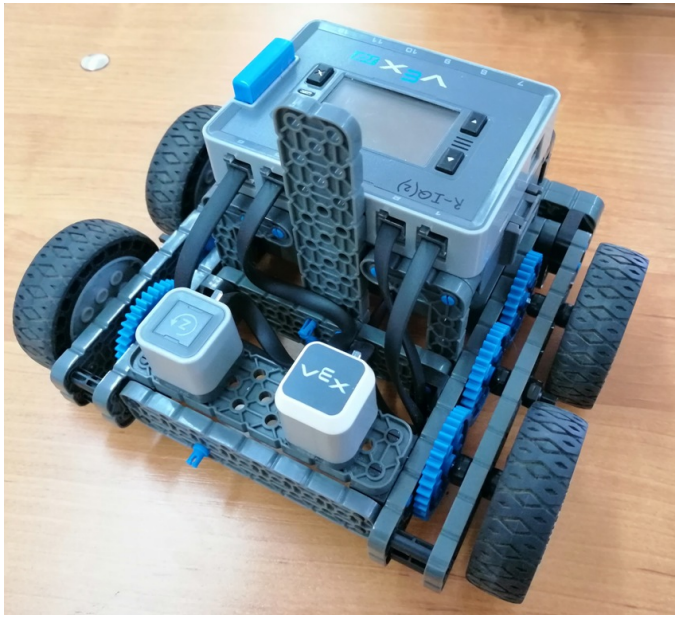
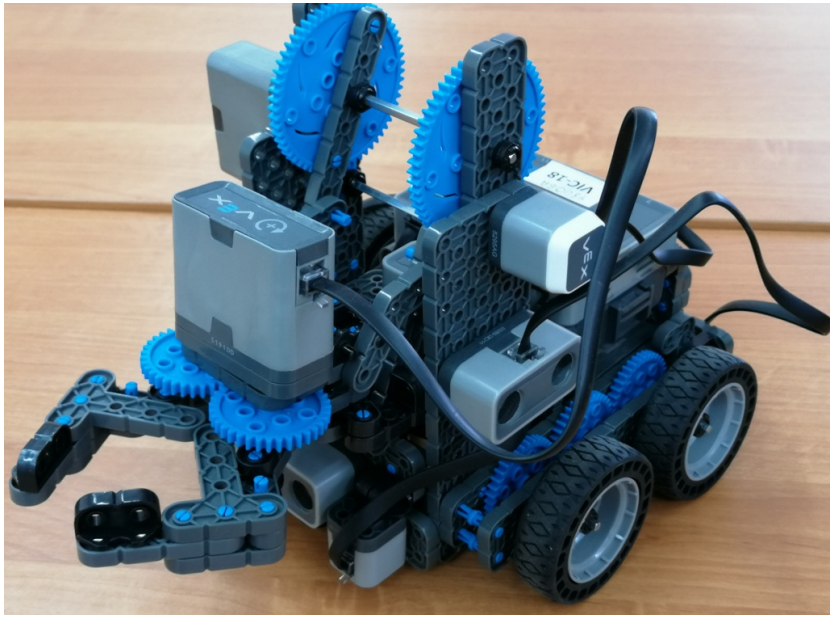
Программирование в среде Robot-C микроконтроллера VEX-IQ

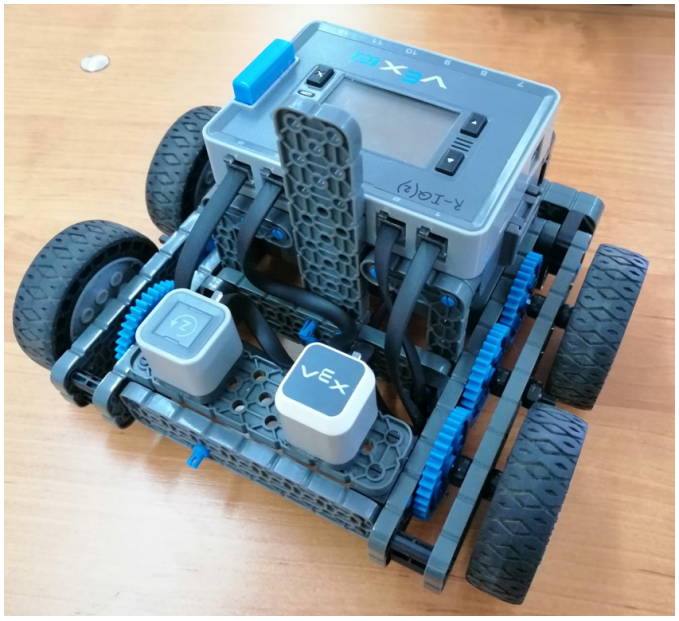


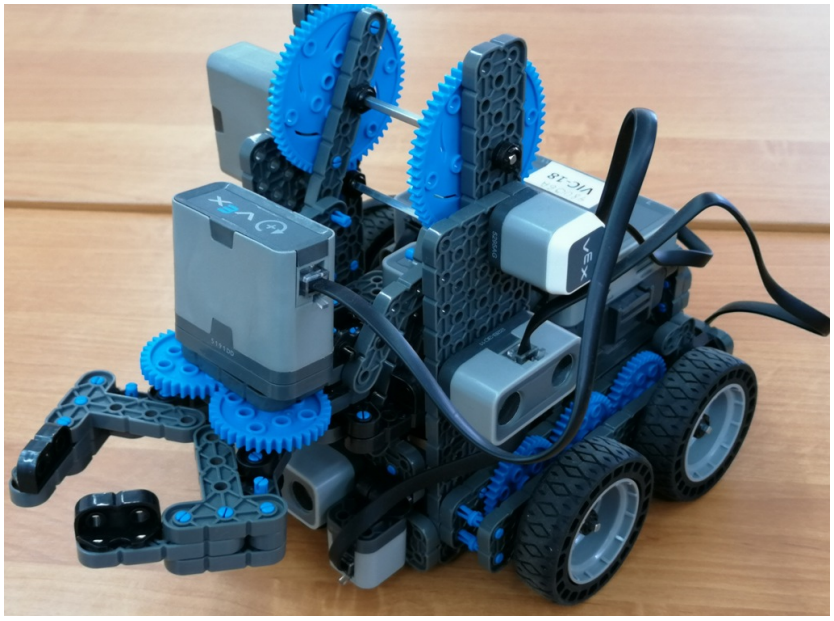


Учитель: Добрый день, ребята! Сегодня мы познакомимся со средой программирования Robot-C и напишем программу по захвату роботом предмета и доставки его в заданную область.

Ученики: Ура! Будет интересно!

Программирование в среде Robot-C микроконтроллера VEX-IQ





Группы устройств и их отличия

Группа	Способ приведения в действие	Наличие программируемых режимов управления	Способность реагировать на изменения окружающей среды
Механические	Мускульная энергия	ДА НЕТ	ДА НЕТ
Непрограммируемые исполнители	Электрическая энергия	ДА НЕТ	ДА НЕТ
Программируемые исполнители	Электрическая энергия	ДА НЕТ	ДА НЕТ
Роботы	Электрическая энергия	ДА НЕТ	ДА НЕТ

место для лишних да\нет

Учитель: А начнём мы с вами с такого задания. В таблице представлены следующие группы устройств: механические, исполнители, программируемые исполнители и роботы. Все они отличаются способом приведения в действие и способностью реагировать на изменения окружающей среды. Кто выйдет и заполнит данную таблицу?

Ученики: Поднимают руки. Один идет к доске.

Группы устройств и их отличия

Группа	Способ приведения в действие	Наличие программируемых режимов управления	Способность реагировать на изменения окружающей среды
Механические	Мускульная энергия	Нет	Нет
Исполнители	Электрическая энергия	Нет	Нет
Программируемые исполнители	Электрическая энергия	Да	Нет
Роботы	Электрическая	Да	Да

Роботы	энергия	Да	Да

Заполните таблицу устройств, согласно их отличиям.

Группы устройств и их отличия

Группа	Способ приведения в действие	Наличие программируемых режимов управления	Способность реагировать на изменения окружающей среды
Механические	Мускульная энергия	ДА НЕТ	ДА НЕТ
Исполнители	Электрическая энергия	ДА НЕТ	ДА НЕТ
Программируемые исполнители	Электрическая энергия	ДА НЕТ	ДА НЕТ
Роботы	Электрическая энергия	ДА НЕТ	ДА НЕТ

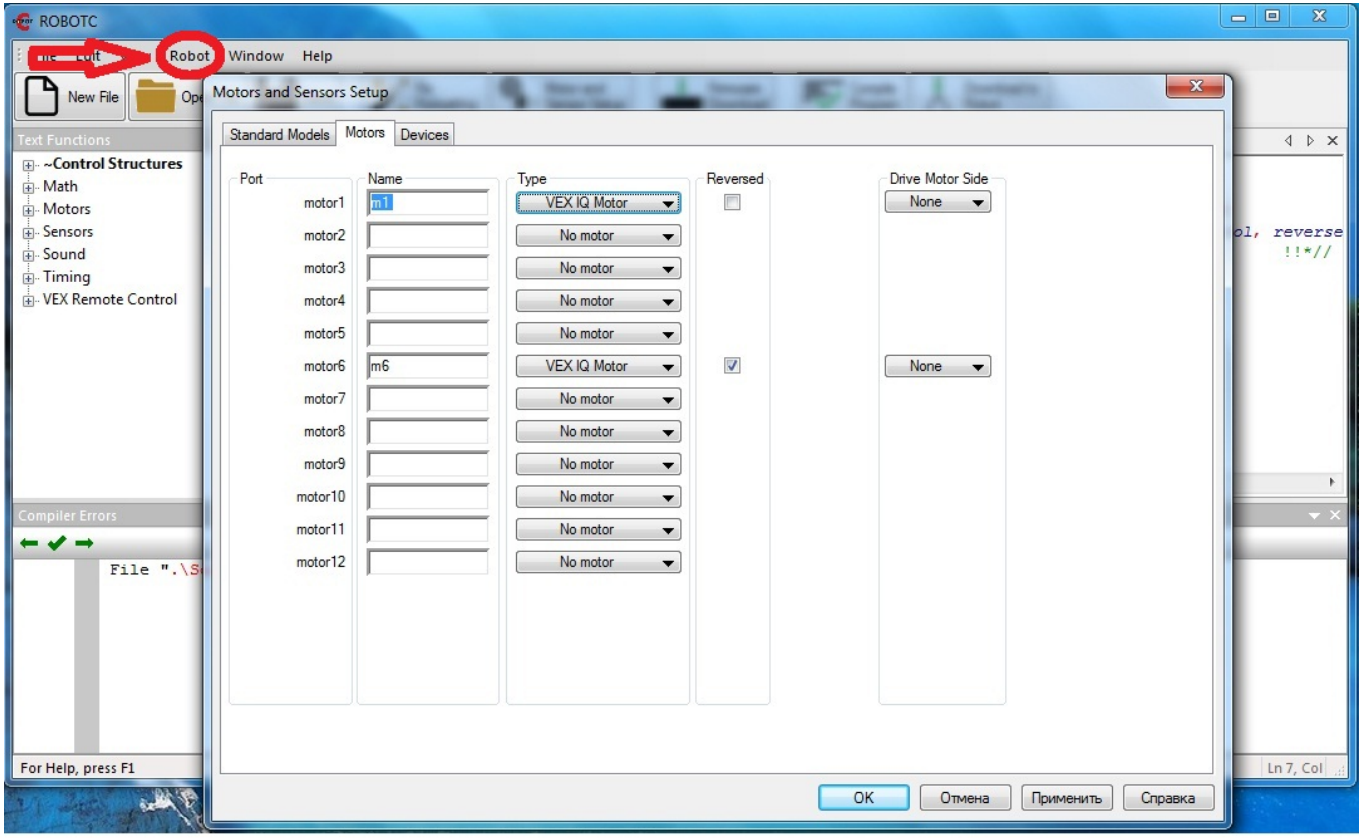
место для лишних да\нет


ROBOT
4.0
ROBOTC for VEX Robotics 4.X


VEXos Utility

Учитель: Робот подключается к компьютеру с помощью USB кабеля, который идет в комплекте. Для программирования робота будем использовать две программы:
Фото1
1. ROBOTC for VEX Robotics 4.X
2. VEXos Utility
Фото2 -VEXos Utility - необходима для того, чтобы диагностировать все подключения датчиков к контроллеру. Если датчик будет неисправен или плохо подключен - программа это покажет.
Фото3 -ROBOTC for VEX Robotics 4.X - среда программирования робота
Ученики: Смотрят внимательно на экран!

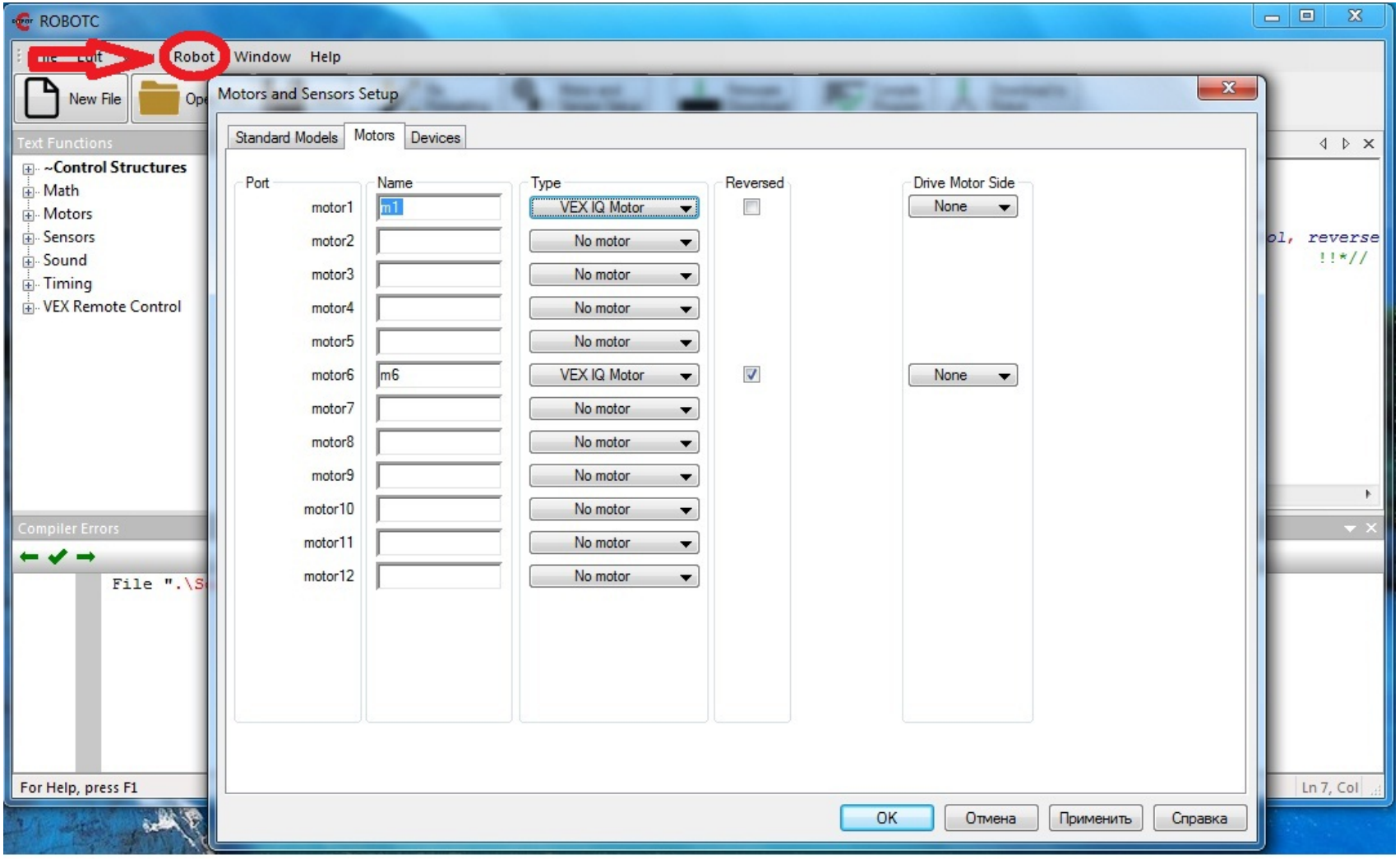
Для программирования робота нам понадобятся две программы:



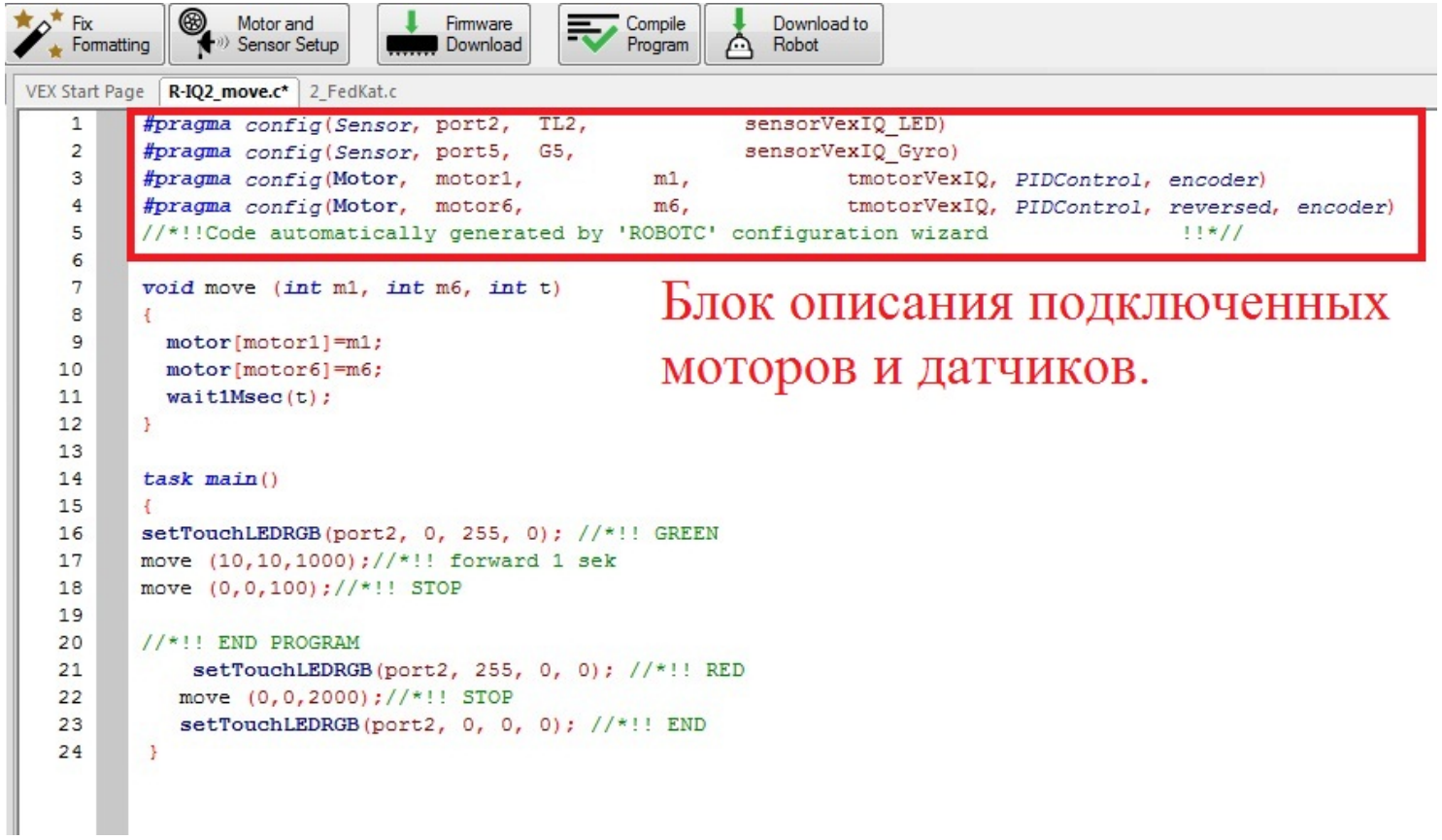
Описание моторов и датчиков в ROBOTC

Учитель: После запуска программы ROBOTC во вкладке **ROBOT-> Motors and Sansors Setup** необходимо прописать, какие моторы (не забыть поставить галочку - **Reversed** у одного из моторов для синхронного вращения) и датчики подключены к контроллеру. Фото1 - описываем моторы, Фото2 - описываем датчики.

Ученики: Смотрят и слушают внимательно.



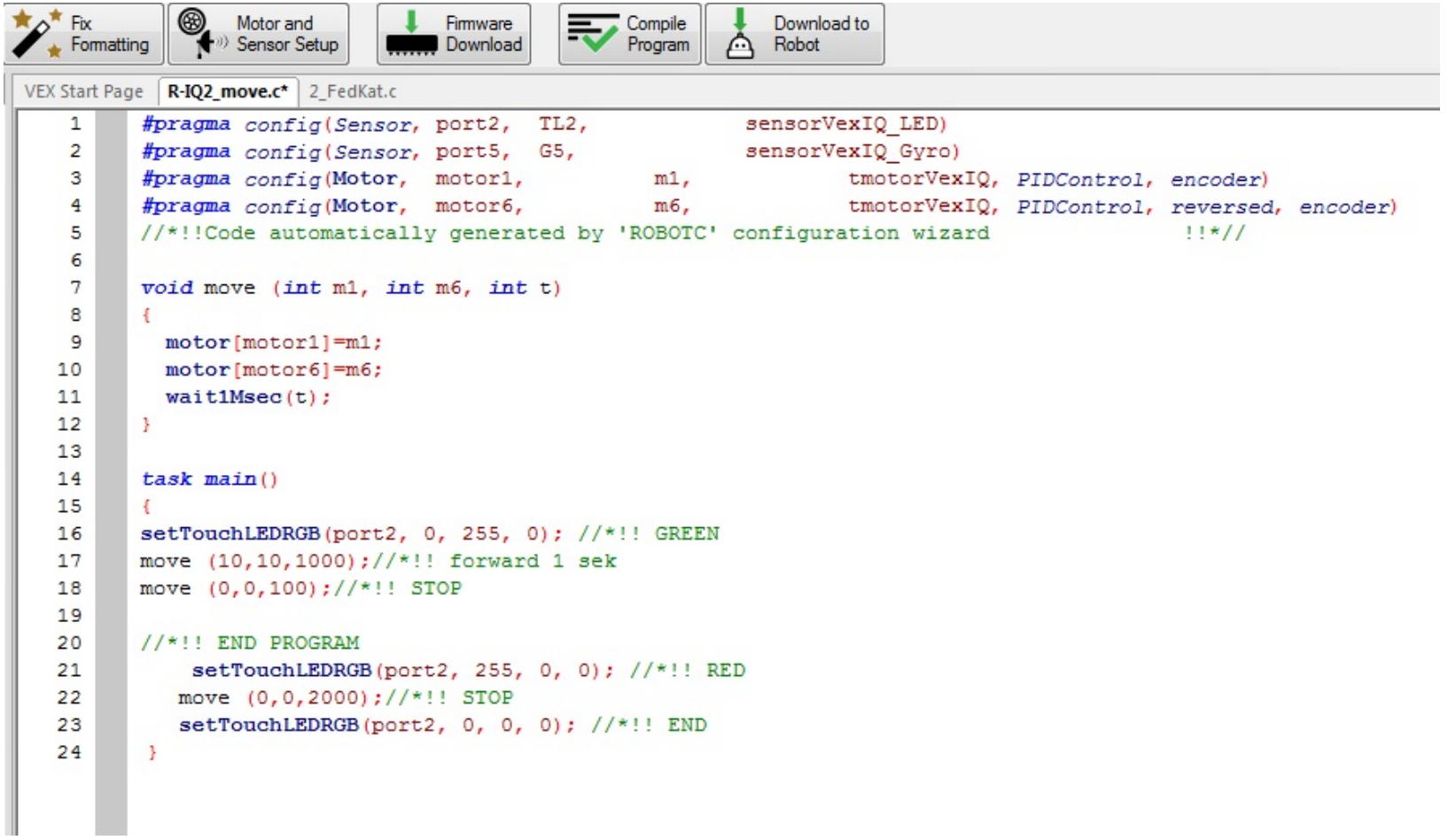
Описание моторов и датчиков в ROBOTC


Fix Formatting
Motor and Sensor Setup
Firmware Download
Compile Program
Download to Robot
VEX Start Page
R-IQ2_move.c*
2_FedKat.c
#pragma config(Sensor, port2, TL2, sensorVexIQ_LED)
#pragma config(Sensor, port5, G5, sensorVexIQ_Gyro)
#pragma config(Motor, motor1, m1, tmotorVexIQ, PIDControl, encoder)
#pragma config(Motor, motor6, m6, tmotorVexIQ, PIDControl, reversed, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
void move (int m1, int m6, int t)
{
motor[motor1]=m1;
motor[motor6]=m6;
wait1Msec(t);
}
task main()
{
setTouchLEDRGB(port2, 0, 255, 0); //*!! GREEN
move (10,10,1000);//*!! forward 1 sek
move (0,0,100);//*!! STOP
//*!! END PROGRAM
setTouchLEDRGB(port2, 255, 0, 0); //*!! RED
move (0,0,2000);//*!! STOP
setTouchLEDRGB(port2, 0, 0, 0); //*!! END
}
Блок описания подключенных моторов и датчиков.

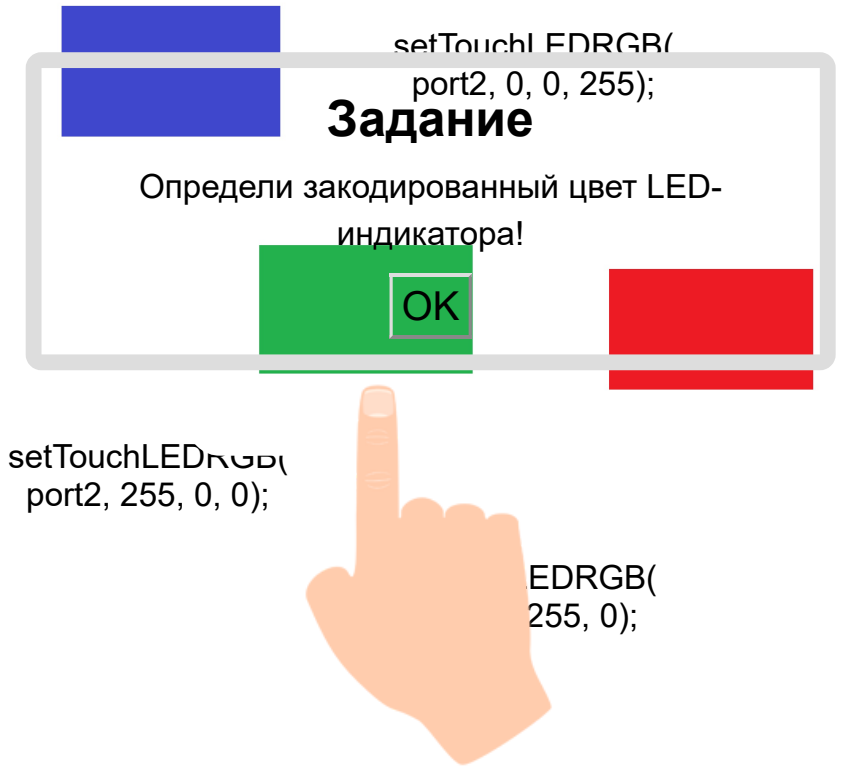
Учитель: Рассмотрим основные элементы программы. Фото1 - Все моторы и датчики, которые мы описали во вкладке *Motors and Sensors Setap* здесь отражены. Фото2 - Функция ***void*** (или еще «общий указатель») - это специальный тип указателя, который может указывать на объекты любого типа данных. В нашем случае, в качестве описания функции движения, будем использовать слово **move.** ***int*** **m1**, ***int*** **m6,** ***int*** **t** - аргументы функции, которые мы должны задать. Они называются переменными. **motor[motor1]=m1;** - задается скорость мотора **m1** от 0 до 100. **wait1Msec**(t); - пауза и t задается в миллисекундах. Фото3 - ***task main()*** - самая главная функция. Её задача заключается в вызове функций, которые написаны у нее в теле (между фигурными скобками **{}**). Фото4 - LED-индикатор способен показывать цвета согласно таблице кодов RGB. В программе это записывается так: **setTouchLEDRGB(port2, 255, 0, 0); //*!! RED**.

Ученики: Смотрят и слушают внимательно.

Программирование робота на движение вперед/назад

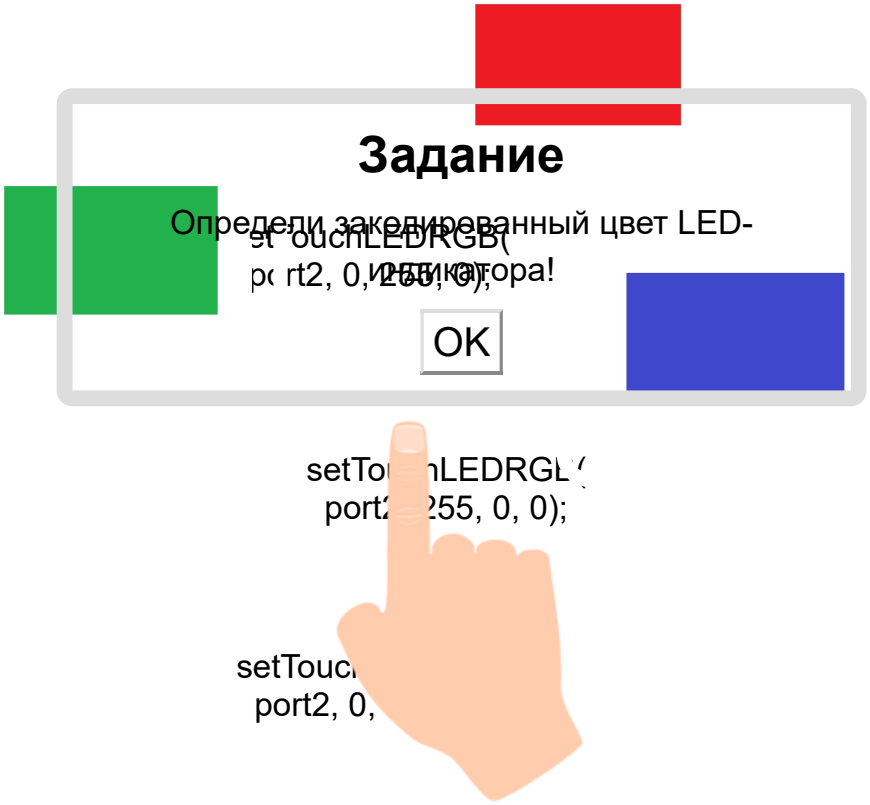


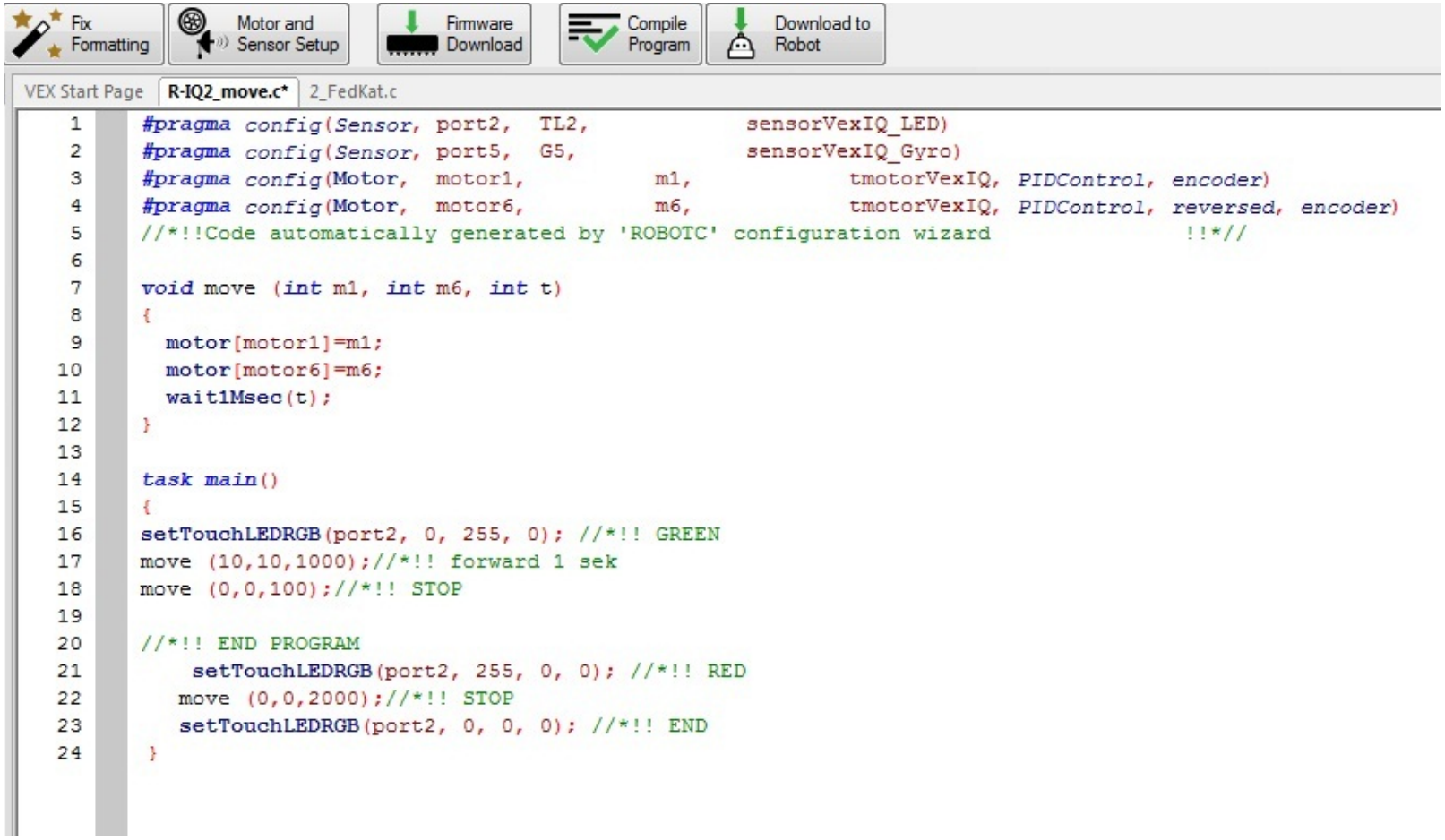

```
1  #pragma config(Sensor, port2,  TL2,            sensorVexIQ_LED)
2  #pragma config(Sensor, port5,  G5,             sensorVexIQ_Gyro)
3  #pragma config(Motor,  motor1,          m1,           tmotorVexIQ, PIDControl, encoder)
4  #pragma config(Motor,  motor6,          m6,           tmotorVexIQ, PIDControl, reversed, encoder)
5  //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
6
7  void move (int m1, int m6, int t)
8  {
9    motor[motor1]=m1;
10   motor[motor6]=m6;
11   wait1Msec(t);
12 }
13
14 task main()
15 {
16 setTouchLEDRGB(port2, 0, 255, 0); //*!! GREEN
17 move (10,10,1000);//*!! forward 1 sek
18 move (0,0,100);//*!! STOP
19
20 //*!! END PROGRAM
21     setTouchLEDRGB(port2, 255, 0, 0); //*!! RED
22    move (0,0,2000);//*!! STOP
23    setTouchLEDRGB(port2, 0, 0, 0); //*!! END
24  }
```


setTouchLEDRGB(
port2, 0, 0, 255);
Задание
Определи закодированный цвет LED-индикатора!
OK
setTouchLEDRGB(
port2, 255, 0, 0);
EDRGB(
255, 0);

Учитель: А сейчас, посмотрим, кто был наиболее внимателен!

Ученики: Тянут руки. Один идёт к доске.


Задание
Определи закодированный цвет LED-индикатора!
OK
setTouchLEDRGB(
port2, 0, 255, 0);
setTouchLEDRGB(
port2, 255, 0, 0);



Программы движения вперед и назад

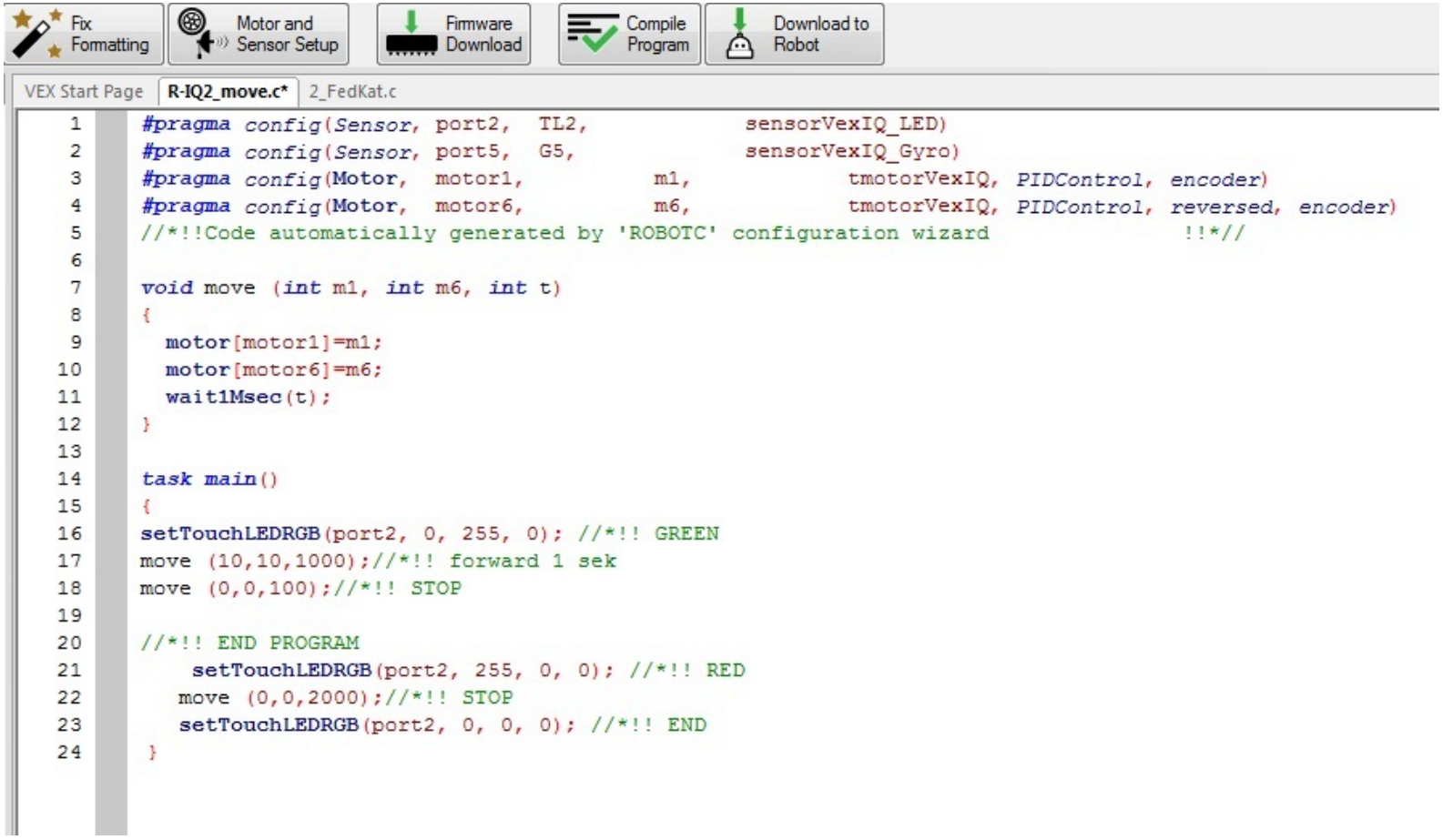
Учитель: Рассмотрим программу, которая поможет нам заставить робота двигаться. Фото1
Первая строка - индикаторная - для сообщения нам о том, что программа роботом исполняется:
setTouchLEDRGB(port2, 0, 255, 0); //*!! GREEN - зеленый цвет LED загорается в самом начале исполнения программы;

В нашей программе основными являются две строчки, которые и отвечают за движение робота:
move (10,10,1000);//*!! forward 1 sek - движение вперёд одну секунду;
move (0,0,100);//*!! STOP - остановка движения.

Учитель: Для того, чтобы робот стал двигаться назад, нам нужно изменить код в программе! Кто ответит, что именно?
Ответ: Поставить знаки "минус": move (-10,-10,1000) Фото2

Ученики: Все тянут руки. Один идёт к доске.

Программы движения вперед и назад

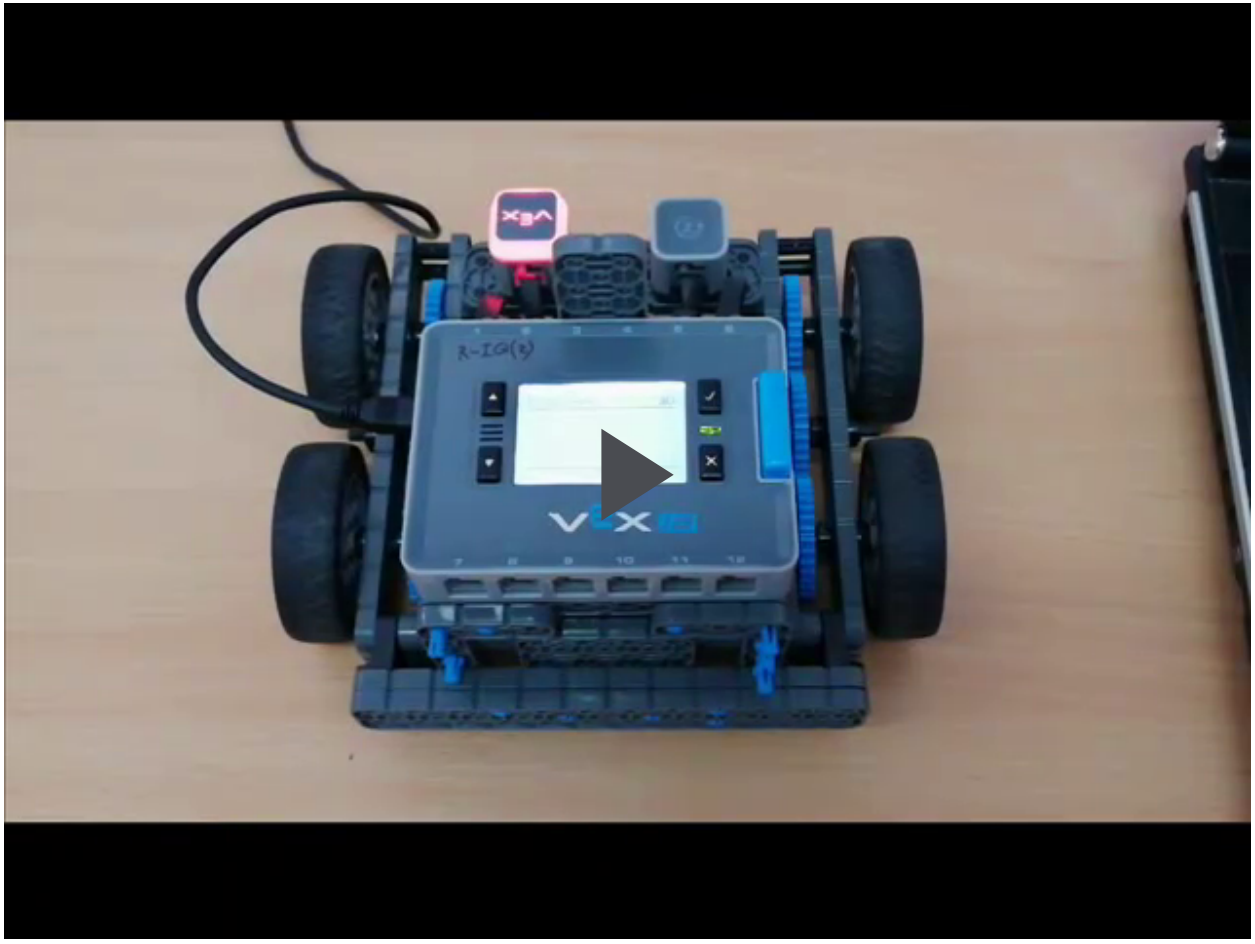



```
1  #pragma config(Sensor, port2,  TL2,            sensorVexIQ_LED)
2  #pragma config(Sensor, port5,  G5,             sensorVexIQ_Gyro)
3  #pragma config(Motor,  motor1,          m1,           tmotorVexIQ, PIDControl, encoder)
4  #pragma config(Motor,  motor6,          m6,           tmotorVexIQ, PIDControl, reversed, encoder)
5  //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
6
7  void move (int m1, int m6, int t)
8  {
9    motor[motor1]=m1;
10   motor[motor6]=m6;
11   wait1Msec(t);
12 }
13
14 task main()
15 {
16 setTouchLEDRGB(port2, 0, 255, 0); //*!! GREEN
17 move (10,10,1000);//*!! forward 1 sek
18 move (0,0,100);//*!! STOP
19
20 //*!! END PROGRAM
21     setTouchLEDRGB(port2, 255, 0, 0); //*!! RED
22    move (0,0,2000);//*!! STOP
23    setTouchLEDRGB(port2, 0, 0, 0); //*!! END
24  }
```

Тест:

Программа для движение робота VEXiq вперед и назад в среде ROBOTC

Загрузка программы в робота и запуск



Учитель: Загрузить программу в контроллер можно только через USB-кабель. Подключаем робота к компьютеру. Запускаем видео.
Ученики: Смотрят и слушают внимательно.

ВНИМАНИЕ НА ЭКРАН!

Fix Formatting | Motor and Sensor Setup | Firmware Download | Compile Program | Download to Robot

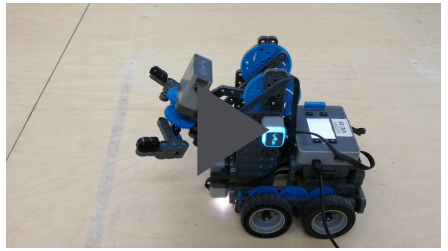
R-IQ1_Take.c*

```
#pragma config(Motor,  motor4,          z4,            tmotorVexIQ, PIDControl, encoder)
#pragma config(Motor,  motor6,          m6,            tmotorVexIQ, PIDControl, reversed, encoder)
#pragma config(Motor,  motor10,         up10,          tmotorVexIQ, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//       !!*//
void move (int m1, int m6, int t)
{
  motor[motor1]=m1;
  motor[motor6]=m6;
  wait1Msec(t);
  }
   void take (int z4,int up10, int t)
{
  motor[motor4]=z4;
  motor[motor10]=up10;
  wait1Msec(t);
}
task main()
{
setTouchLEDRGB(port7, 0,0,255);//*BLUE
   take (-10,0,1000); //*Open 1 sec
   take (0,0,100);//*stop zahvat

setTouchLEDRGB(port7, 0,255,255);//*CYAN
   take (0,10,1500); //*UP 1,5 sec
   take (0,0,10);//*stop zahvat

setTouchLEDRGB(port7, 0,0,255);//*BLUE
   take (10,0,1000); //*Close 1 sec
   take (0,0,100);//*stop zahvat

 setTouchLEDRGB(port7, 0,255,255);//*CYAN
   take (0,-10,1500); //*Down 1,5 sec
   take (0,0,10);//*stop zahvat
```



Программа тестирования манипулятора

Учитель: Фото1 В программу мы добавляем новую функцию *take*, которая будет отвечать за захват объекта:

void ***take*** (int z4,int up10, int t) - объявляем новую функцию *take.*

Далее идет её описание:

motor[motor4]=z4; - мотор, отвечающий за зажим.

motor[motor10]=up10; - мотор, отвечающий за подъем.

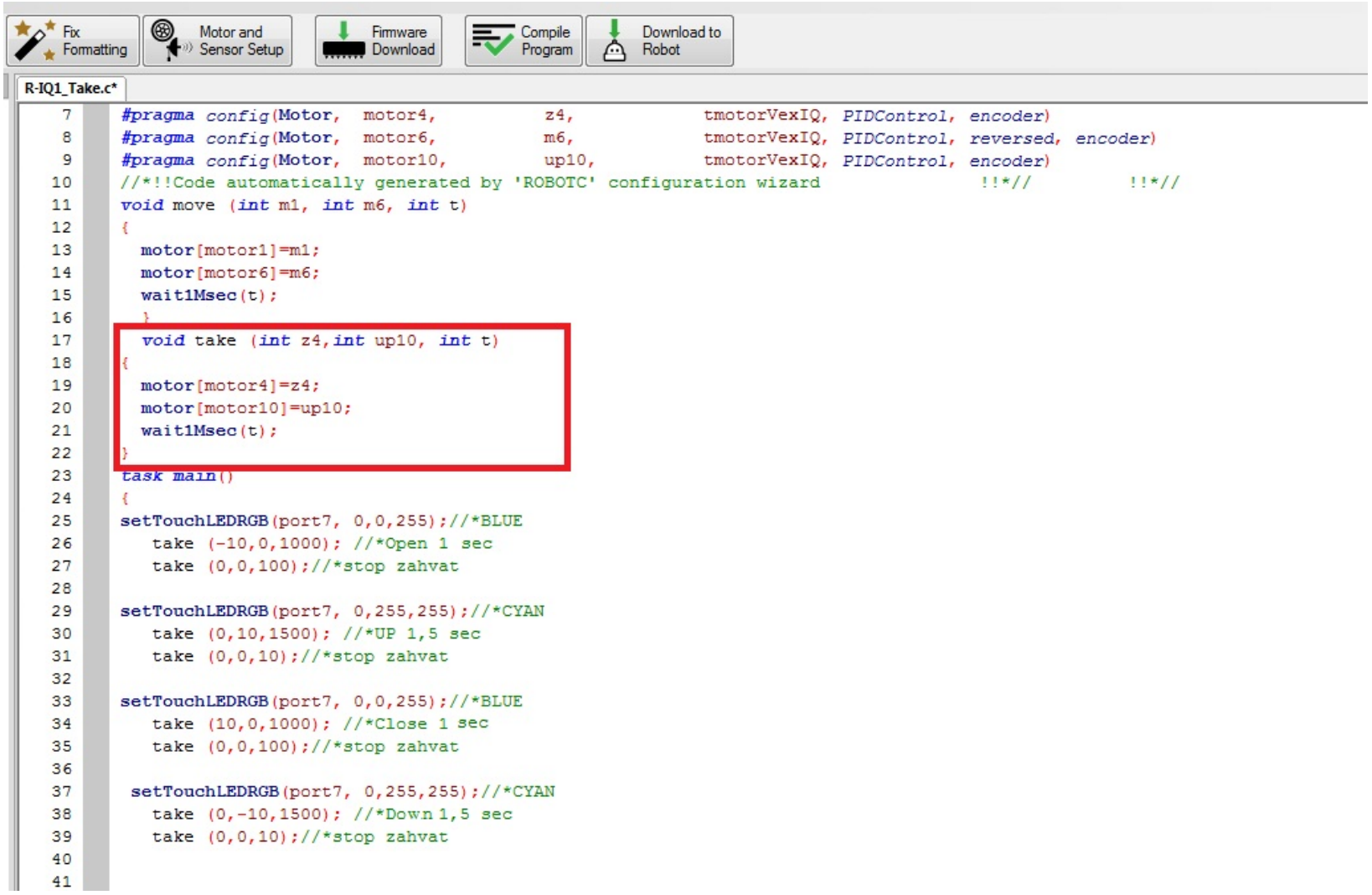
wait1Msec(t); - задаем время работы моторов в миллисекундах.

take (-10,0,1000); - со скоростью 10 захват открывается 1 секунду.

Запускаем видео.

Ученики: Смотрят и слушают внимательно.

Блок программы захвата объекта




```
7  #pragma config(Motor,  motor4,           z4,            tmotorVexIQ, PIDControl, encoder)
8  #pragma config(Motor,  motor6,           m6,            tmotorVexIQ, PIDControl, reversed, encoder)
9  #pragma config(Motor,  motor10,          up10,          tmotorVexIQ, PIDControl, encoder)
10 //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//       !!*//
11 void move (int m1, int m6, int t)
12 {
13   motor[motor1]=m1;
14   motor[motor6]=m6;
15   wait1Msec(t);
16   }
17   void take (int z4,int up10, int t)
18 {
19   motor[motor4]=z4;
20   motor[motor10]=up10;
21   wait1Msec(t);
22 }
23 task main()
24 {
25 setTouchLEDRGB(port7, 0,0,255);//*BLUE
26    take (-10,0,1000); //*Open 1 sec
27    take (0,0,100);//*stop zahvat
28
29 setTouchLEDRGB(port7, 0,255,255);//*CYAN
30    take (0,10,1500); //*UP 1,5 sec
31    take (0,0,10);//*stop zahvat
32
33 setTouchLEDRGB(port7, 0,0,255);//*BLUE
34    take (10,0,1000); //*Close 1 sec
35    take (0,0,100);//*stop zahvat
36
37  setTouchLEDRGB(port7, 0,255,255);//*CYAN
38    take (0,-10,1500); //*Down 1,5 sec
39    take (0,0,10);//*stop zahvat
40
41
```

Тест:

Усвоение программы управления манипулятором робота VEXiq в ROBOTC

Fix Formatting | Motor and Sensor Setup | Firmware Download | Compile Program | Download to Robot

R-IQ1_Take.c | **R-IQ1_Distans.c**

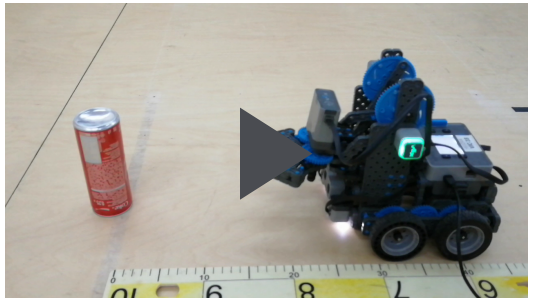
```
#pragma config(Motor,  motor6,          m6,            tmotorVexIQ, PIDControl, reversed, encoder)
#pragma config(Motor,  motor10,         up10,          tmotorVexIQ, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//       !!*//

void move (int m1, int m6, int t)
{
  motor[motor1]=m1;
  motor[motor6]=m6;
  wait1Msec(t);
  }
  void take (int z4,int up10, int t)
{
  motor[motor4]=z4;
  motor[motor10]=up10;
  wait1Msec(t);
}


task main()
{
//*!!____BEGIN_Movement on US-sensor
while(SensorValue[port3]>200) //*move forward US
                    {
                    setTouchLEDRGB(port7, 0,255,0);//*GREEN
                    move (-10,-10,1);//*move forward
                     }
//*!!____END_Movement on US-sensor

   move (0,0,10); //*stop move


//*!! END PROGRAM
    setTouchLEDRGB(port7, 255, 0, 0); //*!! RED
    wait1Msec(2000);//*PAUSE 2 sec
   move (0,0,10); //*stop move
   take (0,0,10);//*stop zahvat
   setTouchLEDRGB(port7, 0, 0, 0); //*!! Light off
```



Движение по ультразвуковому датчику

Учитель: Фото1. Нам осталось научить робота видеть перед собой препятствие и останавливаться на заданном расстоянии от него. Для этого будем использовать следующую функцию:

while(SensorValue[port3]>200) *- пока расстояние более 200 мм выполняется программа движения вперед:*

```
{
 move (-10,-10,1);//*move forward
 }
```

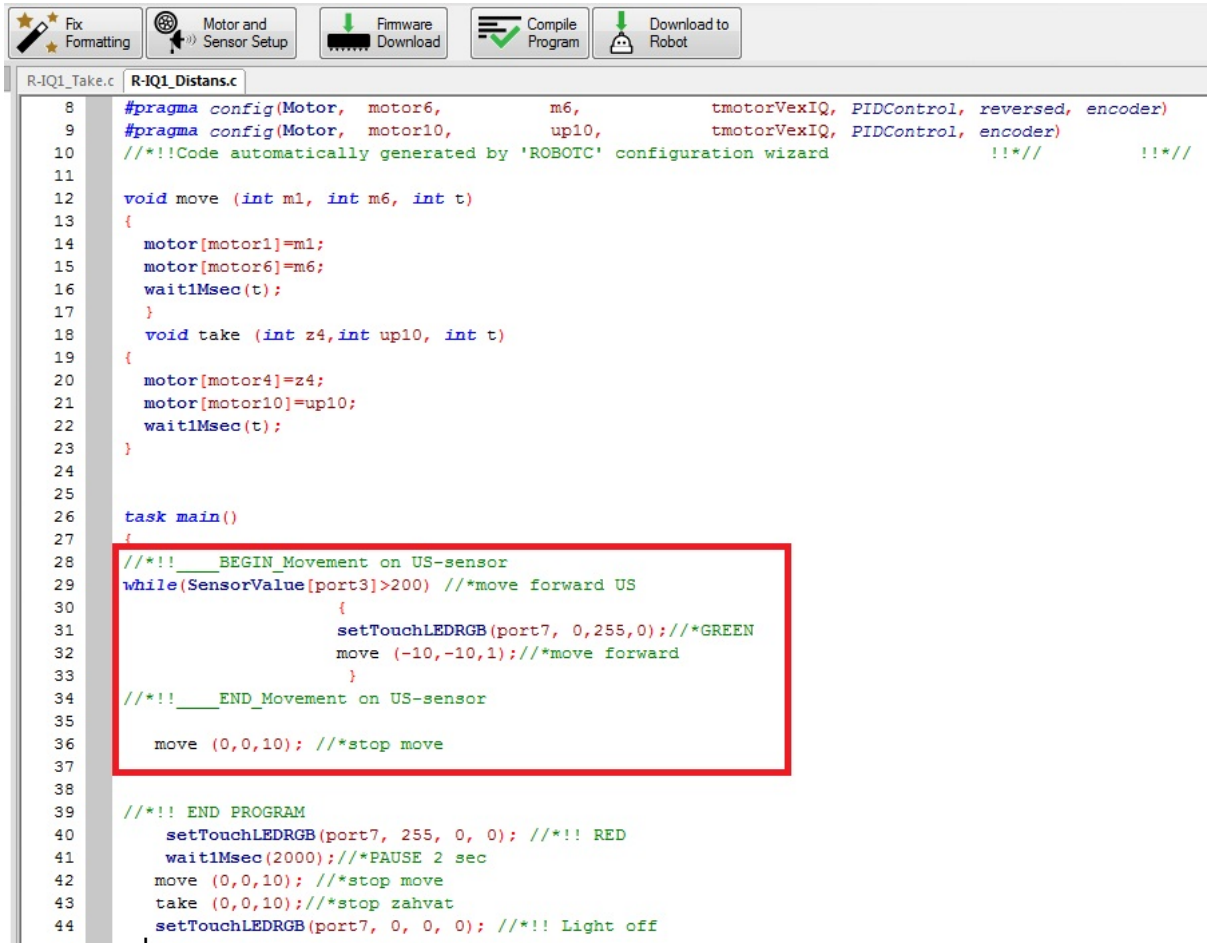
как только условие не выполняется, следует выполнение следующего действия, а именно, остановки:

```
move (0,0,10); //*stop move
```

Запускаем видео.

Ученики: Смотрят и слушают внимательно.

Движение по ультразвуковому датчику




```
#pragma config(Motor,  motor6,          m6,            tmotorVexIQ, PIDControl, reversed, encoder)
#pragma config(Motor,  motor10,         up10,          tmotorVexIQ, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//       !!*//

void move (int m1, int m6, int t)
{
  motor[motor1]=m1;
  motor[motor6]=m6;
  wait1Msec(t);
  }
  void take (int z4,int up10, int t)
{
  motor[motor4]=z4;
  motor[motor10]=up10;
  wait1Msec(t);
}


task main()
{
//*!!___BEGIN_Movement on US-sensor
while(SensorValue[port3]>200) //*move forward US
                    {
                    setTouchLEDRGB(port7, 0,255,0);//*GREEN
                    move (-10,-10,1);//*move forward
                     }
//*!!___END_Movement on US-sensor

   move (0,0,10); //*stop move

//*!! END PROGRAM
    setTouchLEDRGB(port7, 255, 0, 0); //*!! RED
    wait1Msec(2000);//*PAUSE 2 sec
   move (0,0,10); //*stop move
   take (0,0,10);//*stop zahvat
   setTouchLEDRGB(port7, 0, 0, 0); //*!! Light off
```

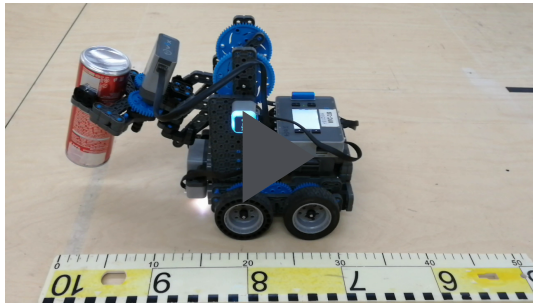
Тест:

Движения по ультразвуковому датчику робота VEXiq

Fix Formatting | Motor and Sensor Setup | Firmware Download | Compile Program | Download to Robot

R-IQ1_Take.c | R-IQ1_take_object.c

```
25
26  task main()
27  {
28  setTouchLEDRGB(port7, 0,0,255);//*BLUE
29     take (-10,0,1000); //*Open 1 sec
30     take (0,0,100);//*stop zahvat
31
32  //*!!___BEGIN_Movement on US-sensor
33  while(SensorValue[port3]>60) //*move forward US
34                       {
35                       setTouchLEDRGB(port7, 255,0,255);//*yellow
36                       move (-10,-10,1);//*move forward
37                        }
38  //*!!___END_Movement on US-sensor
39     move (0,0,10); //*stop move
40
41  setTouchLEDRGB(port7, 0,0,255);//*BLUE
42     take (10,0,1500); //*Close 1,5 sec
43     take (0,0,100);//*stop zahvat
44
45     setTouchLEDRGB(port7, 0,255,255);//*CYAN
46     take (0,10,1500); //*UP 1,5 sec
47     take (0,0,10);//*stop zahvat
48
49     move (10,10,5000);//*move back 5 sec
50     move (0,0,10); //*stop move
51
52     setTouchLEDRGB(port7, 0,255,255);//*CYAN
53     take (0,-10,1500); //*Down 1,5 sec
54     take (0,0,10);//*stop zahvat
55
56     setTouchLEDRGB(port7, 0,0,255);//*BLUE
57     take (-10,0,1500); //*Open 1,5 sec
58     take (0,0,100);//*stop zahvat
59
60     move (10,10,3000);//*move back 3 sec
61     move (0,0,10); //*stop move
62
63  //*!! END PROGRAM
```

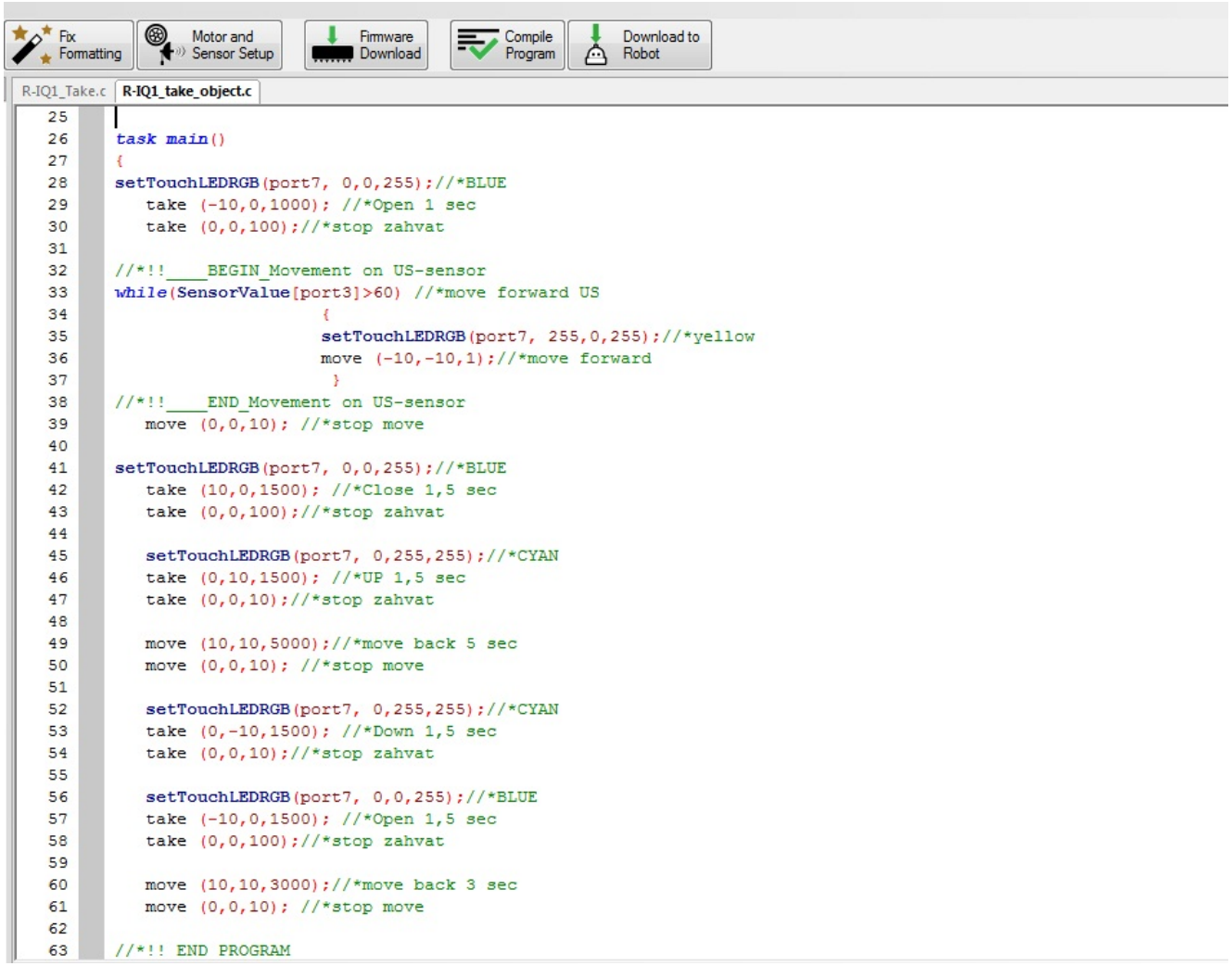


Программа захвата объекта

Учитель: Фото1. Рассмотрим программу захвата объекта. Она собрана из уже рассмотренных нам блоков. Кто попробует, глядя на программу, предугадать, как поведёт себя робот? А после этого мы запустим видео и сравним предсказание с действительностью!

Ученики: Поднимают руки. Один идет к доске и даёт комментарий. Затем запускается видео.

Программа захвата объекта




```
25
26  task main()
27  {
28  setTouchLEDRGB(port7, 0,0,255);//*BLUE
29     take (-10,0,1000); //*Open 1 sec
30     take (0,0,100);//*stop zahvat
31
32  //*!!____BEGIN_Movement on US-sensor
33  while(SensorValue[port3]>60) //*move forward US
34                       {
35                       setTouchLEDRGB(port7, 255,0,255);//*yellow
36                       move (-10,-10,1);//*move forward
37                        }
38  //*!!____END_Movement on US-sensor
39     move (0,0,10); //*stop move
40
41  setTouchLEDRGB(port7, 0,0,255);//*BLUE
42     take (10,0,1500); //*Close 1,5 sec
43     take (0,0,100);//*stop zahvat
44
45     setTouchLEDRGB(port7, 0,255,255);//*CYAN
46     take (0,10,1500); //*UP 1,5 sec
47     take (0,0,10);//*stop zahvat
48
49     move (10,10,5000);//*move back 5 sec
50     move (0,0,10); //*stop move
51
52     setTouchLEDRGB(port7, 0,255,255);//*CYAN
53     take (0,-10,1500); //*Down 1,5 sec
54     take (0,0,10);//*stop zahvat
55
56     setTouchLEDRGB(port7, 0,0,255);//*BLUE
57     take (-10,0,1500); //*Open 1,5 sec
58     take (0,0,100);//*stop zahvat
59
60     move (10,10,3000);//*move back 3 sec
61     move (0,0,10); //*stop move
62
63  //*!! END PROGRAM
```

1 / 5

В какой программе можно определить что датчик, подключенный к контроллеру VEX, исправен?

Задание

Выберите правильный ответ!

OK

Прямая ссылка на тест для вывода QR-кода на экран

Учитель:

1. В какой программе можно определить что датчик, подключенный к контроллеру VEX, не исправен?
Ответ: VEXos Utility.

2. Программа, написанная в среде ROBOTC, загружается быстрее по каналу WiFi или Bluetooth?
Ответ: Только по USB кабелю можно загрузить программу.

3. Как мы используем функцию ***void*** в нашей программе?
*Ответ: С её помощью задаем функцию движения - **muve** и функцию управления манипулятором - **take.***

4. Какую роль играет функция ***task main()*** в нашей программе?
Ответ: Самая главная функция - Ее задача вызывать к работе те функции, что написаны у нее в "теле".

5. Для чего нужны фигурные скобки { } в нашей программе?
Ответ: Для определения "тела" функции.

Ученики: Выходят по одному и отвечают на вопросы.

Для закрепления материала ответим на следующие вопросы:

1. В какой программе можно определить что датчик, подключенный к контроллеру VEX, не исправен?
2. Программа, написанная в среде ROBOTC, загружается быстрее по каналу WiFi или Bluetooth?
3. Как мы используем функцию ***void*** в нашей программе?
4. Какую роль играет функция ***task main()*** в нашей программе?
5. Для чего нужны фигурные скобки **{ }** в нашей программе?