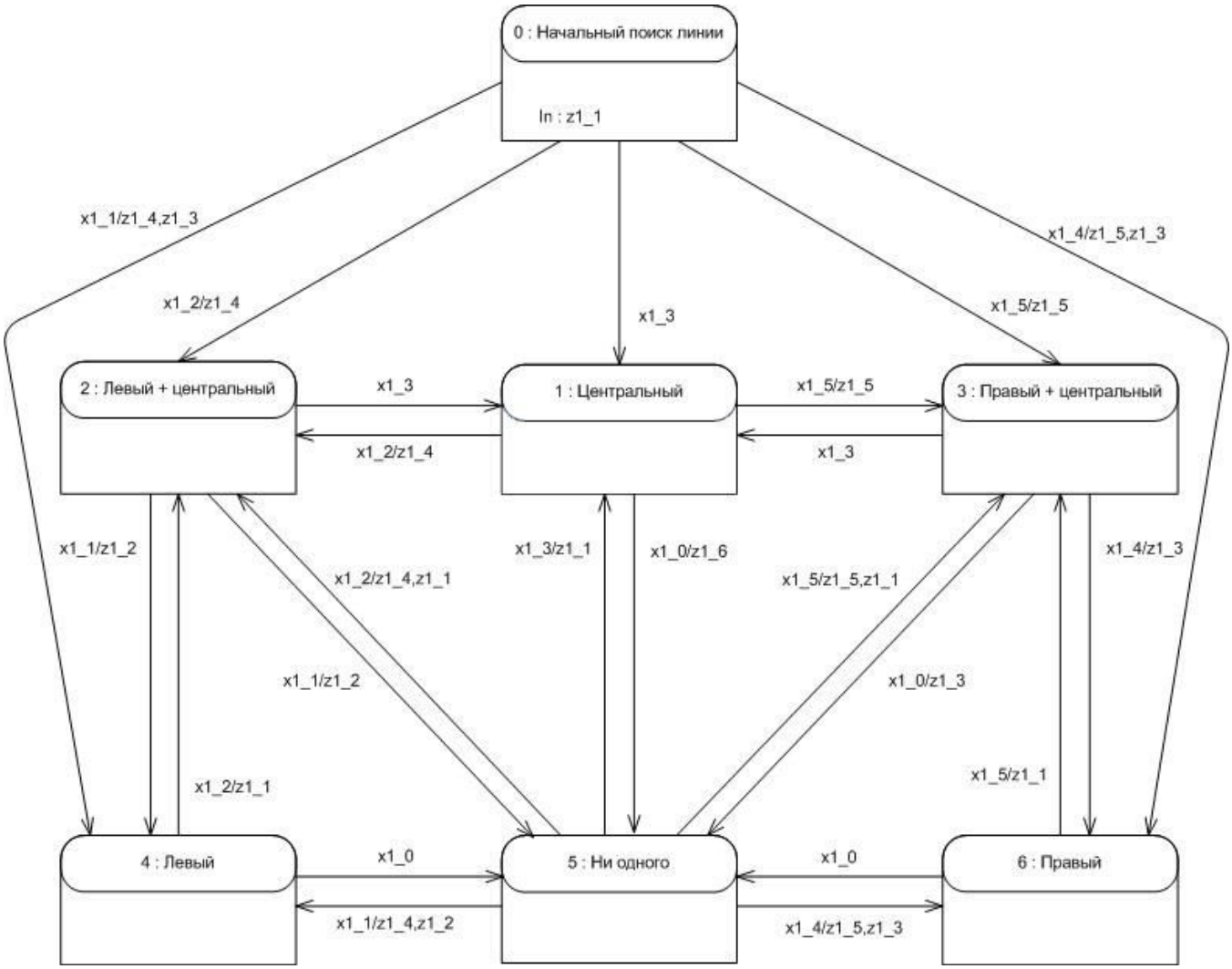


**Рис. 5. Граф переходов автомата *A1***

# 5. Описание пользовательского интерфейса

Пользовательский интерфейс представляет собой графическое *Java*-приложение. Окно программы представляет собой панель с двумя страницами, каждая из которых отвечает за один из режимов работы робота (управляемый и автономный). Переключение между страницами переводит робота в соответствующее состояние.

## 5.1. Управляемый режим

В этом режиме (рис. 6) пользователь имеет возможность задавать направление движения робота с определенной скоростью. Управление осуществляется как с помощью мыши, так и нажатием соответствующей клавиши для направления движения: клавиши *«S»* для останова, клавиши *«Q»* и *«A»* для увеличения и уменьшения скорости движения соответственно.

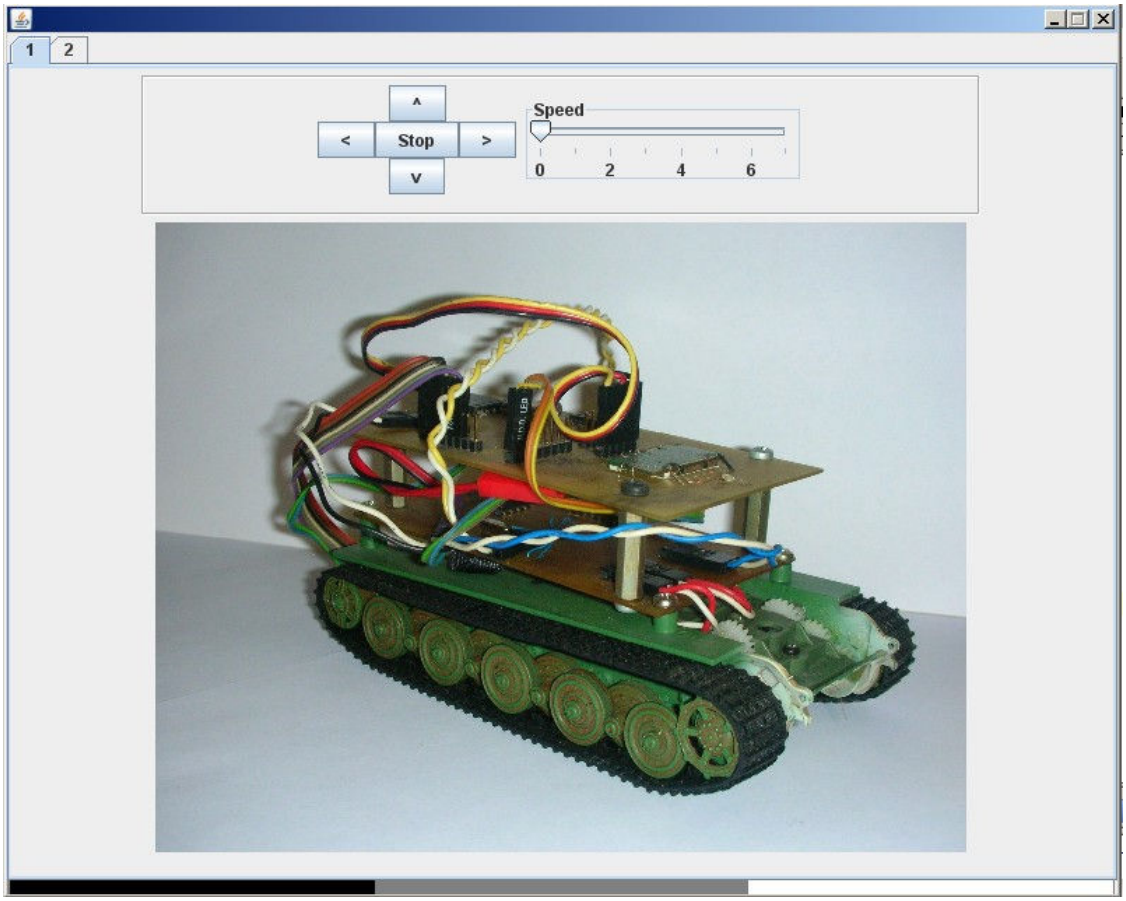


**Рис. 6. Пользовательский интерфейс. Управляемый режим**

## 5.2. Автономный режим

В автономном режиме (рис. 7) пользователь имеет возможность следить за изменением состояний автомата движения. Текущее состояние автоматически подсвечивается синим цветом. В нижней части панели в обоих режимах отображается состояние датчиков (черный цвет – датчик «видит» линию, белый – просто поле). Для работы при различной освещенности и на различных полях движения имеет смысл использовать разные уровни порога яркости для фотоэлементов. Однако экспериментально было установлено, что для черной линии, напечатанной лазерным принтером на белой бумаге, наилучшая разрешающая способность достигается при значении уровня 40, который и установлен в программе.

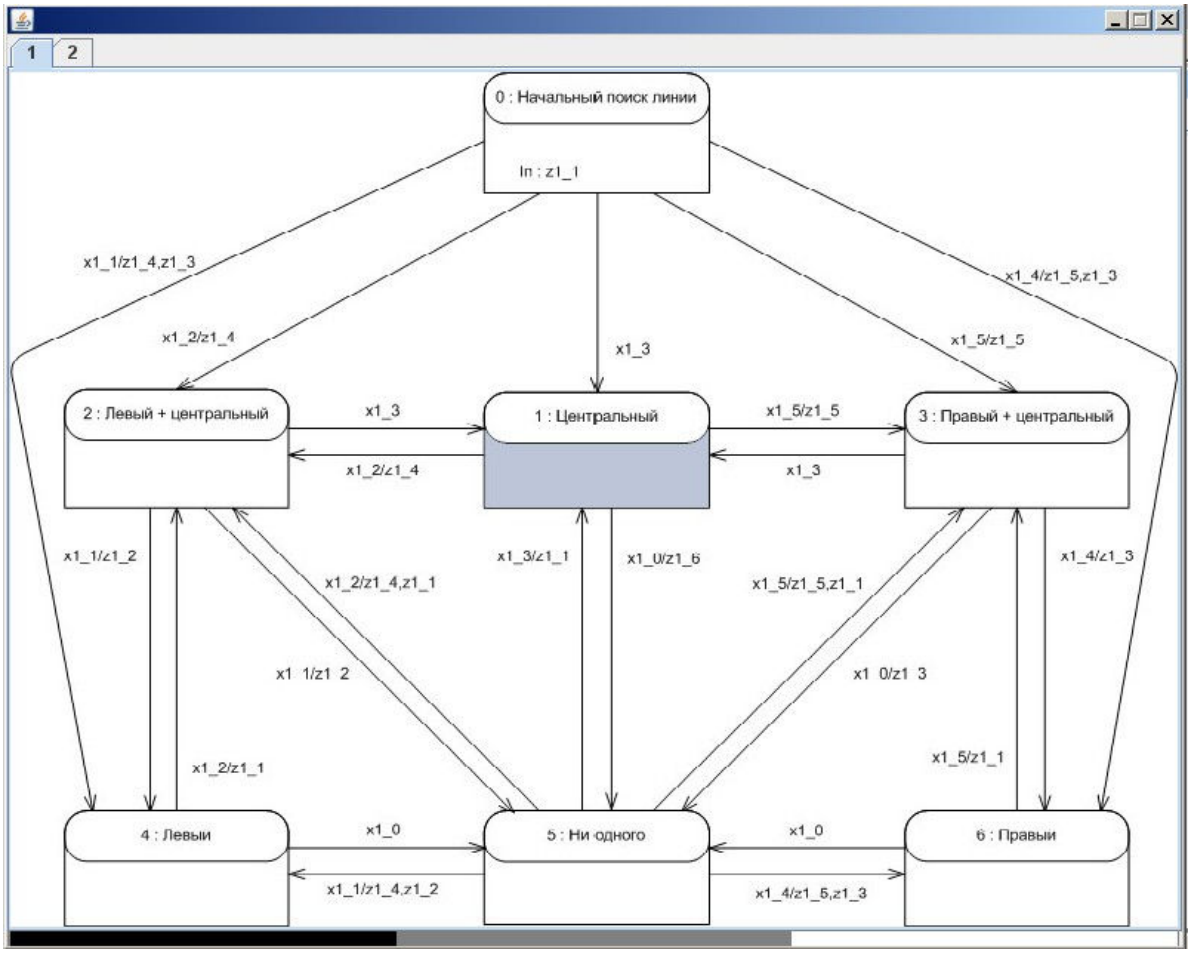


**Рис. 7. Пользовательский интерфейс. Автономный режим**

## 5.3. Обмен сообщениями между пользовательским интерфейсом и модулем управления

*Bluetooth*-адаптер, с точки зрения операционной системы, является *COM*-портом. Для общения с этим портом была использована библиотека *RXTX*, которая была взята с сайта *rxtx.org*. Она позволяет посылать и отправлять команды на модуль управления в обыкновенном потоке ввода-вывода.

## 5.4. Протокол общения модуля управления с модулем движения

Взаимодействие робота и пользовательского интерфейса осуществляется с помощью команд. Как отмечено выше, модули общаются по протоколу *RS232*. Скорость общения достаточно большая – 115.2 кб/с. Это означает, что одна двухбайтная команда будет передаваться приблизительно за *280* микросекунд.

**Однобайтные команды:**
1. ***'s', 'S'*** – остановиться.

**Двухбайтные команды:**
1. ***'W', 'w'*** – задать уровень интенсивности фоторезисторов (второй байт содержит это значение).

2. ***'Q', 'q'*** – задать режим работы робота (второй байт равен *'1'* для автономного режима, *'0'* – для управляемого)
3. ***'F', 'f'*** – движение вперед (во втором байте содержится число от 0 до 7, которое задает скорость).
4. ***'B', 'b'*** – движение назад (во втором байте содержится число от 0 до 7, которое задает скорость).
5. ***'R', 'r'*** – вращение вправо (во втором байте содержится число от 0 до 7, которое задает скорость).
6. ***'L', 'l'*** – вращение влево (во втором байте содержится число от 0 до 7, которое задает скорость).
7. ***'1'*** – задать скорость и направление вращение первой гусеницы, которые содержатся во втором байте: три младших бита – скорость, старший бит – направление.
8. ***'2'*** – задать скорость и направление вращение второй гусеницы, которые содержатся во втором байте: три младших бита – скорость, старший бит – направление.

**Команды, требующие ответа:**
9. ***'D', 'd'*** – запрос текущего состояния датчиков. В ответ на эту команду робот возвращает шесть байт, описывающие значения датчиков в этот момент. Для датчиков справа налево передается двухбайтное значение их уровня освещенности.
10. ***'C', 'c'*** – запрос номера текущего состояния автомата управления (для отображения в пользовательском интерфейсе). В ответ приходит один байт – искомый номер.

## Экспериментальное исследование

При разработке системы управления ключевыми моментами являлись устройство робота и алгоритм его движения. Именно сочетание этих двух механизмов и определяет работоспособность всего комплекса.
Изначально предполагалось оснастить робот сенсором оптической мышки, который и будет следить за линией. Из сигнала, полученного от сенсора, можно извлечь большое количество информации о движении робота и относительного положения линии. Однако получать и обрабатывать такой сигнал сложнее. Однако эта идея не была реализована, так как для работы сенсора необходимо точно выдержать его фокусное расстояние, что практически невозможно в условиях данной модели.
Действуя от сложного к простому, далее стали рассматриваться модели, основанные на датчиках-фотоэлементах. Желание получать больше информации, а следовательно, иметь возможность более тонкой настройки управления роботом, привело к различным конфигурациям, использующих от пяти до девяти датчиков. В этих случаях, правда, возникали трудности на этапе разработки алгоритма, вызванные сложностью системы. При использовании автоматного подхода выяснилось, что достаточно только трех датчиков для обеспечения корректного управления.

## Заключение

Спроектированная система управления роботом решает поставленную задачу для многих классов линий, в том числе незамкнутых и самопересекающихся. Логика работы робота описана графами переходов, что упрощает создание и проверку алгоритма движения. Данный проект демонстрирует успешность применения автоматного подхода для управления роботом.

# Источники


1. *Documentation*. http://rxtx.org/
2. *Roboforum.* http://roboforum.ru/
3. *Предко М.* Устройства управления роботами: схемотехника и программирование. М.: ДМК Пресс, 2004.
4. *Евстифеев А. В.* Микроконтроллеры AVR семейства Tiny и Mega фирмы ATMEL. М.: Додэка XXI, 2005 г.
5. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1
6. *Хорн Б. К.* Зрение роботов. М.: Мир, 1989.