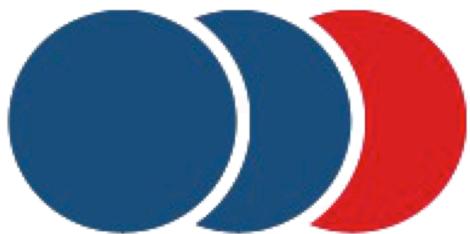


Установка Web - фреймворка N2O на Raspberry Pi3



N2O

So, let's begin...

Описание пошаговой установки фреймворка на следующей конфигурации оборудования:

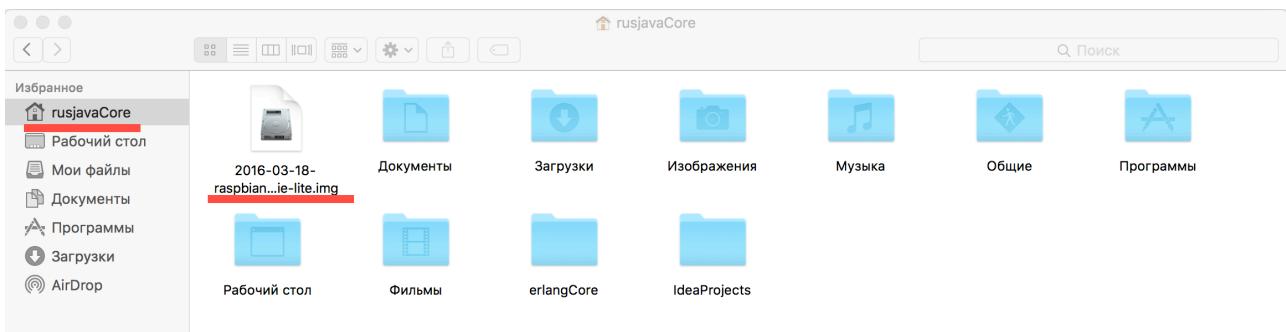
1. iMac (OS X El Capitan – 10.11.4)
2. Raspberry Pi3
3. Micro SD (Sony SR-32UY, class 10, 70 MB/s, 32 GB)

Установку начнём со скачивания, разархивирования и записи образа 'Raspbian Jessie Lite'

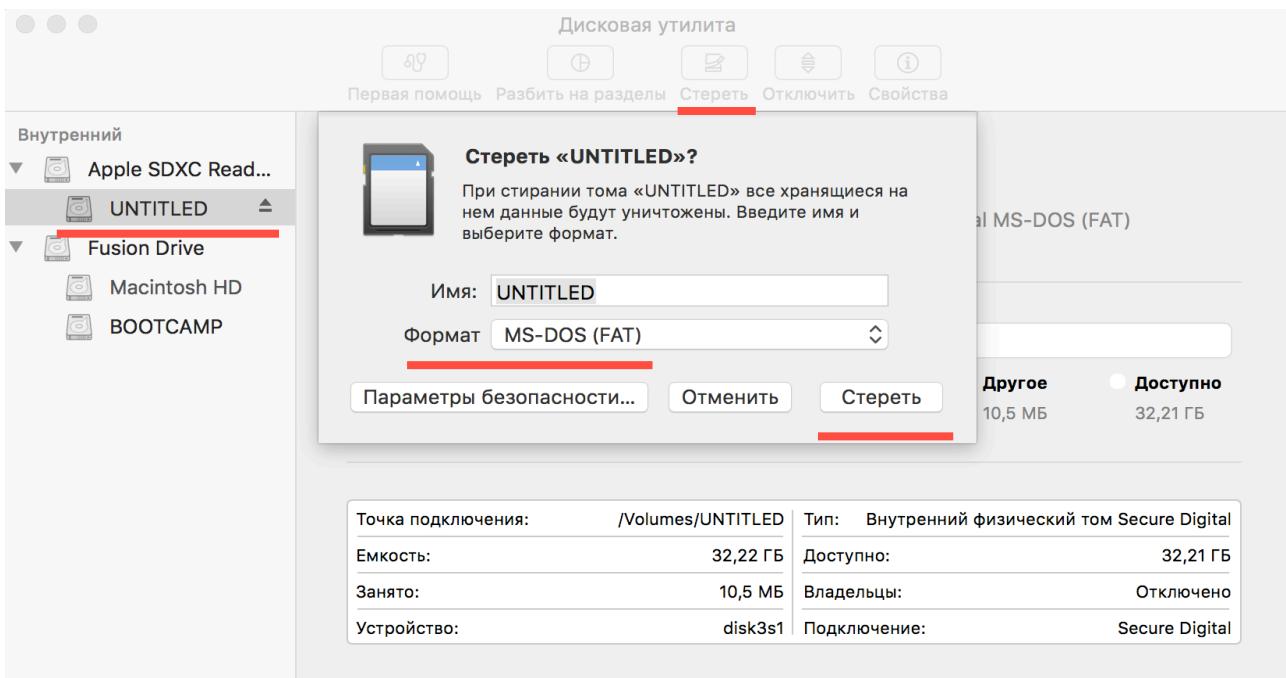
1. Скачиваем актуальный на данный момент образ дистрибутива .zip с официального сайта:

[https://downloads.raspberrypi.org/raspbian\\_lite\\_latest](https://downloads.raspberrypi.org/raspbian_lite_latest) Release date: 2016-03-18

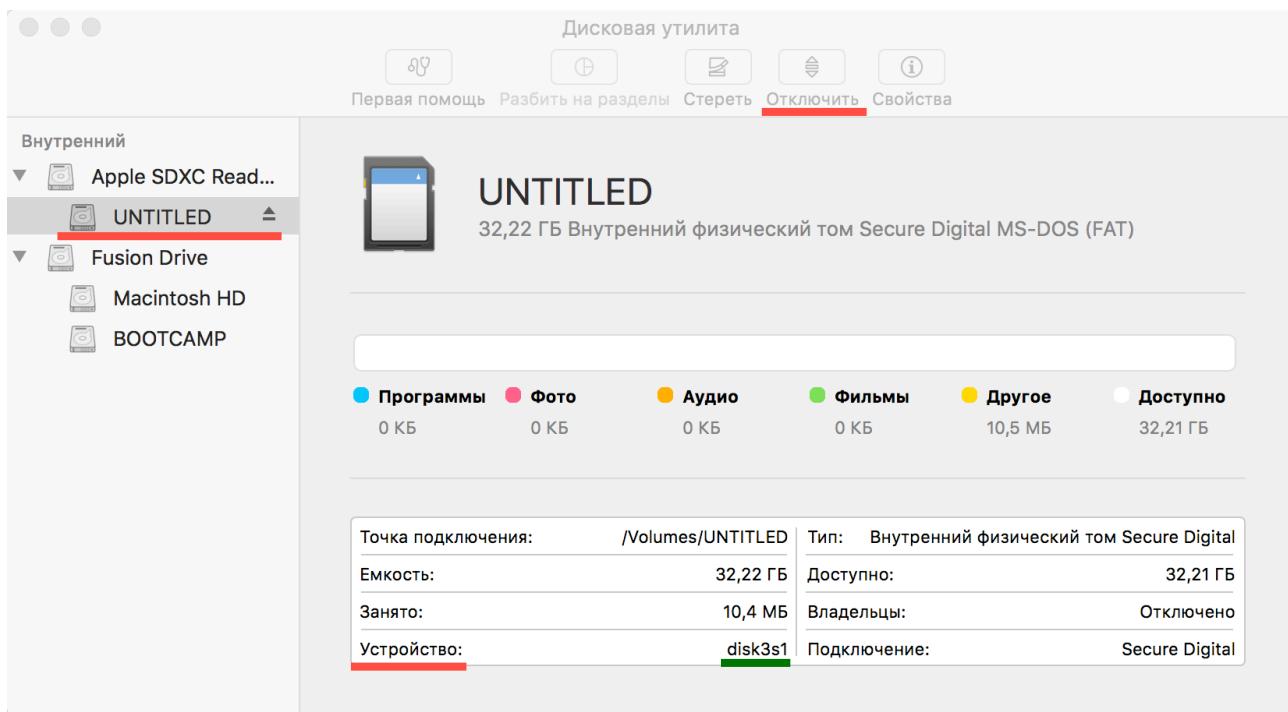
2. Разархивируем, полученный из архива образ .img и закидываем в корневой каталог



3. Подключаем Micro SD к компьютеру, запускаем дисковую утилиту, выбираем Micro SD в списке, нажимаем Стереть, выбираем Формат MS-DOS(FAT) и повторно нажимаем Стереть



4. После завершения форматирования снова выбираем **Micro SD** и нажимаем **Отключить**, а так же запоминаем номер Micro SD в системе, у нас в примере это **Устройство: disk3s1**

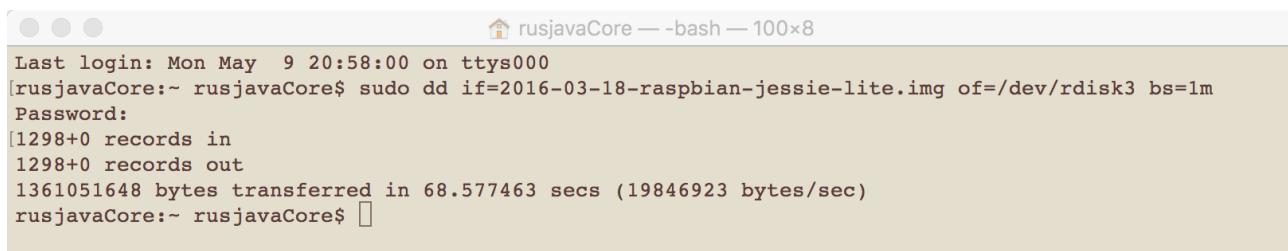


5. Запускаем терминал на компьютере и вводим следующую команду для записи образа:

```
$ sudo dd if=2016-03-18-raspbian-jessie-lite.img of=/dev/rdisk3 bs=1m
```

**2016-03-18-raspbian-jessie-lite.img** - название записываемого на Micro SD образа OS  
**/dev/rdisk3** - путь к Micro SD в системе, где **3** номер устройства (**disk3s1**)

По окончании процесса записи окошко терминала должно выглядеть примерно как на скриншоте ниже, а название Micro SD карты в **Disk - утилите** смениться с **UNTITLED** на **boot**



6. Извлекаем Micro SD из слота для чтения SD-карт компьютера и вставляем в **Raspberry Pi3**, подключаем сетевой кабель, подаём питание.

На этом подготовка завершена, переходим к настройке **Raspberry Pi3**.

7. Теперь нам необходимо определить какой ip - адрес получил **RaspberryPi3**, вы можете посмотреть в списке **DHCP** вашего роутера, либо на экране подключенном через **HDMI** к **RaspberryPi3**, мы будем определять с помощью ранее установленной через **Homebrew** утилиты **nmap**, для этого пишем в терминале следующую команду:

```
$ nmap 192.168.1.0/24 -p22
```

192.168.1.0/24 - диапазон нашей подсети  
-p22 — указание номера порта, в нашем случае искомый порт **22 (ssh)**

Получаем список найденных устройств в локальной сети с открытым **22-ым** портом (**ssh**), методом проб :) выясняем что ip - адрес нашего устройства **RaspberryPi3** — **192.168.1.21**

```
rusjavaCore — bash — 100x37
Last login: Mon May  9 20:58:07 on ttys000
[rusjavaCore:~ rusjavaCore$ nmap 192.168.1.0/24 -p22

Starting Nmap 6.47 ( http://nmap.org ) at 2016-05-09 21:11 MSK
Nmap scan report for 192.168.1.1
Host is up (0.0023s latency).
PORT      STATE      SERVICE
22/tcp    closed    ssh

Nmap scan report for 192.168.1.2
Host is up (0.0052s latency).
PORT      STATE      SERVICE
22/tcp    closed    ssh

Nmap scan report for 192.168.1.12
Host is up (0.0035s latency).
PORT      STATE      SERVICE
22/tcp    filtered  ssh

Nmap scan report for 192.168.1.20
Host is up (0.096s latency).
PORT      STATE      SERVICE
22/tcp    closed    ssh

Nmap scan report for 192.168.1.21
Host is up (0.0032s latency).
PORT      STATE      SERVICE
22/tcp    open      ssh

Nmap scan report for 192.168.1.200
Host is up (0.0058s latency).
PORT      STATE      SERVICE
22/tcp    open      ssh

Nmap done: 256 IP addresses (6 hosts up) scanned in 4.81 seconds
rusjavaCore:~ rusjavaCore$ 
```

8. Подключаемся к **RaspberryPi3**, для этого пишем в терминале следующую команду:

```
$ ssh pi@192.168.1.21
```

Получаем приглашение, для подтверждения согласия пишем **yes** и нажимаем **Enter**

```
rusjavaCore — ssh pi@192.168.1.21 — 100x6
Last login: Mon May  9 21:51:43 on ttys000
[rusjavaCore:~ rusjavaCore$ ssh pi@192.168.1.21
The authenticity of host '192.168.1.21 (192.168.1.21)' can't be established.
ECDSA key fingerprint is SHA256:z6Bc8KJD9pp0JT4LMKYj+BxTdNWmoYxh3KwfKM5/b3A.
Are you sure you want to continue connecting (yes/no)? yes
```

Получаем запрос на ввод пароля, пароль по умолчанию: **raspberry**

```
rusjavaCore — ssh pi@192.168.1.21 — 100x8
Last login: Mon May  9 21:51:43 on ttys000
[rusjavaCore:~ rusjavaCore$ ssh pi@192.168.1.21
The authenticity of host '192.168.1.21 (192.168.1.21)' can't be established.
ECDSA key fingerprint is SHA256:z6Bc8KJD9pp0JT4LMKYj+BxTdNWmoYxh3KwfKM5/b3A.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.21' (ECDSA) to the list of known hosts.
pi@192.168.1.21's password: [key]
```

Если вдруг в терминале появляется сообщение что сохраненный этому ip - адресу ключ не совпадает с новым ключом, то вам поможет следующая команда удаления предидущего ключа:

```
$ ssh-keygen -R 192.168.1.21
```

И так мы успешно авторизовались, о чём нам свидетельствует примерно следующее сообщение в нашем терминале:

```
rusjavaCore — pi@raspberrypi: ~ — ssh pi@192.168.1.21 — 100x16
Last login: Mon May  9 21:51:43 on ttys000
[rusjavaCore:~ rusjavaCore$ ssh pi@192.168.1.21
The authenticity of host '192.168.1.21 (192.168.1.21)' can't be established.
ECDSA key fingerprint is SHA256:z6Bc8KJD9pp0JT4LMKYj+BxTdNWmoYxh3KwfKM5/b3A.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.21' (ECDSA) to the list of known hosts.
pi@192.168.1.21's password:

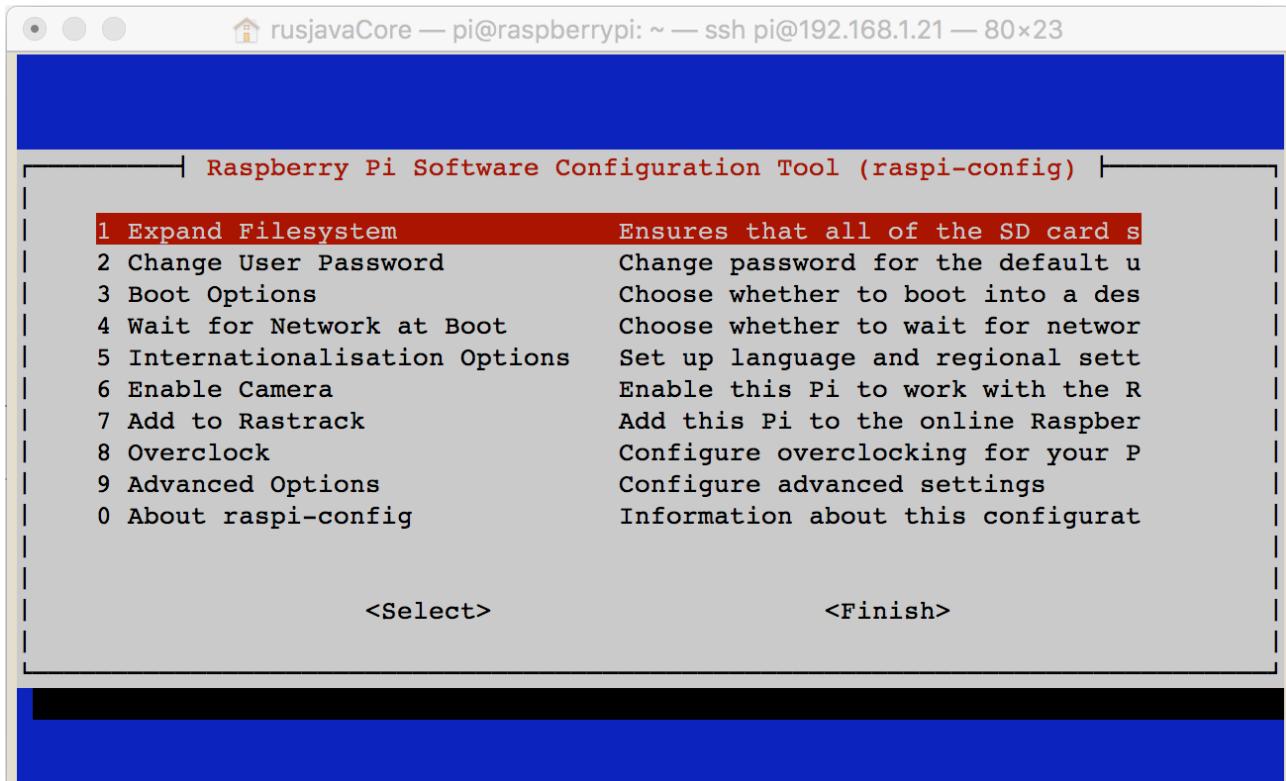
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi:~ $
```

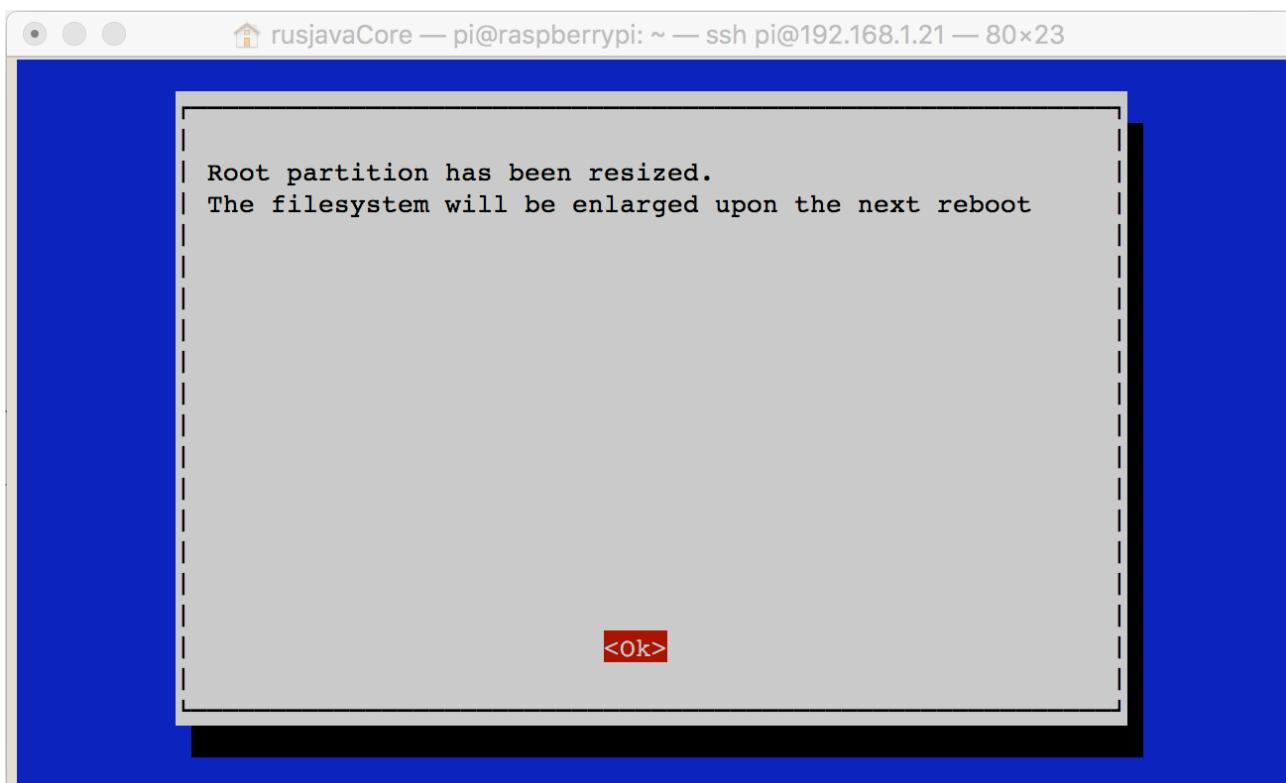
9. Теперь нам необходимо расширить наш образ дистрибутива на весь объем Micro SD, делается это следующим образом, вводим команду:

```
$ sudo raspi-config
```

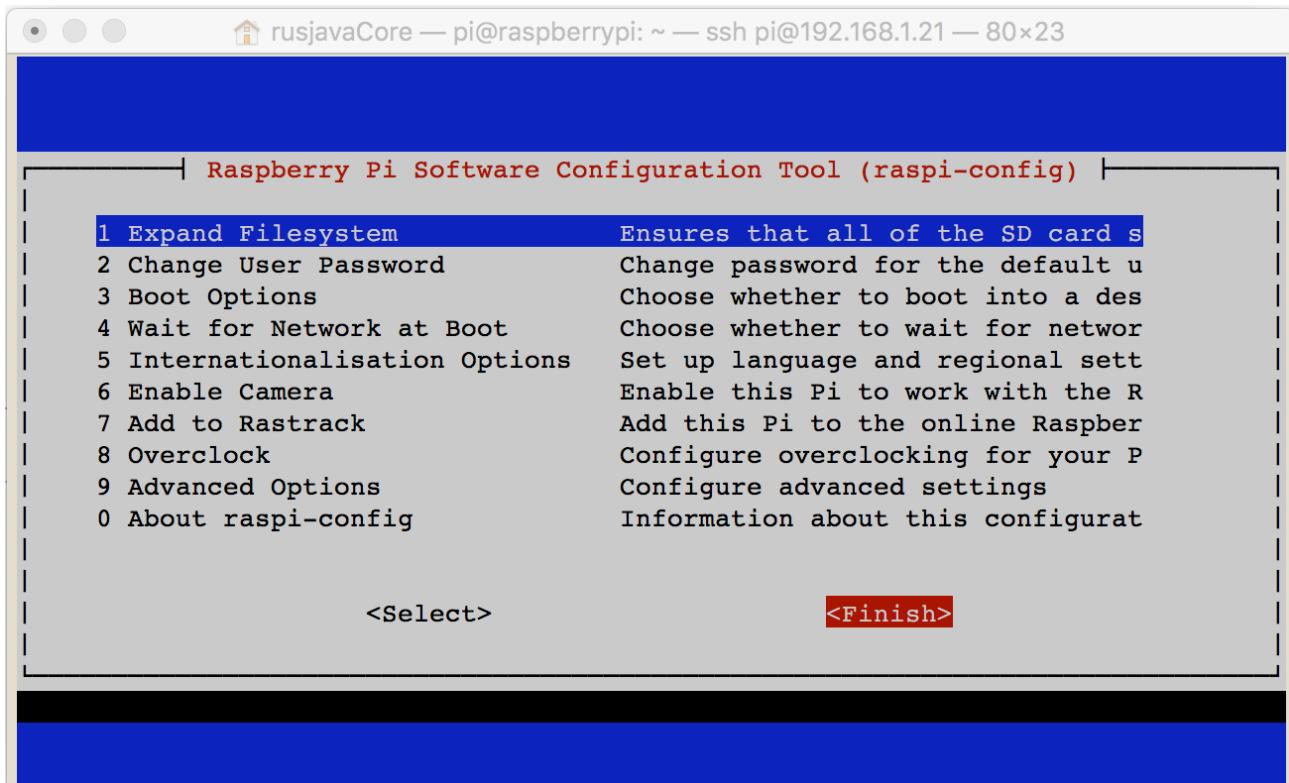
В появившемся меню конфигурации, выбираем пункт **1. Expand Filesystem** нажимаем **Enter**



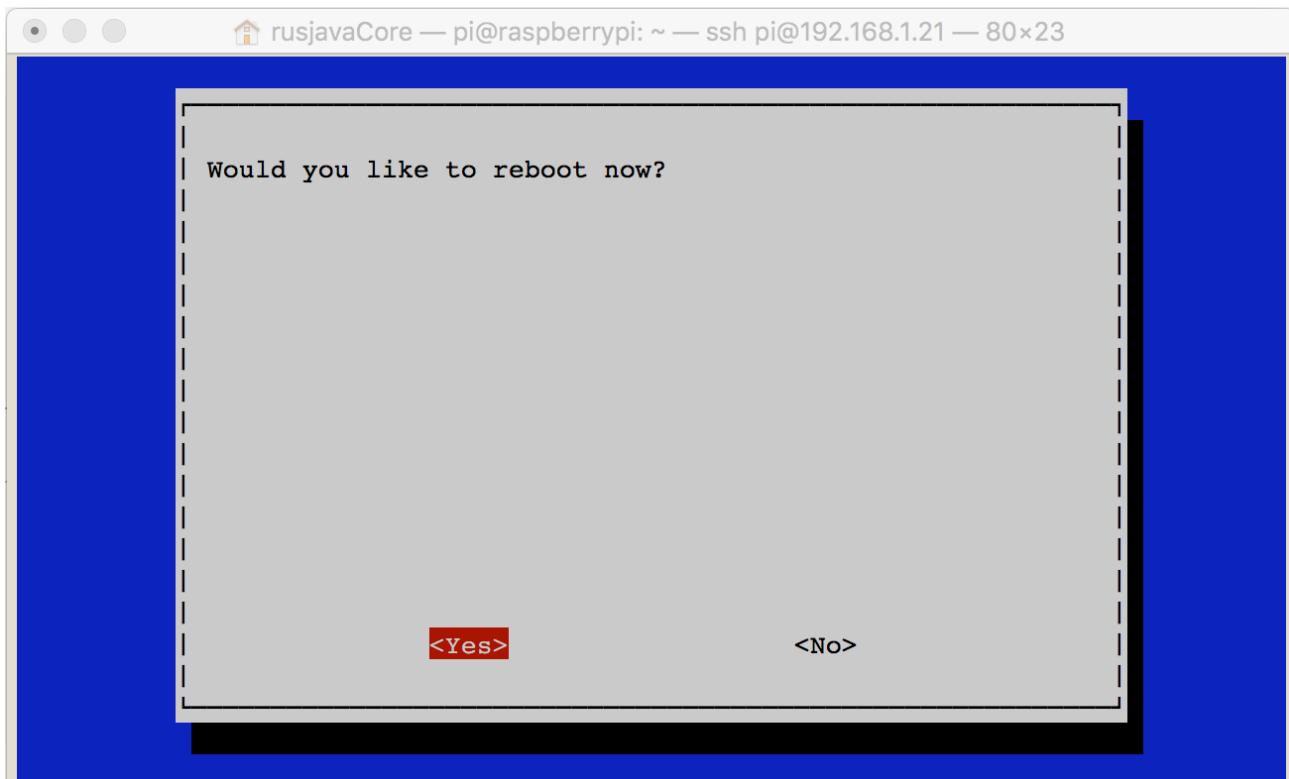
видим надпись **<Ok>** подтверждаем нажатием на кнопку **Enter**



Далее кнопкой табуляции на клавиатуре выбираем пункт меню **<Finish>** и нажимаем **Enter**



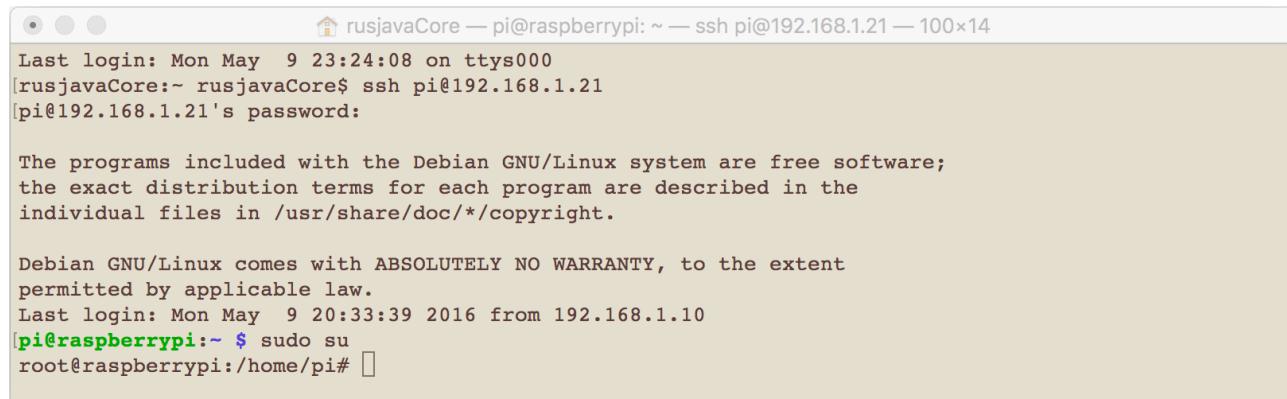
на предложение перезагрузить отвечаем выбором пункта **<Yes>** и нажатием на кнопку **Enter**



10. После перезагрузки, повторно авторизуемся, далее, для того что бы мы могли выполнять команды в терминале от имени суперпользователя **root** нам нужно выполнить следующую команду:

```
$ sudo su
```

В терминале видим следующее:



rusjavaCore — pi@raspberrypi: ~ — ssh pi@192.168.1.21 — 100x14  
Last login: Mon May 9 23:24:08 on ttys000  
[rusjavaCore:~ rusjavaCore\$ ssh pi@192.168.1.21  
[pi@192.168.1.21's password:  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*/\*copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon May 9 20:33:39 2016 from 192.168.1.10  
[pi@raspberrypi:~ \$ sudo su  
root@raspberrypi:/home/pi# ]

11. Обновляем дистрибутив операционной системы связкой следующих команд:

```
$ apt-get update && apt-get dist-upgrade --yes
```

12. Устанавливаем зависимости, скачиваем исходники **VM Erlang**, компилируем:

**A:** Устанавливаем виртуальный пакет **ncurses-dev** и библиотеку **libssl-dev**

```
$ apt-get install ncurses-dev --yes && apt-get install libssl-dev --yes
```

**B:** Скачиваем архив с исходниками **VM Erlang** с помощью утилиты **wget**

```
$ wget http://www.erlang.org/download/otp_src_18.3.tar.gz
```

**C:** Разархивируем

```
$ tar -xzvf otp_src_18.3.tar.gz
```

D: Переходим в папку с исходниками в которую произвели разархивацию

```
$ cd otp_src_18.3/
```

E: Запускаем скрипт **configure** для проверки наличия всех зависимостей

```
$ ./configure
```

F: Запускаем утилиту **make** для компиляции исходного кода в объектные файлы, поскольку процесс не очень быстрый, придётся подождать какое то время до окончания выполнения

```
$ make
```

G: Запускаем процесс сборки которая производит установку файлов в 'нужные места'

```
$ make install
```

H: По окончании установки переходим в корневой каталог

```
$ cd ..
```

I: Удаляем уже не нужный каталог и архив с исходниками

```
$ rm -R otp_src_18.3/ && rm -R otp_src_18.3.tar.gz
```

J: Устанавливаем утилиту **inotify-tools**

```
$ apt-get install inotify-tools --yes
```

I. Устанавливаем утилиту **Git** для того что бы клонировать репозиторий **N2O**

```
$ apt-get install git --yes
```

### 13. Установка фреймворк N2O

#### I. Клонируем репозиторий N2O

```
$ git clone git://github.com/synrc/n2o
```

#### II. Переходим в папку с примером

```
$ cd n2o/samples
```

#### III. Запускаем mad

```
$ ./mad deps compile plan repl
```

#### Параметры mad:

- `deps` – получить зависимости
- `compile` – скомпилировать приложение
- `plan` – создать план запуска
- `repl` – запустить консоль

#### IV. Далее набираем в адресной строке браузера следующее:

```
http://192.168.1.21:8000
```

#### V: Готово!

