Дистанционное управление роботом NXT с помощью графического планирования движения по заданному маршруту»

Оглавление

1. Введение

Робототехника - область науки и техники, связанная с изучением, созданием и использованием принципиально нового технического средства комплексной автоматизации производственных процессов - робототехнических систем.

С развитием робототехники, роботы стали использоваться во многих областях науки, техники и промышленности, в первую очередь там, где жизнедеятельность человека либо затруднена, либо вообще невозможна, например, в зонах радиоактивного или химического загрязнения, в условиях боевых действий, при проведении подводных или космических исследований и т.п.

Задачи проекта:

1. Спроектировать и реализовать робота, отвечающим следующим требованиям:
   - Возможность движения вперед
   - Возможность поворота налево и направо
2. Спроектировать протокол передачи данных по bluetooth
3. Разработать графическое приложение на ПК, которое позволит проектировать движение робота по заданному маршруту.
4. Разработать программное обеспечение робота, которое позволит ему на основании интерфейса bluetooth принимать команды от графического приложения с помощью спроектированного протокола и отдавать результат выполнения команды.

2. Основная часть

2.1 Сборка робота

При сборке робота необходимо учитывать прочность конструкции и простоту движения.

Выделены основные части робота:

- два сервоприводах
- поворотное колесо

Такая конструкция позволяет роботу легко перемещаться по любой поверхности пола: и по ковру, и по твёрдому покрытию.

Во время сборки были рассчитаны количество оборотов сервопривода для поворотов налево и направо. Так же был рассчитан количество оборотов для движения прямо и назад.

Данный робот будет использоваться как простой автомобиль, умеющий поворачивать и двигаться прямо.

2.2 Протокол передачи данных

В качестве языка реализации был выбран C++. Для управления роботом по bluetooth была разработана библиотека, которая расширяет BOOST ASIO. Данная библиотека имеет поддержку асинхронного взаимодействия между двумя приложениями с помощью объекта службы ввода-вывода(I/O).

2.2.1 Формат данных

Пакет данных: Состояние робота

Для изменения состояние робота в получении новых команд необходимо изменить состояние робота.

Поле	**Тип данных**	**Описание**
Статус робота	8-битовый без знака	Может принимать два состояния: активен(1), остановлен(0)

Пакет данных: Команды

Для отправки команд движения и поворота роботом был разработан формат общения между роботом и графическим приложением:

Поле	**Тип данных**	**Описание**
Команда	8-битовый без знака	Команда управления
Значение	8-битовый без знака	Значение команды
Размер блока карты	8-битовый без знака	Данное значение является коэффициентом для размера карты (в случае если командой является движение прямо и назад)

Список доступных команд: Движение прямо, Движение назад, Движение влево, Движение вправо

Пакет данных: Статус активной команды

Статус выполнение той или иной команды после выполнения отправляется роботом в графическое приложения, для получения состояния пройденного этапа.

Поле	**Тип данных**	**Описание**
Статус	8-битовый без знака	Может принимать два состояния: да(1), нет(0)

2.3 Графическое приложение

Для реализации графического приложения был выбран язык C++. Зависимые библиотеки: boost, blez, imgui, sfml.

Доступные платформы: Linux

2.3.1 Соединение с роботом

Перед запуском построенного маршрута, необходимо подключиться к роботу в графическом приложении по интерфейсу Bluetooth.

Первоначальная настройка подключения к роботу состоит из нескольких шагов:

1. Включить Bluetooth на обоих устройствах и установить режим видимый для всех.

2. Включить поиск устройств на ПК и в появившемся списке выбрать робота

3. В ответ на запрос к подключению робот издаст звуковой сигнал и предложит пароль для установления соединения, нужно его запомнить и нажать на желтую кнопку, чтобы подтвердить свой выбор.

4. Ввести сохраненный пароль с робота в появившуюся строку на компьютере.

5. Соединение между ПК и роботом установлено

6. Получить MAC адрес робота

MAC адрес будет необходим для подключения графического приложения к роботу.

Для того чтобы подключится к роботу из приложения. Необходимо в верхнем меню выбрать пункт меню «Configure» и выбрать «Connect».

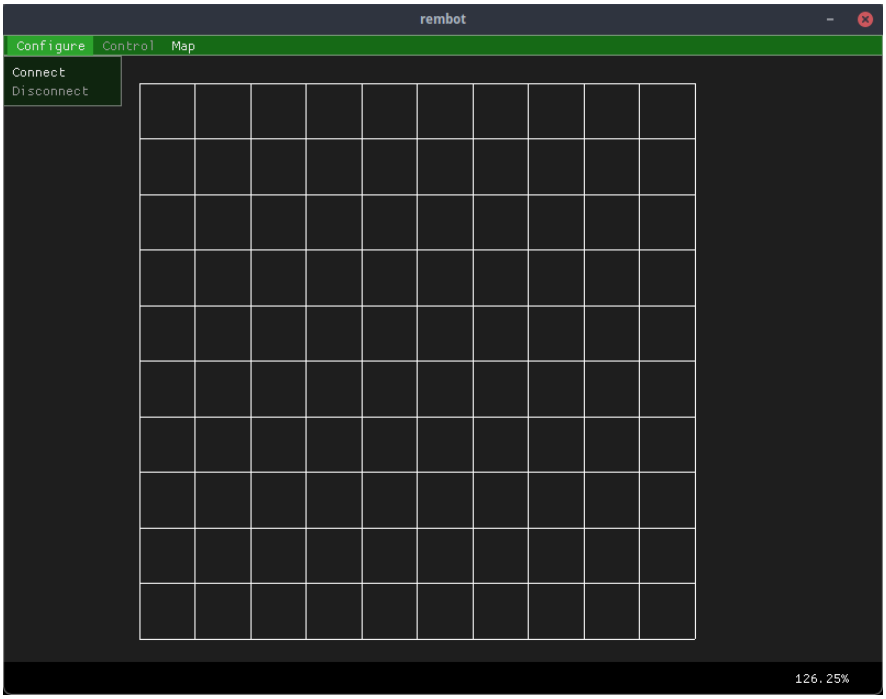

Рисунок 1. Меню "Configure"

Произойдет открытие нового окна для подключения.

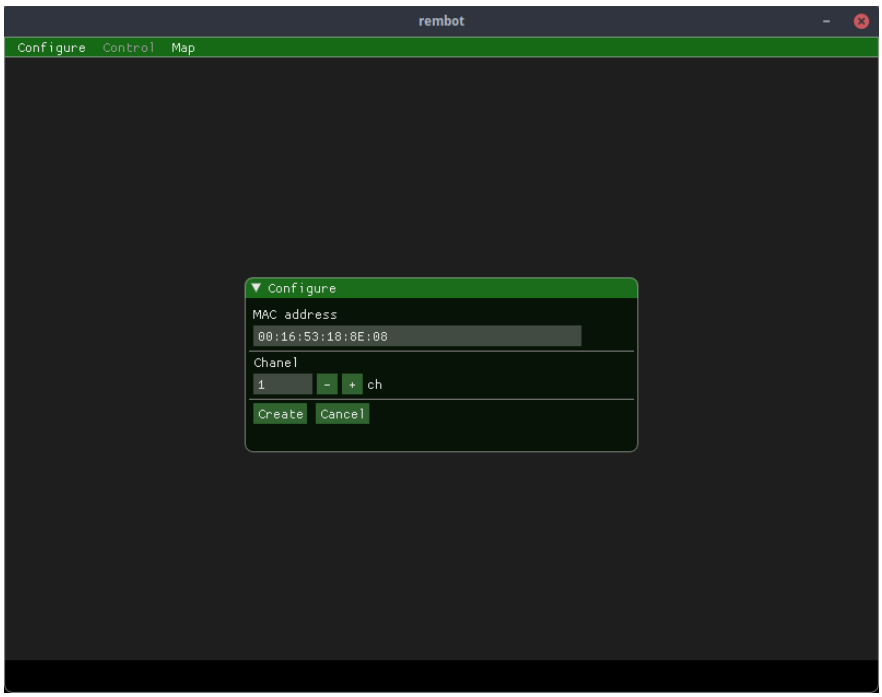

Рисунок 2. Окно настройки подключения "Configure"

Необходимые параметры для подключения:

- MAC адрес

- канал взаимодействия (1,2,3)

После ввода необходимых параметров, при нажатии на кнопку «Connect», произойдет попытка подключения к роботу.

В случае если появится необходимость в отключении приложения от робота, был предусмотрен пункт меню «Disconnect» в «Configure». При нажатии на него произойдет немедленное закрытия соединения с роботом.

2.3.2 Проектирование маршрута

Маршрут — список действий(команд), который необходимо выполнить поэтапно роботу.

В качестве условной единицы взаимодействия между графическим приложением и роботом был выбран 1см.

Для того чтобы спроектировать маршрут, необходимо создать карту. Создание карты происходит путем открытия пункта меню «Map» в верхней части панели и дальнейшим выбором подпункта меню «New map».

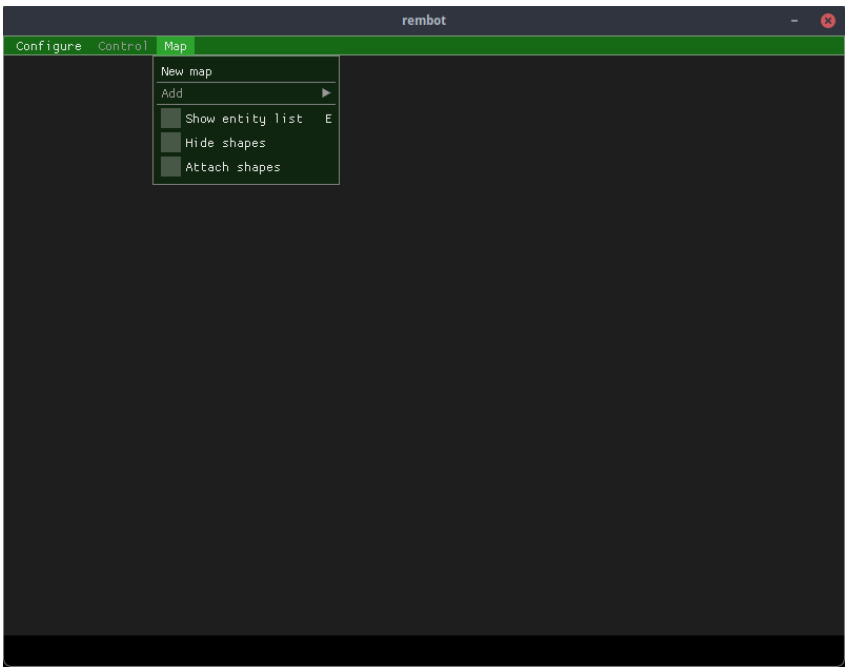

Рисунок 3. Меню "Map"

После выбора пункта меню, произойдет открытие окна создания новой карты, где будет необходимо заполнить параметры для нашей будущей карты: размер карты и размер блока карты.

- Размер карты — Количество блоков
- Размер блока карты — Размер блока вычисляемых в см.

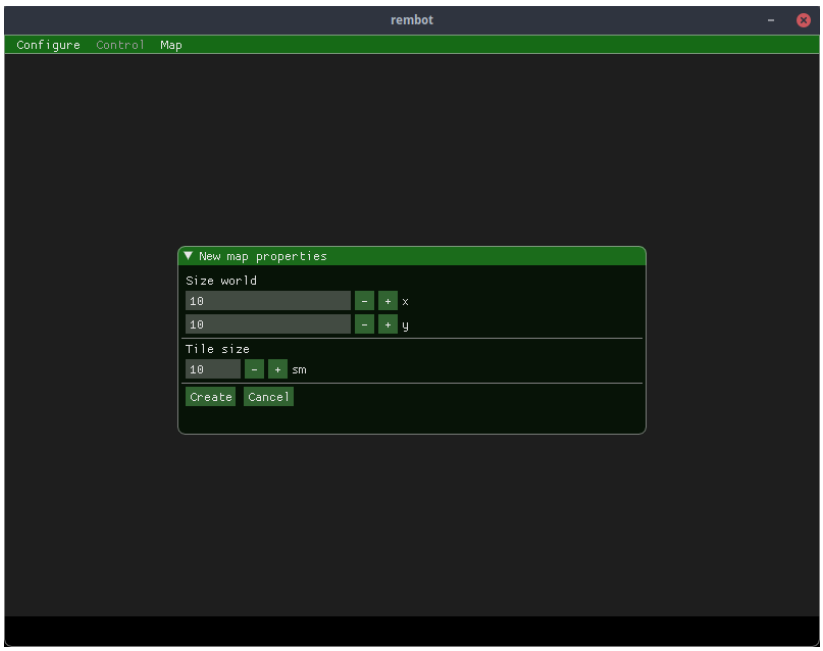

Рисунок 4. Окно создания новой карты "New map properties"

Данные параметры в совокупности представляют карту для выполнения команд. Например, если указать размер блока равным в 5 см и размер карты в 100 по x и 100 по y, мы получим карту размером в 5м.

После ввода данных, при нажатии на кнопку «Create», происходит создание сетки с заданными параметрами.

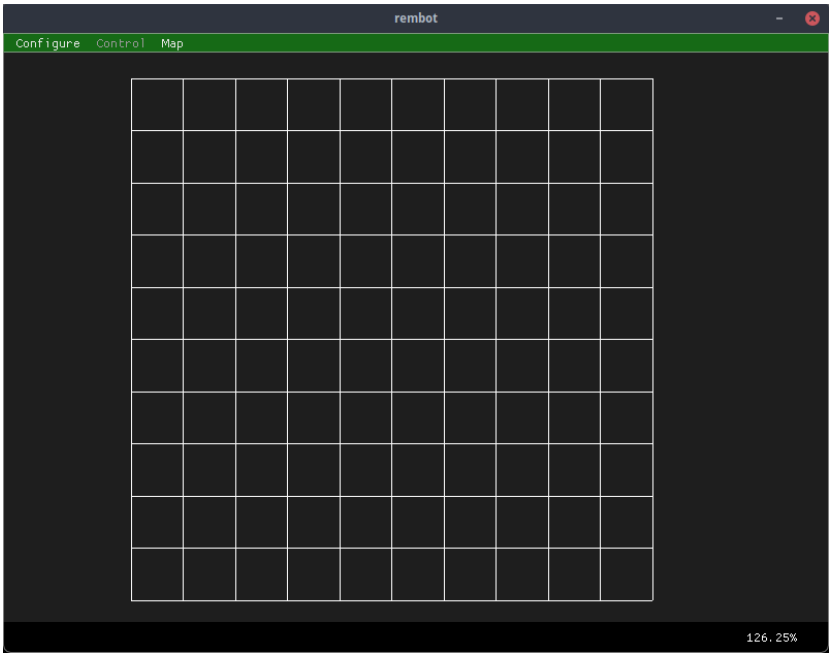

Рисунок 5. Карта 10x10

Для добавление нового маршрута необходимо перейти в пункт меню «Map», потом «Add» и выбрать «Patch».

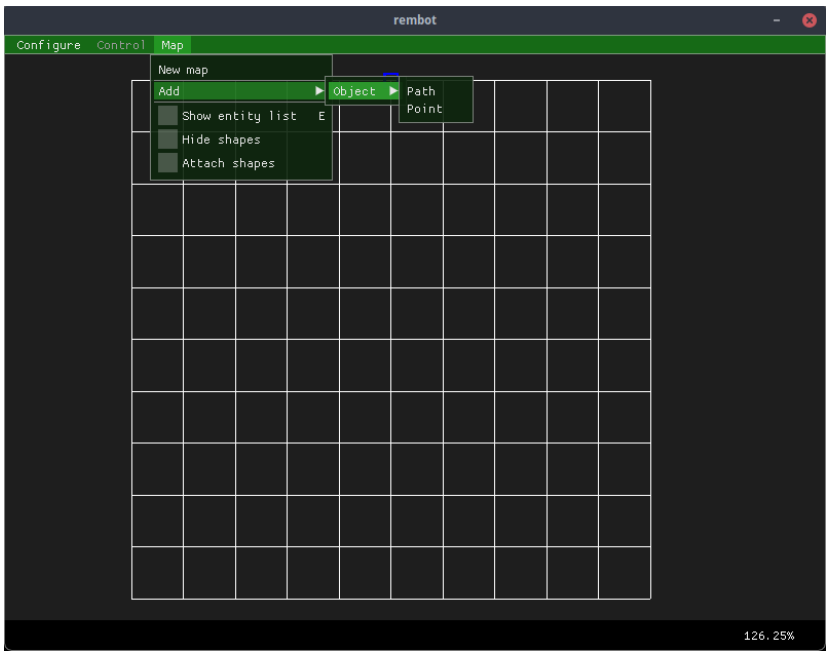

Рисунок 6. Создание маршрута

Путем выбора доступного блока, который подсвечивается синим квадратом и нажатием левой кнопки мыши, произойдет добавление команды для заданной координаты, которая потом будет преобразована с помощью формул в команды для робота.

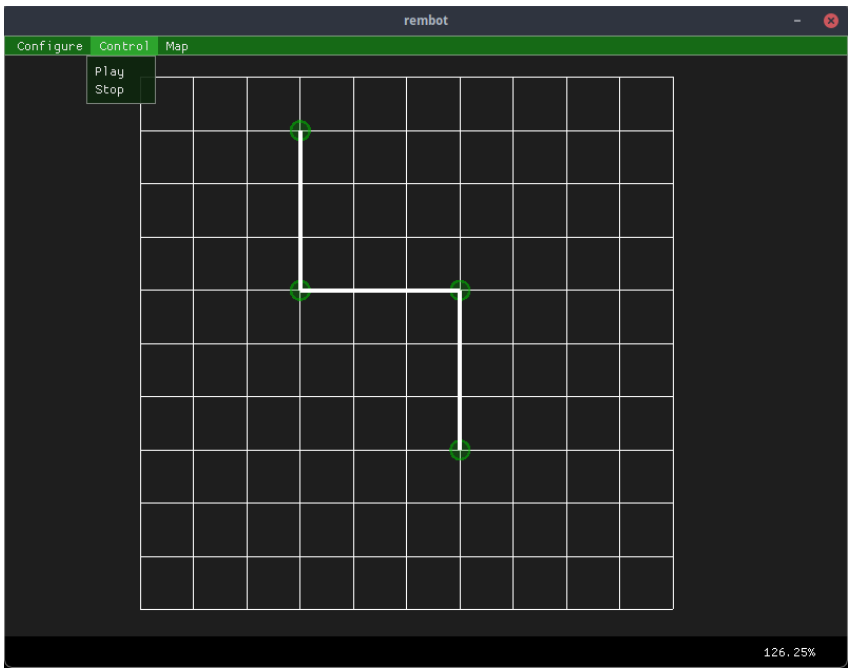

Рисунок 7. Пример маршрута движения робота

После добавление всех координат, при нажатии правой кнопки мыши, происходит добавление созданного маршрута в список доступных для запуска.

Для редактирования доступных маршрутов, необходимо перейти в верхнем меню в пункт «Map» и выбрать «Entity List». Откроется новое окно с списком доступных моделей на карте.

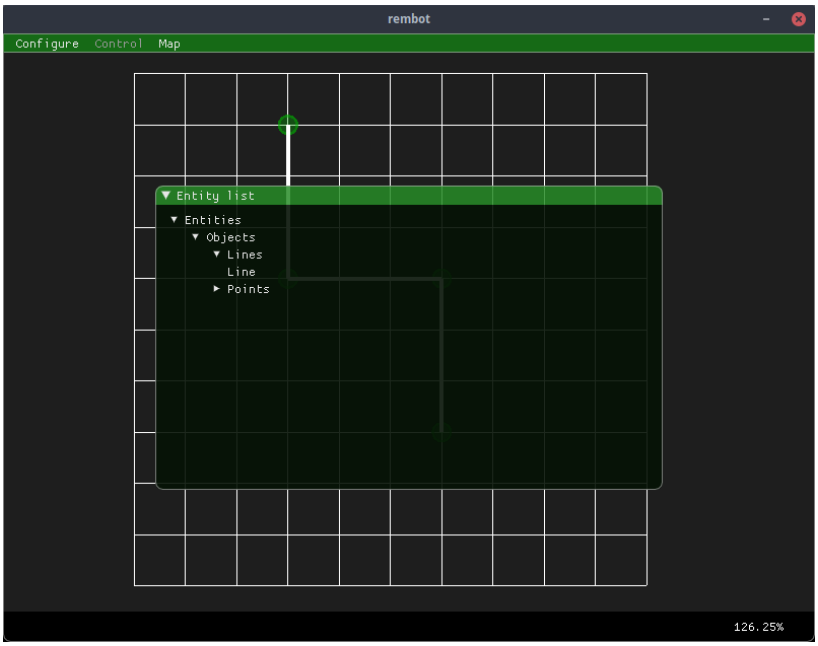

Рисунок 8. Окно выбора маршрута для редактирования

При нажатии элемент из списка произойдет открытие окна редактирования выбранной модели.

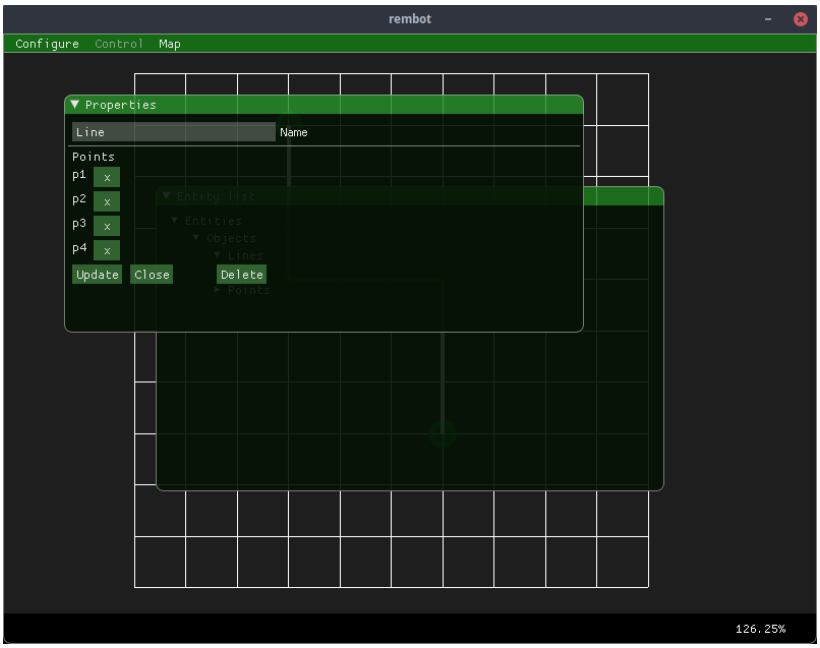

Рисунок 9. Окно редактирование маршрута

2.3.3 Отправка команд роботу

Для запуска выполнения маршрута необходимо активное подключение к роботу и наличие доступных маршрутов на карте.

В случае если происходит обрыв связи между графическим приложением и роботом (например закрытие приложения или выключение робота), графическое приложение переходит в состояние остановлен.

При открытии меню «Control» в верхнем меню, будет доступно два подпункта меню. Первый «Play» отвечает за запуск выполнения маршрута на роботе, путем передачи команд. Второй «Stop» отвечает за остановку выполнения маршрута.

Для запуска выполнения маршрута необходимо выбрать пункт меню «Play». Откроется новое окно с настройками запуска маршрута. В данном окне будет необходимо выбрать маршрут для выполнения. Кнопка «Play» запустит систему отправки команд.

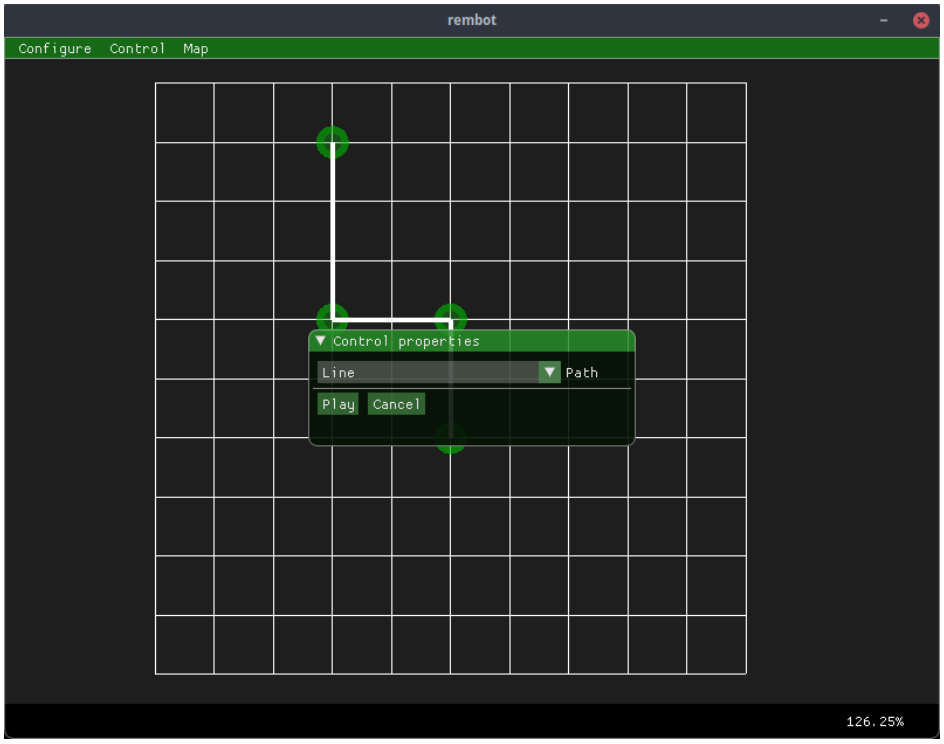

Рисунок 9. Окно выбора и запуска маршрута

3. Заключение

В данной работе, был разработан программный комплекс, состоящий из графического приложения и программного обеспечения робота, которое позволяет ему на основании интерфейса bluetooth принимать команды движения от графического приложения и отдавать результат выполнения команды обратно.

4. Список используемой литературы

1. Официальный сайт библиотеки Motor Control [Электронный ресурс]. Режим доступа: http://www.mindstorms.de/trac/wiki/MotorControl
2. Boost.Asio. Документация. [Электронный ресурс]. Режим доступа: https://www.boost.org/doc/libs/1_68_0/doc/html/boost_asio.html
3. OpenGL Tutorial. Уроки по OpenGL. [Электронный ресурс]. Режим доступа: http://www.opengl-tutorial.org/ru/
4. SFML 2.5 Tutorials. Уроки по SFML 2.5. [Электронный ресурс]. Режим доступа: https://www.sfml-dev.org/tutorials/2.5/
5. BLUEZ. Official Linux Bluetooth protocol stack. [Электронный ресурс]. Режим доступа: http://www.bluez.org/
6. IMGUI. GUI Library C++ [Электронный ресурс]. Режим доступа: https://github.com/ocornut/imgui

Приложение А (Код на роботе)

```
/*****************************************************************************/
/*                                  T Y P E S                                */
/*****************************************************************************/

typedef ubyte ReadBytesBuffer[64];

typedef enum {
      NONE = 0,
      UP = 1,
      DOWN = 2,
      LEFT = 3,
      RIGHT = 4,
      STOP = 12
} Command;

typedef enum {
      STATUS_NO = 0,
      STATUS_OK = 1
} StatusCommand;

/*****************************************************************************/
/*                               G L O B A L S                               */
/*****************************************************************************/

/** Buffer for reading from serial port */
ReadBytesBuffer g_btReadBuffer;

/** Current command */
Command command = NONE;

/*****************************************************************************/
/*                                 DEFINITION                                */
/*****************************************************************************/

StatusCommand runCommand(Command curCommand, ubyte length, ubyte size);

/*****************************************************************************/
/*                           B L U E T O O T H  I O                          */
/*****************************************************************************/

/** Checks for connected bluetooth status and reports negative results */
void BTcheckLinkConnected()
{
      if (nBTCurrentStreamIndex >= 0)
            return;  // An existing Bluetooth connection is present.

      //
      // Not connected. Audible notification and LCD error display
      //
      PlaySound(soundLowBuzz);
      PlaySound(soundLowBuzz);
      //  eraseDisplay();
      nxtDisplayCenteredTextLine(3, "Computer Not");
      nxtDisplayCenteredTextLine(4, "Connected");
      wait1Msec(3000);
      StopAllTasks();
}

/** Enables raw Bluetooth mode so that we can talk directly over the serial
port */
```

```
void BTenableRawMode()
{
	// Set Bluetooth to "raw mode".
	setBluetoothRawDataMode();
	wait1Msec(50);

	// While the Bluecore is still NOT in raw mode (bBTRawMode == false);
	while (!bBTRawMode)
	{
		setBluetoothRawDataMode();
		// Wait for Bluecore to enter raw data mode.
		wait1Msec(5);
	}
}

/** Reads the desired number of bytes from the raw bluetooth feed */
void BTreadBytes(ReadBytesBuffer& buffer, int bytesToRead)
{
	//int bytesRead = 0;
	ubyte localBuf[1];

	//while (bytesRead < bytesToRead)
	for (int i = 0; i < bytesToRead; ++i)
	{
		while (0 == nxtReadRawBluetooth(localBuf, 1))
		{
			wait1Msec(10);
		}

		// Store value in the given buffe
		buffer[i] = localBuf[0];
	}
}

void reciveCommand(ubyte uCommand) {

	const int kHexDigitsPerLine = 8;
	char bufferBytes[kHexDigitsPerLine];

	for (int i = 0; i < kHexDigitsPerLine; ++i)
		bufferBytes[i] = 0;

	command = (Command)uCommand;
	switch(command) {
	case UP :
	case DOWN :
	case LEFT :
	case RIGHT : {
			BTreadBytes((ReadBytesBuffer)bufferBytes, 2);
			Command curCommand = command;
			runCommand(curCommand, bufferBytes[0],bufferBytes[1]);
		} break;
	case STOP : {

		} break;
	default:
		command = NONE;
		break;
	}
}

task readRawData()
{
	ubyte localBuf[1];
```

```
	while (true)
	{
		BTreadBytes((ReadBytesBuffer)localBuf, 1);
		reciveCommand(localBuf[0]);
	}
}

StatusCommand upCommand(ubyte* buf,  ubyte length, ubyte size) {
		nMotorEncoder(motorA) = 0;
	nMotorEncoder(motorC) = 0;
	while (nMotorEncoder[motorA] < length*size*17.6)
	{
		int nNumbBytesRead = nxtReadRawBluetooth(buf, 1);
		if (nNumbBytesRead > 0)
		{
			motor[motorA] = 0;
			motor[motorC] = 0;
			reciveCommand(buf[0]);
			return STATUS_NO;
			} else {
			motor[motorA] = 30;
			motor[motorC] = 30;
		}
	}
	motor[motorA] = 0;
	motor[motorC] = 0;
	if (nMotorEncoder[motorA] >= length*size*17.6) {
		return STATUS_OK;
	}
	return STATUS_NO;
}

StatusCommand downCommand(ubyte* buf,  ubyte length, ubyte size) {
		nMotorEncoder(motorA) = 0;
	nMotorEncoder(motorC) = 0;
	while (-nMotorEncoder[motorA] < length*size*17.6)
	{
		int nNumbBytesRead = nxtReadRawBluetooth(buf, 1);
		if (nNumbBytesRead > 0)
		{
			motor[motorA] = 0;
			motor[motorC] = 0;
			reciveCommand(buf[0]);
			return STATUS_NO;
			} else {
			motor[motorA] = -30;
			motor[motorC] = -30;
		}
	}
	motor[motorA] = 0;
	motor[motorC] = 0;
	if (-nMotorEncoder[motorA] >= length*size*17.6) {
		return STATUS_OK;
	}
	return STATUS_NO;
}

StatusCommand leftCommand(ubyte* buf,  ubyte length, ubyte size) {
		nMotorEncoder(motorA) = 0;
	nMotorEncoder(motorC) = 0;
	while (nMotorEncoder[motorA] < 395.829)
	{
		int nNumbBytesRead = nxtReadRawBluetooth(buf, 1);
```

```
            if (nNumbBytesRead > 0)
            {
                  motor[motorA] = 0;
                  reciveCommand(buf[0]);
                  return STATUS_NO;
                  break;
                  } else {
                  motor[motorA] = 30;
            }
      }
      motor[motorA] = 0;
      if (nMotorEncoder[motorA] >= 395.829) {
            return STATUS_OK;
      }
      return STATUS_NO;
}

StatusCommand rightCommand(ubyte* buf,  ubyte length, ubyte size) {
            nMotorEncoder(motorA) = 0;
      nMotorEncoder(motorC) = 0;
      while (nMotorEncoder[motorC] < 395.829)
      {
            int nNumbBytesRead = nxtReadRawBluetooth(buf, 1);
            if (nNumbBytesRead > 0)
            {
                  motor[motorC] = 0;
                  reciveCommand(buf[0]);
                  return STATUS_NO;
                  break;
                  } else {
                  motor[motorC] = 30;
            }
      }
      motor[motorC] = 0;
      if (nMotorEncoder[motorC] >= 395.829) {
            return STATUS_OK;
      }
      return STATUS_NO;
}

/*************************************************************************/
/*                C O M M S   P R O T O C O L  E N G I N E               */
/*************************************************************************/

StatusCommand runCommand(Command curCommand, ubyte length, ubyte size) {
      ubyte localBuf[1];
      StatusCommand status = STATUS_NO;

      switch(curCommand) {
      case UP : {
                  status = upCommand(localBuf, length, size);
            } break;
      case DOWN : {
                  status = downCommand(localBuf, length, size);
            } break;
      case LEFT: {
                  status = leftCommand(localBuf, length, size);
                  break;
            }
      case RIGHT: {
                  status = rightCommand(localBuf, length, size);
                  break;
            }
      }
```

```
	ubyte BytesToSend[1];
	BytesToSend[0] = (int)status;
	nxtWriteRawBluetooth(BytesToSend, 1);
	return status;
}

/*****************************************************************************/

task main()
{

	bBTHasProgressSounds = true;

	bBTSkipPswdPrompt = true;

	BTcheckLinkConnected();
	BTenableRawMode();

	eraseDisplay();
	bNxtLCDStatusDisplay = true;

	StartTask(readRawData);

	while(true) {
		nxtDisplayTextLine(1, "Bluetooth Enabled");
		nxtDisplayBigStringAt(0, 31, "Cmd Stat");
		nxtDisplayBigStringAt(0, 15, "%02X", (int)command);
		wait1Msec(100);
	}

	return;
}
```