VEX ROBOTICS АКАДЕМИЯ
WWW.VEXACADEMY.RU

СЦЕНАРИЙ УРОКА
ФУНКЦИОНАЛЬНОЕ УПРАВЛЕНИЕ РОБОТОМ

Цель урока: познакомиться с функциональным управлением роботом и научиться с помощью функции описывать 9 видов движения: вперед, остановка, назад, разворот вперед налево, разворот вперед направо, разворот назад налево, разворот назад направо и разворот на месте.

Результаты:

- Познакомиться с особенностями функционального управления и отличиями от управления с помощью двоичного кодирования.
- Умение эффективно использовать функциональное управление для замедления движения.
- Умение организовать работу с пультом дистанционного управления.
- Формулирование выводов по результатам эксперимента.

Формируемые компетенции:

предметные:
- умение собрать конструкцию согласно инструкции;
- умение подключить микроконтроллер VEX IQ к компьютеру;
- умение использовать функцию `getJoystickValue()`;
- умение подключить пульт дистанционного управления;
- умение запустить программу;

общепредметные:
- умение формулировать выводы по результатам эксперимента;
- умение ориентироваться на заданные критерии;
- умение выбрать из нескольких решений более эффективное;

ключевые:
- поиск и использование обратной связи;
- использование ресурсов;
- способность принимать решения;
- способность к совместной работе ради достижения цели.

Необходимые материалы:

- конструктор VEX IQ на каждую команду;
- компьютер на каждую команду;
- рабочий лист, напечатанный для каждого ученика;
- компьютер и проектор для демонстрирования справочного видео.

Ход урока:

Обсуждение темы урока:

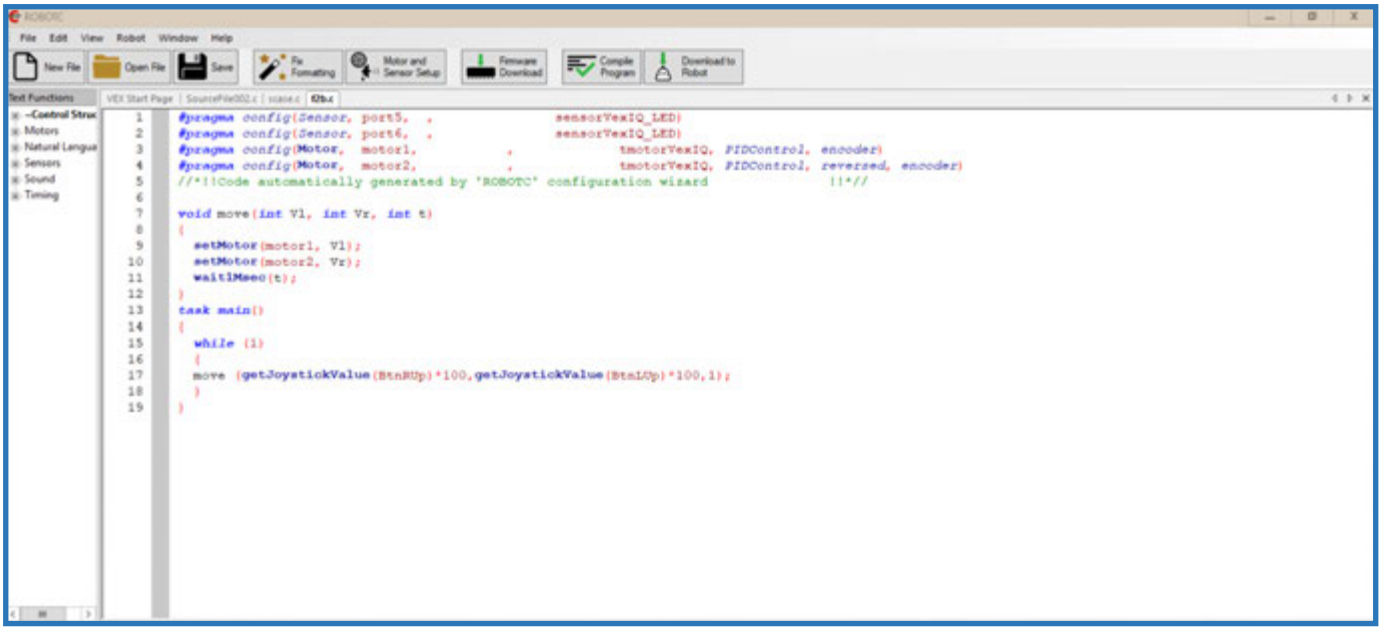
1. На предыдущем уроке была рассмотрена работа с организацией всех видов движения с помощью структуры switch-case. Данный урок будет посвящен устройству функционального управления роботом.
2. Вспомните с учащимися, каким образом на предыдущем занятии была устроена программа по управлению тележкой? С помощью структуры `switch-case` и двоичного кодирования.
3. Чем данный способ был удобен и наоборот?
4. Для организации стандартного набора движений и замедления в нужных моментах существует еще один способ, более универсальный. Назовем его функциональное управление. Его суть состоит в том, что движение описывается одной большой составной функцией (она может состоять из нескольких более простых). На одном из первых уроков по программированию рассматривалась функция **s=v*t**, с помощью которой модно описать движение тележки в любой момент времени. Точно также происходит и с функциональным управлением роботом.
5. На данном занятии учащиеся напишут функцию управления робота через `getJoystickValue()` и простому способу, как замедлить робота в любой момент времени с помощью кнопки.
6. Запишите цель занятия в пункт №1 Рабочего листа.
7. Вспомним, какие функции уже использовались в программах на предыдущих уроках и для чего они нужны? Выполните задание №1.1 Рабочего листа.
8. С помощью какой функцией, представленной в задании, можно попробовать описать 4 вида движения - вперед, развороты и остановка? Это функция `getJoystickValue()`. Используя пример ее работы в задании, а также функцию `move()`, напишите программу движения робота.

Этап конструирования и программирования:

9. Предложите каждой команде собрать тележку по инструкции или самостоятельно.
10. Далее необходимо запрограммировать работу тележки от пульта дистанционного управления.
11. Подключите робота к компьютеру и сделайте инициализацию датчиков и моторов с помощью утилиты **VEXos Utility**.
12. Каждой команде необходимо сделать проверку готовности своего робота к тестированию, отметив все пункты в задании №2.1 Рабочего листа.

13. Первый шаг создания программы функционального управления роботом состоит из написания функции **move()**.

14. Далее в теле **task main()** необходимо вызвать функцию **move()**, но в качестве аргументов записать **getJoystickValue(BtnRUp)*100** для того, чтобы задать скорости левых колес, и **getJoystickValue(BtnLUp)*100**, чтобы задать скорости правых колес.



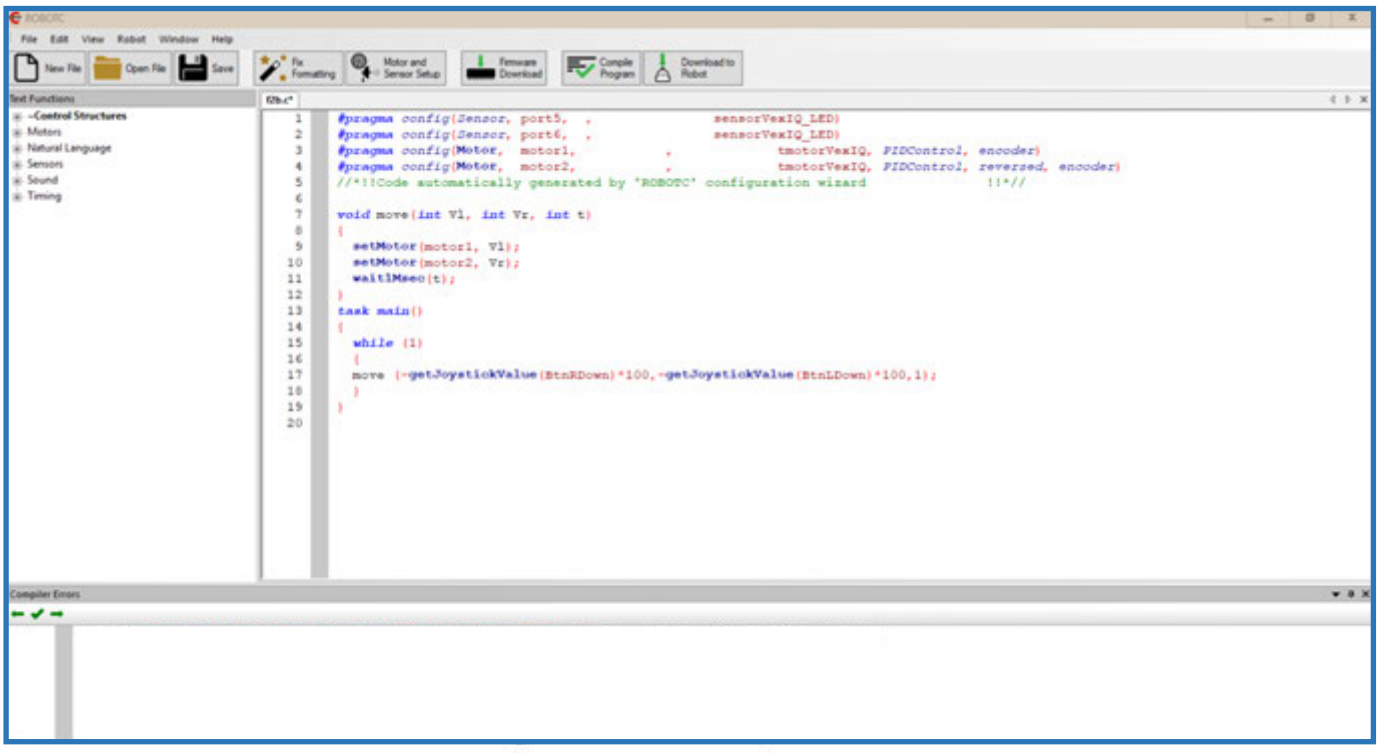
```
#pragma config(Sensor, port5,  ,               sensorVexIQ_LED)
#pragma config(Sensor, port6,  ,               sensorVexIQ_LED)
#pragma config(Motor,  motor1,           ,             tmotorVexIQ, PIDControl, encoder)
#pragma config(Motor,  motor2,           ,             tmotorVexIQ, PIDControl, reversed, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

void move(int Vl, int Vr, int t)
{
  setMotor(motor1, Vl);
  setMotor(motor2, Vr);
  wait1Msec(t);
}
task main()
{
  while (1)
  {
  move (getJoystickValue(BtnRUp)*100,getJoystickValue(BtnLUp)*100,1);
  }
}
```

15. Проверьте работоспособность программы.

Подробно принцип работы программы описан в теоретических сведениях

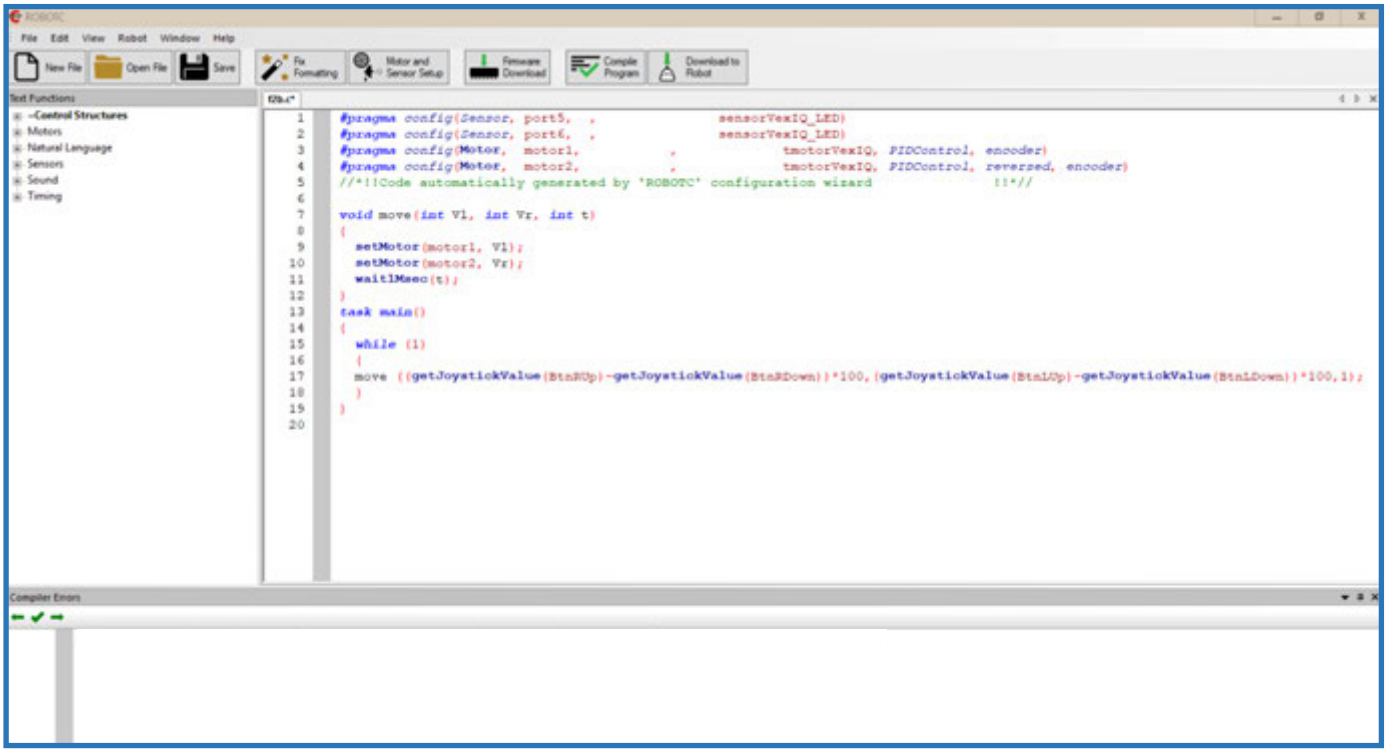
16. Далее поменяйте знаки перед аргументами функции **move()** и измените имена кнопок на те, что будут использоваться для движения назад. Тележка станет двигаться назад и разворачиваться.



```
#pragma config(Sensor, port5,  ,               sensorVexIQ_LED)
#pragma config(Sensor, port6,  ,               sensorVexIQ_LED)
#pragma config(Motor,  motor1,           ,             tmotorVexIQ, PIDControl, encoder)
#pragma config(Motor,  motor2,           ,             tmotorVexIQ, PIDControl, reversed, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

void move(int Vl, int Vr, int t)
{
  setMotor(motor1, Vl);
  setMotor(motor2, Vr);
  wait1Msec(t);
}
task main()
{
  while (1)
  {
  move (-getJoystickValue(BtnRDown)*100,-getJoystickValue(BtnLDown)*100,1);
  }
}
```

17. Таким образом, есть две отдельных функции для движения вперед и назад, необходимо их совместить.
18. Предложите учащимся самостоятельно придумать способ их совместить.
19. Если получить разность аргументов и умножить ее на 100, то получится универсальный способ движения как вперед, так и назад.



20. Рассмотрим подробно, как это работает. Предположим, на пульте нажаты кнопки `BtnRUp` и `BtnLDown`. В таком случае:

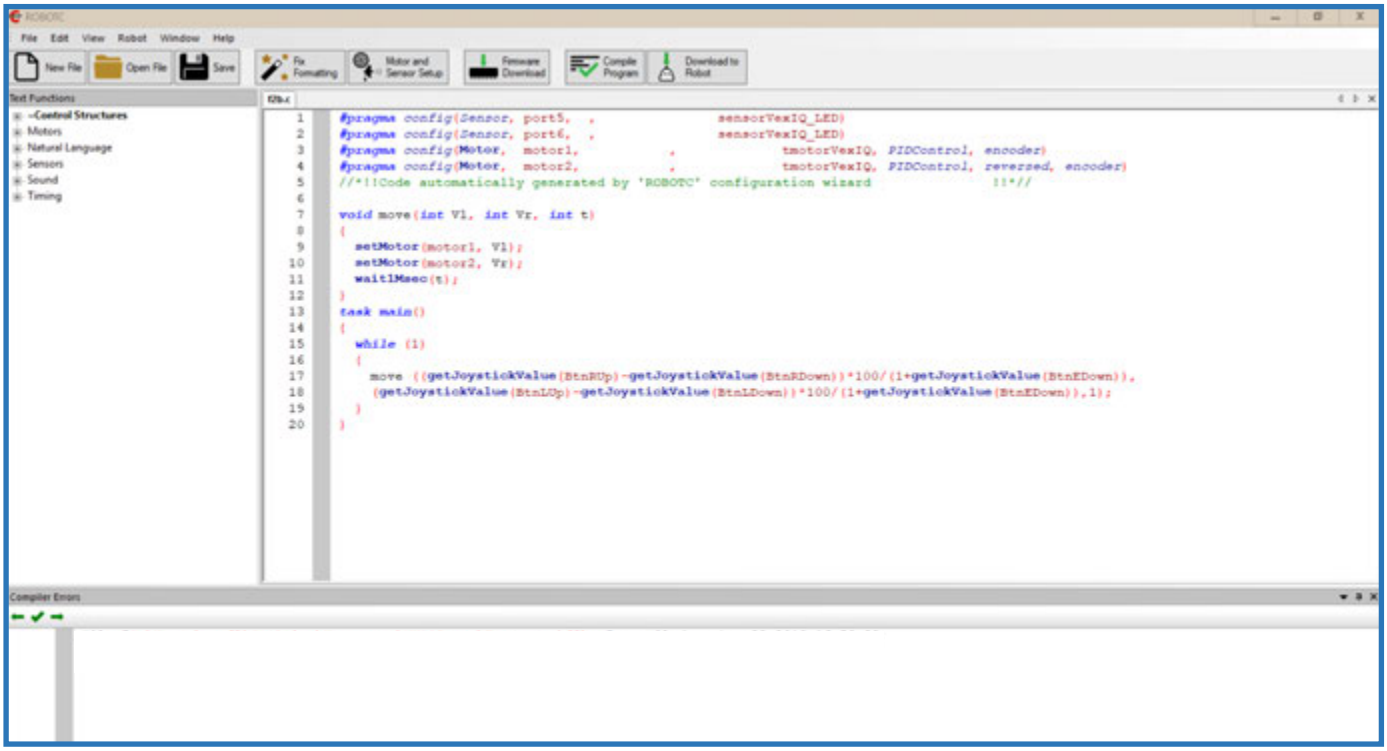
- `getJoystickValue(BtnRUp) = 1;`
- `getJoystickValue(BtnLUp) = 0;`
- `getJoystickValue(BtnRDown) = 0;`
- `getJoystickValue(BtnLDown) = 1;`

21. Подставляем значения в функцию и получаем разворот на месте:

- `move((1-0)*100, (0-1)*100)`
- `move(1*100, -1*100)`
- `move(100, -100)`

22. Далее по такому же принципу необходимо добавить деление на `(1+getJoystickValue(BtnEDown))`. Это позволит в любой момент времени при нажатии на кнопку `BtnEDown` сделать тележку в два раза медленнее.

Подробно замедление описано в теоретических сведениях



Этап проведения эксперимента:

7. Протестируйте тележку на предмет точного соответствия программы и условий задачи, то есть все кнопки пульта работают в соответствие с задумкой и тележка может ехать вперед, разворачиваться 5 разными способами, останавливаться, а также замедляться в любой момент работы робота по нажатию заранее определенной для этого кнопки.
8. Запишите, какое будет осуществляться движение для сочетаний кнопок в задании №2.2 Рабочего листа.

Этап рефлексии:

9. **Обсудите с учащимися** как они понимают, для чего нужно двоичное кодирование? Когда использовать **`switch-case`**, а когда функциональное управление? Почему?
10. Предложите учащимся записать, чему они научились сегодня в задание №3.1, ориентируясь на основной вопрос урока.

Этап приведения кабинета в порядок:

15. Предложите ребятам разобрать тележки следующим образом: разберите на своем рабочем месте все детали и разложите их по видам. Каждый вид положите отдельно в коробку с конструктором.