УДК 004.42+004.896

ИЗУЧЕНИЕ ПРИНЦИПОВ ПРОГРАММИРОВАНИЯ МОБИЛЬНЫХ РОБОТОВ, РЕАЛИЗОВАННЫХ НА ПЛАТФОРМЕ VEX

Сиваченко Д.С., студ., Руммо.Д.С., студ., Черненко Д.В., ст. преп., Соколова А.С., ст. преп.
Витебский государственный технологический университет, г. Витебск, Республика Беларусь

Реферат. *В статье рассмотрены основные принципы разработки программного обеспечения для мобильных роботов серии VEX Robotics.*

Ключевые слова: программирование робота, VEX IQ gen 1, VEXcode IQ, C++.

Роботы VEX подразделяются на серии, такие как: 123, GO, IQ, EXP, V5, PRO. Каждая следующая серия является усовершенствованной версией предыдущей, с добавлением нового функционала и расширением уже имеющегося. Исследуемый в данной работе робот относится к первому поколению серии IQ.

Основой задачей разработки стало написание программы для объезда препятствий, независимо от их размера и формы. Программа должна была иметь концепцию, схожую с принципом работы робота-пылесоса.

Первым этапом разработки стало создание прототипа. Для базового функционала понадобились следующие комплектующие: 3 датчика расстояния, 3 датчика касания, 2 двигателя и микроконтроллер Robot Brain. Итоговая модель имеет почти круглую форму, два управляемых колеса и одно для поддержания устойчивости робота.

Второй этап – этап разработки программного обеспечения.

Программирование VEX IQ gen 1 осуществляется на языке C++. Для создания программ на данном языке имеется удобная среда VEXcode IQ.

Для простоты работы с оборудованием робота имеется готовая библиотека, основными классами которой являются:

- Brain – используется, для добавления звуковых эффектов, работы с выводом на экран какой-либо информации. Его наследниками являются классы Screen, Timer и др.;
- Bumper – используется для программирования робота с бамперами;
- Controller – используется для получения значений с удаленного контроллера, а также для вывода на экран контроллера;
- Drivetrain – используется для конфигурации параметров движения робота;
- Motor – используется для управления моторными устройствами;
- Thread – используется для создания потоков и управления ими;
- Vision – используется для программирования датчиков технического зрения.

В среде разработки VEXcode IQ данные классы и их методы разнесены по определенным категориям, например Looks, в которой собраны все методы классов для работы с изображением на экране, или Events, в которой собраны все методы классов для работы с возможными событиями (нажатие бампера и др).

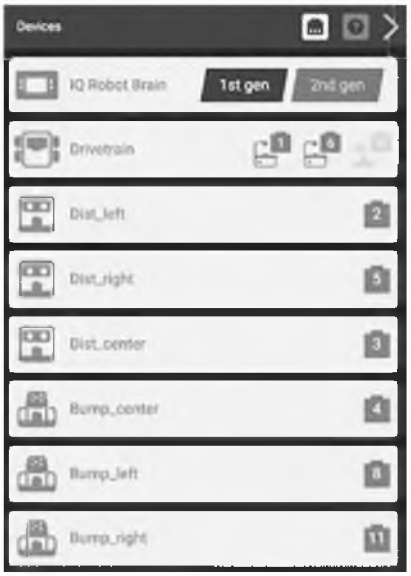
В процессе программирования обнаружены физические ограничения работы модели робота. К примеру, датчик расстояния может считывать расстояния от 50 мм до 1000 мм, но, когда предмет выходит за эти рамки, датчик показывает число 999999. Это связано с устройством самого датчика и необходимо учитывать в процессе написания программы.

Фрагменты полученной программы представлены на рисунках 1–4.

В функции на рисунке 3 0.54 – это коэффициент, позволяющий точно рассчитать угол поворота. Данных коэффициент зависит от базы и радиуса используемых колес.

```
// Robot configuration code.
motor LeftDriveSmart = motor(PORT1, 1, false);
motor RightDriveSmart = motor(PORT6, 1, true);
drivetrain Drivetrain = drivetrain(
  LeftDriveSmart,
  RightDriveSmart,
  100, 150, 0, mm, 1
);
sonar Dist_left = sonar(PORT2);
sonar Dist_right = sonar(PORT5);
sonar Dist_center = sonar(PORT3);
bumper Bump_center = bumper(PORT4);
bumper Bump_left = bumper(PORT8);
bumper Bump_right = bumper(PORT11);
```

а



б

Рисунок 1 - Конфигурация робота: а – код; б – графический вариант

```
void printDist() {
  Brain.Screen.clearLine(1);
  Brain.Screen.clearLine(2);
  Brain.Screen.clearLine(3);

  Brain.Screen.print("dist_left: %f", Dist_left.distance(mm));
  Brain.Screen.newLine();
  Brain.Screen.print("dist_center: %f", Dist_center.distance(mm));
  Brain.Screen.newLine();
  Brain.Screen.print("dist_right: %f", Dist_right.distance(mm));

  Brain.Screen.setCursor(1, 1);
}
```

Рисунок 2 – Функция вывода на экран робота показаний датчиков дистанции

```
void rotateWhereMoreSpace(int deg) {
  if (Dist_left.distance(mm) < Dist_right.distance(mm)) {
      Drivetrain.turnFor(right, deg * 0.54, degrees);
    } else if (Dist_left.distance(mm) > Dist_right.distance(mm)) {
      Drivetrain.turnFor(left, deg * 0.54, degrees);
  }
}
```

Рисунок 3 – Функция для поворота в ту сторону, где расстояние до объекта больше

```
int minRange = 90;  //минимальное расстояние для реагирования
int rangeToSlowDown = 320;  //дистанция, но которой стоит начать замедляться
int minVelocity = 25; //минимальная скорость
int maxVelocity = 55; //максимальная скорость
int curVelocity = 25; //текущая скорость

Drivetrain.setDriveVelocity(minVelocity, percent);
```

Рисунок 4 – Объявление некоторых переменных и установка начальной скорости робота

Действующий по разработанной программе робот способен объехать практически любое препятствие, не выходящее за рамки ограничений его комплектующих.

УДК 004.4

ИНФОРМАЦИОННАЯ СИСТЕМА УПРАВЛЕНИЯ УЧЕБНЫМИ ПЛАНАМИ

Казаков В.Е., к.т.н., доц., Ринейский К.Н., начальник ЦИТ, Демидов Д.Д., выпускник, Карнилов М.С., асс.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В статье представлен обзор разработки клиентской части информационной системы управления учебными планами, представлены предпосылки его внедрения в учреждение образования и круг решаемых им задач.*

Ключевые слова: front-end, учебный план, JavaScript, React.

Учебный план – нормативный документ, регламентирующий общее направление и основное содержание подготовки специалиста, последовательность и интенсивность, сроки изучения учебных дисциплин, основные формы организации обучения, формы и сроки проверки знаний и умений учащихся. В УО «ВГТУ» разработан REST-сервис, обеспечивающий хранение данных учебных планов [1].

Разработана клиентская часть данной информационной системы, которая представляет собой браузерное приложение, не требующее инсталляции. Приложение разработано на языке TypeScript на основе фреймворка React.js [2]. Применялись также: библиотека стилизации компонентов emotion и фреймворк material UI; средство организации централизованного хранилища приложения redux; в качестве средства валидации форм использовалась связка библиотек yup и formic; библиотека доступа к REST-сервисам axios, библиотека управления нотификациями hot-toast, асинхронность обработки запросов реализована с помощью библиотеки thunk, а также средство автоматизации создания селекторов reselect.

Особое внимание было уделено выбору библиотеки реализации табличного визуального компонента. Создание таблиц является одной из наиболее распространенных задач веб-разработки. Вместе с тем, создание качественной таблицы, которая будет удовлетворять всем требованиям пользователей, может быть сложной задачей.

Были исследованы три популярные библиотеки для создания таблиц на базе React: AG Grid, DataGrid и Material React Table. Все они предоставляют множество функций для работы с данными, таких как сортировка, фильтрация, группировка и агрегация. Они также поддерживают виртуализацию и могут работать с большими объемами данных. AG Grid и DataGrid имеют платные версии, которые предоставляют дополнительные функции, такие как экспорт данных, поддержка графиков и диаграмм, а также поддержка различных форматов данных. В то время как Material React Table имеет только бесплатную версию, которая предоставляет основные функции. Исходя из требований к табличному компоненту, удобства использования и скорости освоения была выбрана библиотека AG Grid [3].

AG Grid предоставляет множество возможностей для настройки таблицы и ее поведения;