Instruction Manual to Setup Interface of different Sensors/Peripherals with Raspberry pi 3B/3B+ Computing Unit to Build RC Autonomous Car 1/10 Scale Model

Prepared By:

Document Owner	**Project**
Rohit Nagesh Gawas	**RC Autonomous Driving Car**

Report Version Control:

Version	Date	Author
1.0	27-08-2019	Rohit N. Gawas

Contents

1 . Introduction

1.1 Purpose
1.2 Goal
1.3 Overview

2. Sensor Configuration

3. Computing Units

4. How to Setup Raspberry Pi 3B+ with Linux based on Raspbian distribution and ROS-Kinetic

5. How to install ROS Kinetic on Raspberry Pi 3B(Ubuntu Mate)-Fresh Installation.

6. Master and Slave Configuration

7. Setting up the Sensors

7.1 Setting up the RPLidar A1 Sensor on the Rasp Pi

7.2 Setting up the Neato XV-11 Lidar Sensor on the Raspberry Pi

7.3 Setting up the Raspberry Pi Camera in ROS Kinetic

7.4 Setting up the Communication with IMU-GY85 Sensor

7.5 Setting up of Movidius Neural Compute Stick with Raspberry Pi 3B For Object detection

1. Introduction

An **autonomous car**, also known as a **robot car**, or **driverless car**, is a vehicle that is capable of sensing its environment and moving with little or no human input.

Autonomous cars combine a variety of sensors to perceive their surroundings, such as radar, Lidar, sonar, Cameras , GPS, odometry and inertial measurement units. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage.

1.1 Purpose : The Instruction Manual documents over Sensor Configuration Used to build RC Autonomous Car Model, How to setup the communication with different Sensor using Raspberry Pi Computer and Arduino as Controller.

1.2 Goal : The aim of this Instruction Manual is to guide on how to setup communication with different sensor's with Raspberry Pi computing unit Arduino as controller in order to build an RC Autonomous Car by integrating different sensors , Computing Unit and

1.3 Overview:

The report is organized as follows:

- Section 2 Describes the Sensor Configuration – Sensor's used in the RC Autonomous Car.
- Section 3 Describes the Computing Unit Used.
- Section 4 Describes on How to Setup Raspberry Pi 3B+ with Linux based on Raspbian and ROS-Kinetic
- Section 5 Describes on How to install ROS Kinetic on Raspberry Pi 3B(Ubuntu Mate) – Fresh Installation.
- Section 6 Describes Master and Slave Configuration
- Section 7 Describes on Setting up the Sensors

2. Sensor Configuration

2.1 LIDAR Selection

The Autonomous RC Car Model is to be used for academic purpose, Therefore we decided to use RPLIDAR A1 and Neato XV-11 , which are cheap , easy to configure in ROS and serves our purpose. Below (Figure 1) gives the comparison between RPLIDAR A1 and Neato XV-11 Lidar.

Type	RPLIDAR A1 360' Lidar Scanner	Neato XV 360' Lidar Scanner
Figure		
Distance Range(m)	0.15 – 12	0.15 - 6
Angular Range (Degree)	360'	360'
Angular Resolution (Degree)	1'	1'
Accuracy	< 1.5m: 0.5mm	< 1.5m: 0.5mm
Scan Rate(Hz)	5 HZ	5 HZ
Scan Method	Evenly Mapped	Evenly Mapped
Application	• To get the 2D Measurement of the world around the robot. • Obstacle detection • Mapping • Localization • Odometry	• To get the 2D Measurement of the world around the robot. • Obstacle detection • Mapping • Localization • Odometry

Figure 1. Comparison between RPLIDAR A1 and Neato XV-11.

2.2 Camera Selection

To cover more field of view in-front of RC Car, Raspberry pi camera module with 160 degree FOV was selected . Also , Rasp cameras are highly compatible with raspberry pi and processing time is also less. Therefore we decided to choose Raspberry pi Camera module . For other tasks like video streaming , Logitech C920 Webcam is selected .

Below (Figure 2) is the chart that compares two different Camera module we have considered.

Type	Raspberry Pi Camera w/fish eye lens	Logitech C920 Webcam
Figure		
Resolution	5 Megapixel	Full HD
Viewing Angle Degrees	160’	80’
Frame-Rate	1080p@30fps 720p@60fps	1080p@30fps
Application	Suitable for Monitoring low speed traffic moving objects	Suitable for Monitoring High Speed traffic moving objects Because of its FHD Display
Performance	Compatible device and Directly can be interfaced with Raspberry Pi Computer. Hence Good Real time Image Processing Performance	Non Compatible Device & Should be interface via USB. Therefore, Real time image processing Performance is low. Also, With FHD Video recording . Direct Real time Image processing on Raspberry pi computer will be slow.
Application	• Object detection • Indoor Vision SLAM	• Video Streaming

Figure 2. Comparison between Raspberry Pi Camera module & Logitech C920 Webcam

2.3 IMU

For measuring the heading direction and orientation angle of the RC Car on the 2D Surface , GY85 9 DOF IMU Sensor is selected.

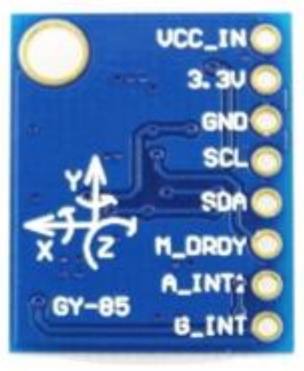


Figure 3. GY85 9DOF IMU Sensor

2.4 Intel Movidius Neural Stick

Object classification task on Raspberry pi as a stand-alone cannot be performed faster. To accelerate the Object Classification task using Deep Learning , Intel Movidius Neural Stick is selected. It supports TensorFlow & Caffe frameworks.

Figure 4. Intel Movidius Neural Stick

3. Computing Units

In order to Process the data from all the different sensors , We decided to use one dedicated computing unit Raspberry Pi-3B+ Model. One dedicated Arduino Mega board to control the Servo and the ESC(Electronic stability control) . One dedicated Arduino Nano board to read the data from the IMU Sensor , process and communicate it to the Raspberry pi .

3.1 Raspberry Pi-3 B+

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range. It has Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz , 1GB LPDDR2 SDRAM2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0Extended 40-pin GPIO header, 4 USB 2.0 ports , CSI Camera Port , DSI Display Port , Micro SD Port for loading your operating system and storing data and 5V/2.5A DC power input.

Type	Raspberry Pi 3 Model B
SoC	Broadcom BCM2837B0
CPU	Quad Cortex A53 @ 1.2GHz
Instruction set	ARMv8-A
GPU	400MHz Video Core IV
RAM	1GB SDRAM
Storage	Micro – SD
Ethernet(Gigabyte)	10 /100
Wireless	802.11n / Bluetooth 4.0
Video output	HDMI / Composite
Audio output	HDMI / Headphone
GPIO	40

Figure 5: Hardware Spec Raspberry Pi 3 Model B+

The computing Unit will run on Ubuntu OS & ROS. Both software components are run from an SD card. This is the main unit on which we will process “Perception” & “Planning” Software Modules which are the most important Software modules in Autonomous Car Design.

3.2. Arduino Mega

The Arduino Mega is a microcontroller board based on the ATmega1280. The controller will be serially interfaced with Raspberry Pi Computing Unit and will be used to control the Servo and the ESC .

Type	Arduino Mega 2560
Processor	ATmega 2560
Clock Speed	16 MHz
Flash Memory (kilo Byte)	256
EEPROM (kilo Byte)	4
SRAM (kilo Byte)	8
Voltage Level	5V
Digital I/O Pins	54
Digital I/O with PWM Pins	15
Analog Pins	16
USB Connectivity	Standard A/B USB
Shield Compatibility	Yes
Ethernet/Wifi/Bluetooth	No (a Shield/module can enable it)

Figure 6: Hardware Specification of Arduino Mega 2560

3.3. Arduino Nano

The Arduino Nano is a microcontroller board based on the ATmega168. The controller will be serially interfaced with Raspberry Pi Computing Unit and will be used to read the data from the IMU Sensor and communicate it to Raspberry pi unit.

Type	Arduino Nano
Processor	ATmega 168
Clock Speed	16 MHz
Flash Memory (kilo Byte)	16
EEPROM (kilo Byte)	1
Voltage Level	5V
Digital I/O Pins	14
Digital I/O with PWM Pins	6
Analog Pins	8
USB Connectivity	Standard A/B USB

Figure 7: Hardware Specification of Arduino Nano 168

4. How to Setup Raspberry Pi 3B+ With Linux based on Raspbian and ROS-Kinetic

4.1 Initial Setup

ROS Kinetic will be used as a controlling software and will be installed on the Raspberry Pi 3B+. At present , Ubuntu Mate 16.0.4 OS is not compatible with Raspberry Pi 3B+ . Therefore we will install Linux based on Raspbian instead.

Step 1: Preparation

Get your Raspberry Pi 3B+, an SD memory card with 16Gb or more memory (class 10 or more), a micro USB cable (for power), Monitor, Keyboard and Mouse.

Step 2: Install Linux based on Raspbian

- Download the Linux distro image based on Raspbian which is pre-installed with ROS-Kinetic from the download
- After download, unzip the downloaded file.
- Guide to burn the image to SD card
    - Visit etcher.io and download and install the Etcher SD card image utility.
    - Run Etcher and select the Linux image you downloaded on your computer or laptop.
    - Select the SD card drive.
    - Click Burn to transfer the image to the SD card.

Step 3: Raspberry Pi PC

- After the installation, you can login with username **pi** and password **turtlebot**. In this case, you have to connect your Raspberry Pi to your monitor using an HDMI cable, and connect your keyboard and mouse to the Raspberry Pi.
- Expand filesystem to use a whole SD card.

```
sudo raspi-config
(select 7 Advanced Options > A1 Expand Filesystem)
```

- Also, It is possible to connect Raspberry Pi to a wireless network instructions
- To change the Password, Locale and Time zone setting (Optional):
    - sudo raspi-config > 1 Change User Password
    - sudo raspi-config > 4 Localisation Options > I1 Change Locale

- sudo raspi-config > 4 Localisation Options > I2 Change Timezone

Note : Initial Setup Tutorial for Raspberry Pi 3B+ is available in the Tutorial .

5. How To install ROS Kinetic on Raspberry Pi 3B (Ubuntu Mate) - Fresh Installation

Raspberry Pi 3B Model is fully compatible with Ubuntu Mate 16.0.4. Since , the ROS Framework is fully compatible with Ubuntu Mate – Linux Distribution . We will look for steps on How to install ROS Kinetic on Raspberry Pi 3B Model with Ubuntu Mate 16.04.02 OS.

5.1 Install OS

We will use Ubuntu MATE as OS on our Raspberry Pi 3B. The version Ubuntu MATE 16.04 fully supports the built-in-Bluetooth and Wi-Fi on the Raspberry Pi 3B.

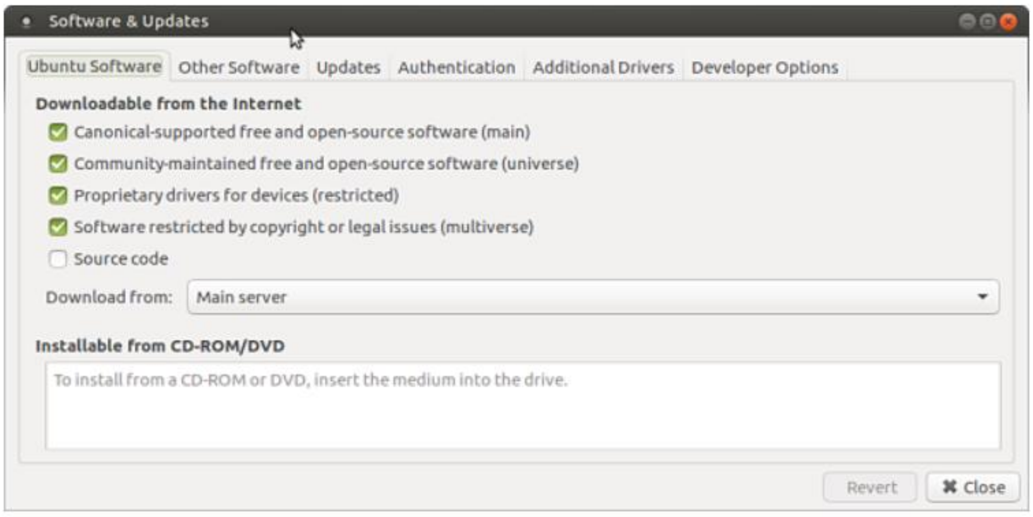
You will need a microSD Card that is 8GB or greater. The file systems will be automatically resized, on first boot, to occupy the unallocated space of the microSD card. The OS can be installed from here :

https://ubuntu-mate.org/raspberry-pi/

5.1.1 Install ROS Kinetic

Step 1 **:** Go to System -> Administration -> Software & Updates

Step 2: Check the checkboxes to repositories to allow “restricted,” “universe,” and “multiverse.”


Software & Updates

Ubuntu Software | Other Software | Updates | Authentication | Additional Drivers | Developer Options

Downloadable from the Internet

- [✓] Canonical-supported free and open-source software (main)
- [✓] Community-maintained free and open-source software (universe)
- [✓] Proprietary drivers for devices (restricted)
- [✓] Software restricted by copyright or legal issues (multiverse)
- [ ] Source code

Download from: Main server

Installable from CD-ROM/DVD

To install from a CD-ROM or DVD, insert the medium into the drive.

Revert | ✖ Close

Software and Updates

Step 3 : Setup your sources.list

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/so
urces.list.d/ros-latest.list'
```

Step 4 : Setup your keys

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BAD
E8868B172B4F42ED6FBAB17C654
```

Step 4: Make sure your Debian package index is up to date

```
sudo apt-get update
```

Step 5: **Desktop Install:** ROS, rqt, rviz, and robot-generic libraries

```
sudo apt-get install ros-kinetic-desktop
```

Step 6: Initialize rosdep

```
sudo rosdep init
```

```
rosdep update
```

Step 7: Environment setup

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Following all the instructions from section 4, you should be ready with Ubuntu Mate OS and ROS Kinetic successfully installed on your Computing Unit Raspberry Pi 3B Unit . Now , Let’s move on to next section on “ Setting up the Sensors” in the ROS Environment.

Step 8 : Dependencies for building packages

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Step 9: Create and initialize the catkin workspace

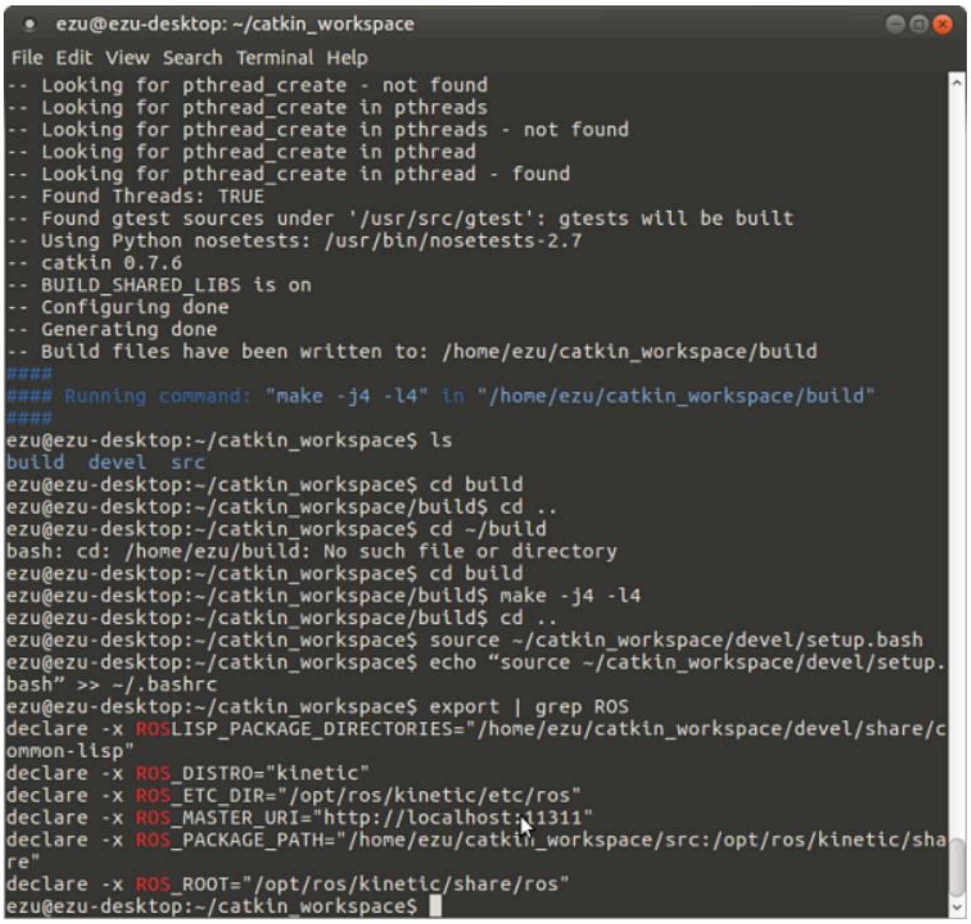
```
mkdir -p ~/catkin_workspace/src
cd catkin workspace/src
cd ~/catkin_workspace/
catkin_make
```

Step 10 : Add the catkin_workspace to your ROS environment

```
source ~/catkin_workspace/devel/setup.bash
echo “source ~/catkin_workspace/devel/setup.bash” >> ~/.bashrc
```

Step 11: Check the ROS environment variables

```
export | grep ROS
```



```
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.6
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ezu/catkin_workspace/build
####
#### Running command: "make -j4 -l4" in "/home/ezu/catkin_workspace/build"
####
ezu@ezu-desktop:~/catkin_workspace$ ls
build  devel  src
ezu@ezu-desktop:~/catkin_workspace$ cd build
ezu@ezu-desktop:~/catkin_workspace/build$ cd ..
ezu@ezu-desktop:~/catkin_workspace$ cd ~/build
bash: cd: /home/ezu/build: No such file or directory
ezu@ezu-desktop:~/catkin_workspace$ cd build
ezu@ezu-desktop:~/catkin_workspace/build$ make -j4 -l4
ezu@ezu-desktop:~/catkin_workspace/build$ cd ..
ezu@ezu-desktop:~/catkin_workspace$ source ~/catkin_workspace/devel/setup.bash
ezu@ezu-desktop:~/catkin_workspace$ echo “source ~/catkin_workspace/devel/setup.
bash” >> ~/.bashrc
ezu@ezu-desktop:~/catkin_workspace$ export | grep ROS
declare -x ROSLISP_PACKAGE_DIRECTORIES="/home/ezu/catkin_workspace/devel/share/c
ommon-lisp"
declare -x ROS_DISTRO="kinetic"
declare -x ROS_ETC_DIR="/opt/ros/kinetic/etc/ros"
declare -x ROS_MASTER_URI="http://localhost:11311"
declare -x ROS_PACKAGE_PATH="/home/ezu/catkin_workspace/src:/opt/ros/kinetic/sha
re"
declare -x ROS_ROOT="/opt/ros/kinetic/share/ros"
ezu@ezu-desktop:~/catkin_workspace$ 
```

The setup looks like in the picture

Note : This tutorial to Setup ROS-Kinetic on Raspberry Pi 3B Model is followed from this link

6. Master and Slave configuration

Visualization of Sensor data on the Raspberry pi computer will slow down the processes which are running on Raspberry Pi Computing Unit. Therefore , It’s good practice to do the visualization of sensor data on the Slave computer with high processing power . Hence the master & slave setup has to be established between raspberry pi and laptop.

Raspberry pi is configured as Master and Laptop will be configured as slave.

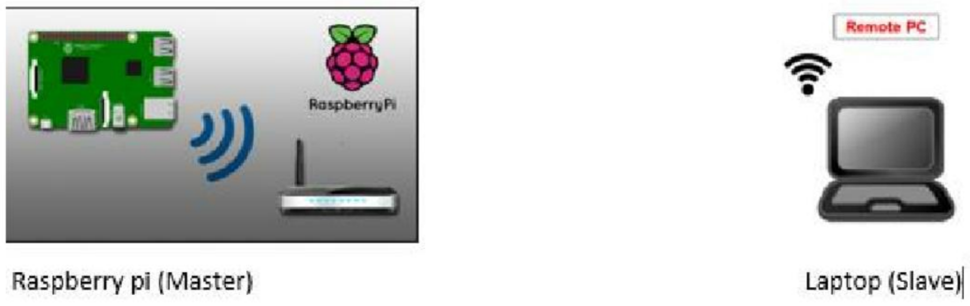


Figure 8. Master Slave Configuration

6.1 Configuring the .bashrc file (Master configuration)

So here is the main thing that needs to be done. We need to say our computer who is the master computer running ROS and what is there’s IP in the network. Since Computers running ROS don’t know their IP address.

First in the Raspberry pi computer open terminal and type:

```
$ ifconfig
Get the ip address of Raspberry pi.
```

```
$ gedit .bashrc
```

go to the bottom of your .bashrc file and paste then save it.

```
export ROS_MASTER_URI=http://localhost:11311/
export ROS_HOSTNAME= raspberrypi_ip_address

export ROS_IP=raspberrypi_ip_address
```

In first line we specify the address of the master along with the port to operate.

```
ROS_MASTER_URI=http://localhost:11311/
```

we basically say our master PC that you are the Host so in order to specify the IP address we say take your own **localhost** IP address.

finally source your .bashrc file

```
$ source .bashrc
```

Now we need configure .bashrc file of our Laptop/Computer i.e Slave Computer

Get the ip address of Slave Computer. Open terminal in Laptop and type

```
$ ifconfig
```

```
$ gedit .bashrc
```

go to the below of your file and paste then save

```
export ROS_MASTER_URI=http://raspberrypi_pi_address:11311/

export ROS_HOSTNAME=Laptop_ip_address

export ROS_IP=Laptop_ip_address
```

You can see the change , we here specified **ROS_MASTER_URI** as http://raspberry_pi_address:11311/here **raspberry_ip_address** is IP address of our master PC. We tell the IP address of master PC to our Laptop/computer. Rest of the line specifies IP address of the own just it.

finally you have to source your .bashrc file

```
$ source .bashrc
```

Now we just have to test if they are communicating with each other. We can do different thing to check it like running a **publisher** in Raspberry Pi PC and **subscribing** the topic in our Slave Computer.

7. Setting up the Sensors

In this section , We go through over on how to bring sensors running up in ROS Environment.

7.1 Installing the RPLidar A1 Sensor on the Raspberry Pi

A RPLIDAR A1 is a low cost LIDAR sensor (i.e., a light-based radar, a “laser scanner”) from Robo Peak suitable for indoor robotic applications.

We are using RPLidar A158 , the assembly instructions for the RPLidar is straightforward. It comes with a ribbon cable , you simply attach to the bottom of the board , and then to the USB adapter.

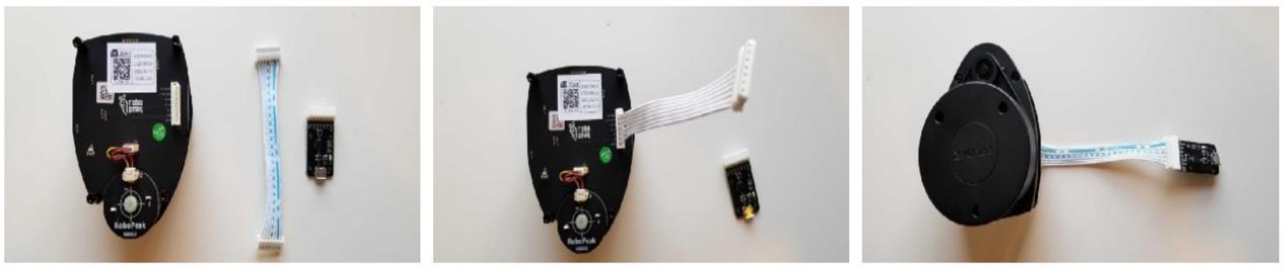


Figure 9. A158 RPLidar , ribbon cable and USB Adapter

Now , make the connections of the RPLidar to Pi as shown in the figure.

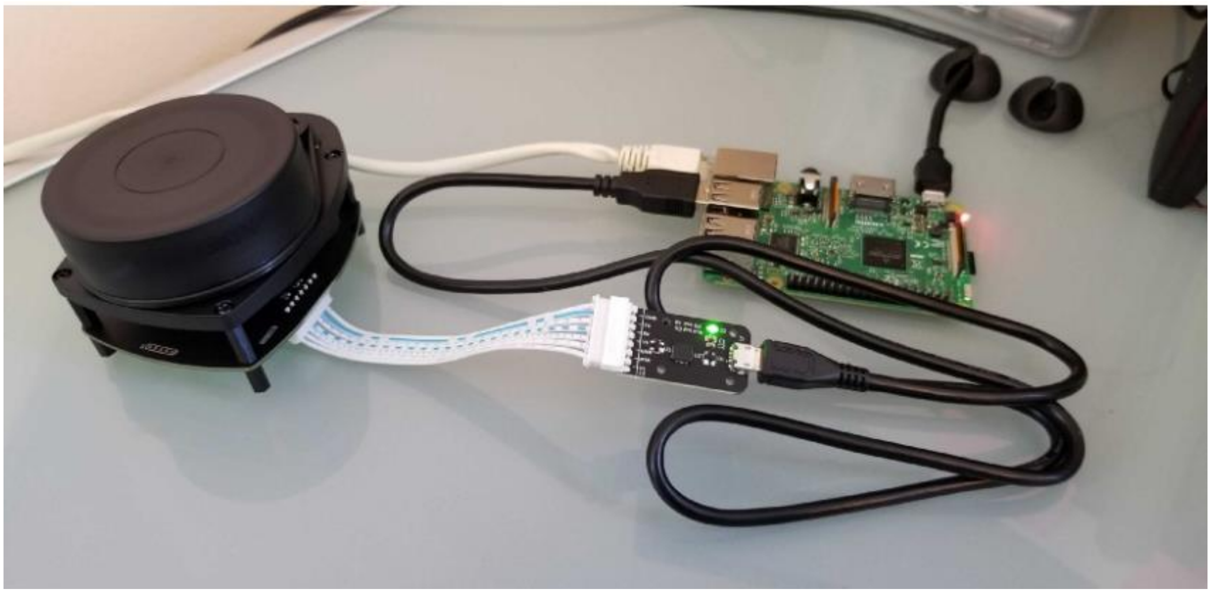


Figure 10. A158 RPLidar connection to the Raspberry Pi.

Step 1: Preparation

Get your Raspberry Pi 3B+, up and with ROS Kinetic running. Make the connections as shown in the Figure 10.

Step 2: *Create and initialize the catkin workspace*

```
mkdir -p ~/rplidar_ws/src
cd rplidar_ws/src
cd ~/rplidar_ws/
catkin_make
```

Step 3 : *Add the rplidar_ws to your ROS environment*

```
source ~/rplidar_ws/devel/setup.bash
echo "source ~/rplidar_ws/devel/setup.bash" >> ~/.bashrc
```

Step 4: Clone this project to your rplidar's workspace src folder

```
cd rplidar_ws/src
```

```
sudo git clone https://github.com/Slamtec/rplidar_ros.git
```

Step 5: Running catkin_make to build rplidarNode and rplidarNodeClient

```
cd ~/rplidar_ws/
catkin_make
```

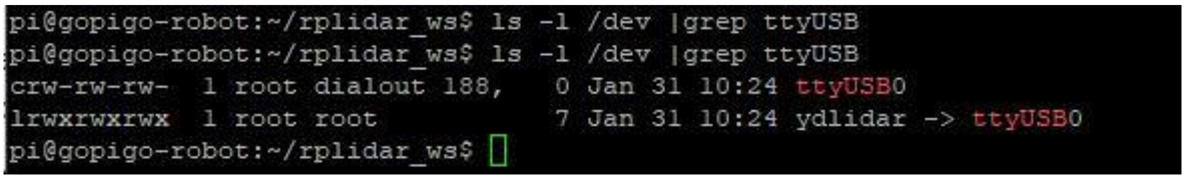
Step6: Activate the Serial Port for RPLidar

The RPLidar is going to use the serial ports in Ubuntu. We need to give the proper permissions for it to work. **Before you begin, plug the RPLidar into your USB Port.**

First, check the location of the RPlidar serial port, use the command

```
ls -l /dev|grep ttyUSB
```

This should list all the available serial devices:



```
pi@gopigo-robot:~/rplidar_ws$ ls -l /dev |grep ttyUSB
pi@gopigo-robot:~/rplidar_ws$ ls -l /dev |grep ttyUSB
crw-rw-rw-  1 root dialout 188,   0 Jan 31 10:24 ttyUSB0
lrwxrwxrwx  1 root root           7 Jan 31 10:24 ydlidar -> ttyUSB0
pi@gopigo-robot:~/rplidar_ws$
```

We’re looking for the number that comes after ttyUSB, in the above image it’s 0 (zero). So finally we give permissions to the serial 0 line:

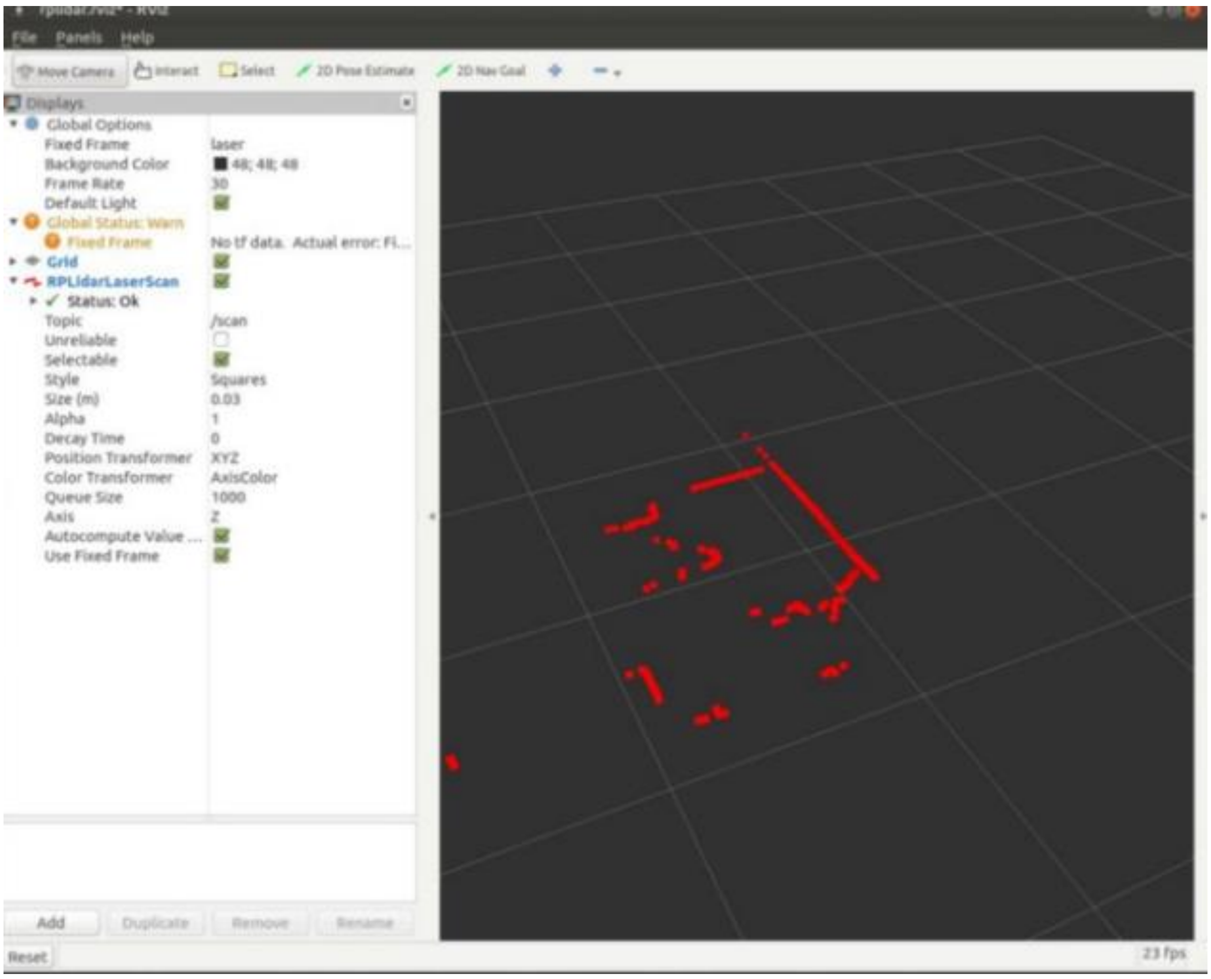
```
sudo chmod 666 /dev/ttyUSB0
```

Step6: Test

The RPLidar should be spinning at this point. Start a new terminal session and type:

```
roslaunch rplidar_ros view_rplidar.launch
```

And you should see something like this, with the Lidar sensor now scanning the room!



Global Options	
Fixed Frame	laser
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Fi...
Grid	✓
RPLidarLaserScan	✓
Status: Ok	
Topic	/scan
Unreliable	☐
Selectable	✓
Style	Squares
Size (m)	0.03
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	AxisColor
Queue Size	1000
Axis	Z
Autocompute Value ...	✓
Use Fixed Frame	✓

Figure 11. Scanning the Room with a laser using the RPLidar!

7.2 Setting the Neato XV-11 Lidar Sensor on the Raspberry Pi

Laser 360 range sensor from vacuum cleaner Neato XV-11 is one of the most popular and cheapest way to get lidar for robotics experiments.

Sensor has two connectors - first is an ordinary serial port, another is power for motor. Of course there are drivers for ROS, which consume raw sensor data and convert it Laser Scan format.

LaserScan is an array of range sensor data within some angle range. Neato Lidar provides range data in 360 degrees. LaserScan information can be used in robotics for SLAM - map building, obstacle avoidance, etc…

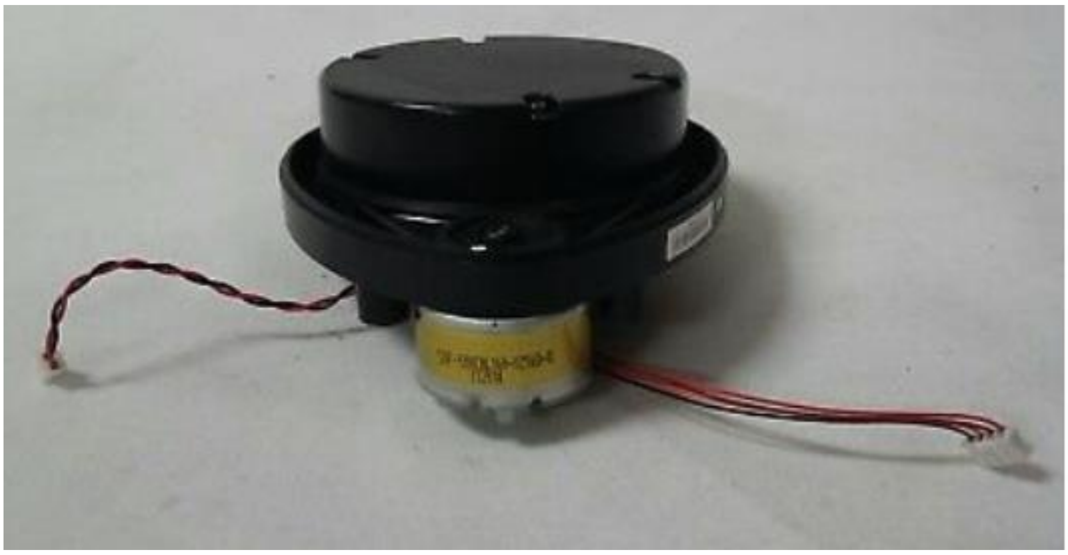


Figure 12. Neato XV-11 Lidar

Now , make the connections of the Neato XV-11 Lidar to Pi as shown in the Figure 13.

Step1: Preparation

Parts we need for connection Neato lidar to RPi

- Neato XV-11 Lidar
- 16 Mhz Arduino board , diode and N-MOSFET for PWM output for motor.

Now , make the connections to the Neato XV-11 Lidar as shown in the Figure 13.

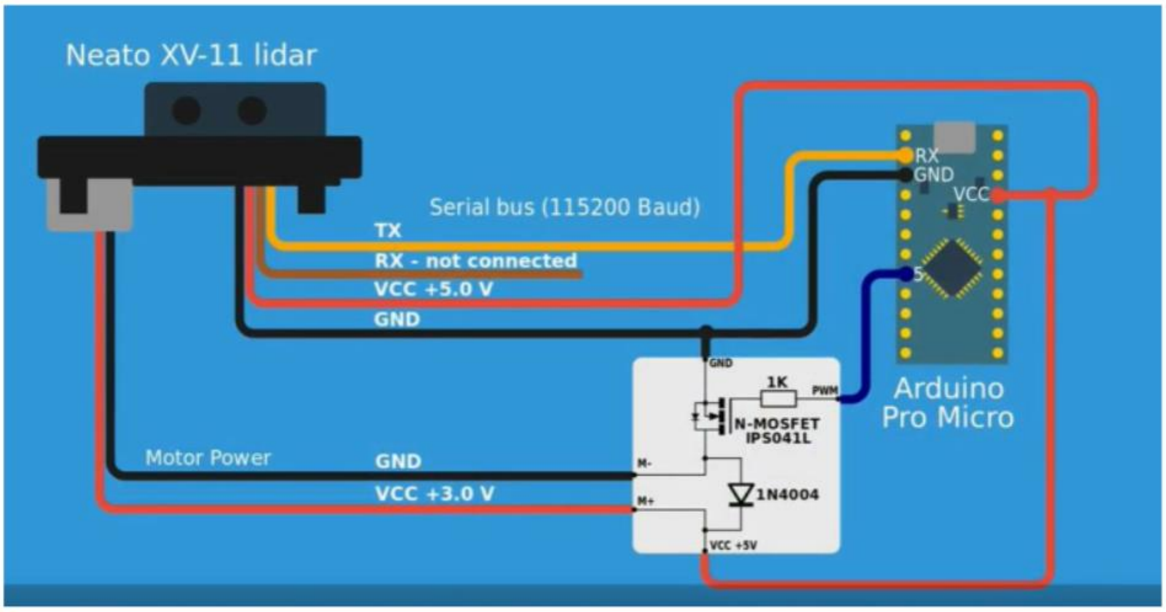


Figure 13. Neato XV-11 Lidar Connection with Arduino board, MOSFET & diode.

Now, Connect the Neato XV-11 Setup to the Raspberry Pi with as shown in Figure 14.

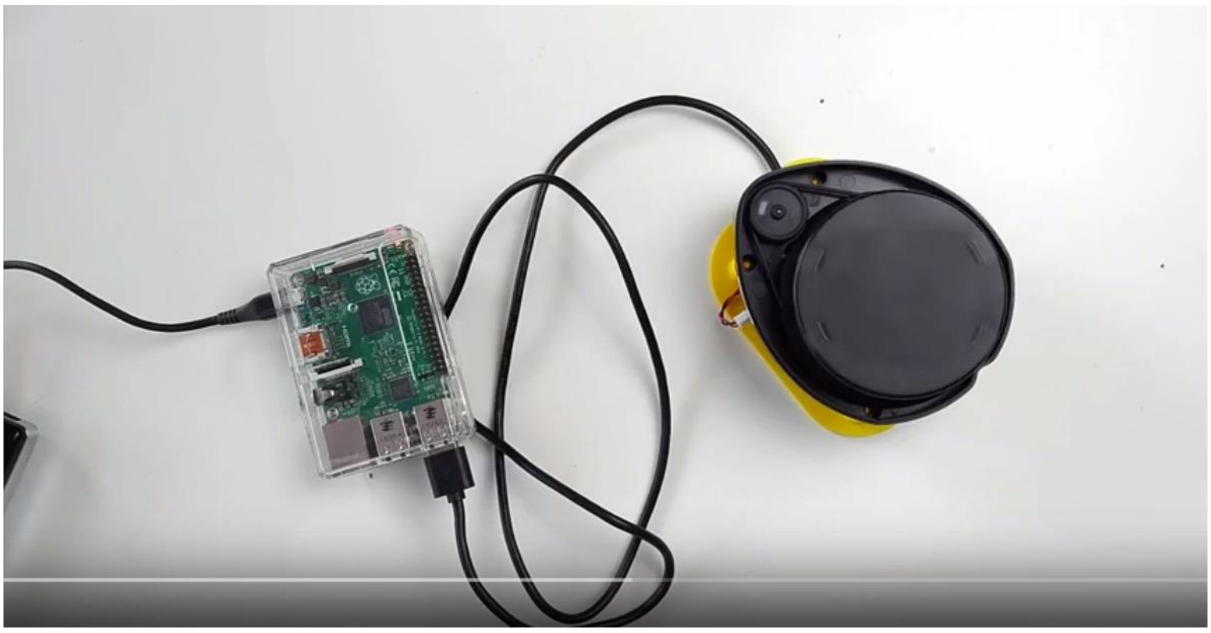


Figure 14. Connection setup of Neato XV-11 Lidar with Raspberry Pi.

Wiring is rather simple: +5V DC for logic and maximum 3V DC for motor. But note that some old revisions were powered by 3.3V, so please check what version do you have! Remember that the motor needs **3.0V instead of 3.3V** or it will spin too fast.

Step 2 : Now, let's install needed driver . Create and initialize the catkin workspace

```
mkdir -p ~/catkin_ws/src
```

```
cd catkin_ws /src
```

```
cd ~/ catkin_ws /
```

```
catkin_make
```

```
cd catkin_ws /src
```

```
git clone https://github.com/rohbotics/xv_11_laser_driver.git
```

```
cd catkin_ws
```

```
catkin_make
```

Step 3 : Add the catkin_ws to your ROS environment

```
source ~/rplidar_ws/devel/setup.bash
```

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Step 4: Let's check if lidar is connected:

```
$ lsusb
$ ls -la /dev/ttyACM*
```

Step 5: Open new terminal and run roscore in background process:

```
 $ roscore
```

Step 6: Now, launch the neato_laser_publisher_node:

```
$ rosrun xv_11_laser_driver neato_laser_publisher _port:=/dev/ttyACM0 _firm
ware_version:=2
```

Where:
xv_11_laser_driver – package

Neato_laser_publisher – node name

Parameters:

_port:=/dev/tty/ACM0 – hardware port with lidar

_firmware_version:= lidar firmware version

Vast majority have version 2

Step 7: To see LaserScan data we can use Rviz. Default fixed frame – "neato_laser".

```
  $ rviz
```

Step 8: You can now view all the published topic by entering the following command in a new Terminal window.

```
$ rostopic list
```

Step 9: To view the data published by the Neato-Lidar , type the following command in a new Terminal window.

```
$ rostopic echo /scan
```

Note: For more information on the steps , you can follow the link

7.3 Setting up Raspberry Pi camera in ROS Kinetic

Step1: Preparation

Parts we need

- Raspberry Pi Camera Module

Now , make the connections of Camera module to the Pi as shown in the Figure 15.

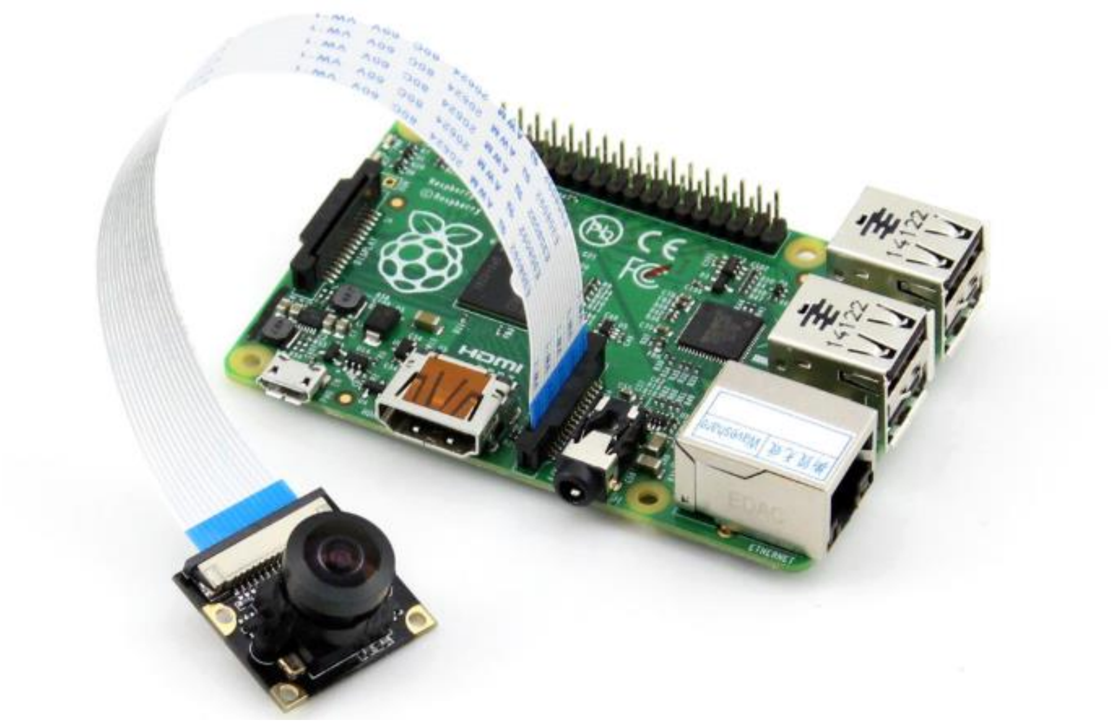


Figure 15. Raspberry Pi Camera Module with Pi Setup

Step 2: Type in your Raspberry Pi termina1.

```
sudo apt-get install libraspberrypi-dev

sudo pip install picamera
```

This will install all the necessary dependency packages.

Step 3: Now open the file /boot/config.txt and add the following two lines at the bottom of the file:

```
start_x = 1

gpu_mem=128<span style="font-family: arial, helvetica, sans-serif; font-siz
e: 14pt;"> </span>
```

Step 4: Just reboot the Raspberry Pi

```
sudo reboot
```

Step 5: To test everything went ok and works, just execute this tiny python script. If you get the image , then the camera is working perfectly:

```
Import picamera

camera=picamera.PiCamera()

camera.capture('testimage.jpg')

camera.close()
```

Step 6: To publish Image Stream into ROS Topics

Go to the terminal in your Raspberry Pi and type:

```
cd ~/catkin_ws/src

git clone https://github.com/dganbold/raspicam_node.git

cd ~/catkin_ws/

catkin_make –pkg raspicam_node
```

Step 7 : If the compilation went Ok, you now can use it. For this, you have to first start the node with all the services for the camera.

```
cd ~/catkin_ws/

source devel/setup.bash

roslaunch raspicam_node camera_module_v2_640x480_30fps.launch
```

Step 8: Open another terminal, start the camera service , which will activate the camera and start publishing.

```
rosservice call /raspicam_node/start_capture
```

Step 9: And now you are ready to view the images and use them in ROS. You can use rqt_image_view, for example, to connect to the image published in a topic and see it. here are two options to see the video stream:

```
rosrun image_view image_view image:=/raspicam_node/image_raw

rosrun rqt_image_view rqt_image_view
```

7.4 Setting up Communication between IMU-GY85 Sensor, Arduino Nano & Raspberry Pi camera in ROS Kinetic

Step1: Preparation

Parts we need

- IMU-GY85 Sensor Module
- Arduino Nano
- Raspberry Pi 3B
- Jumper Pins to connect IMU-GY85 Sensor to Arduino Nano
- USB Cable to connect Arduino Nano to Raspberry Pi 3B Board

Now , make the connections of IMU-GY85 Sensor to Arduino Nano as shown in the Figure 16. And also, connect the Arduino Nano board to Raspberry Pi 3B Board as shown in Figure 17.

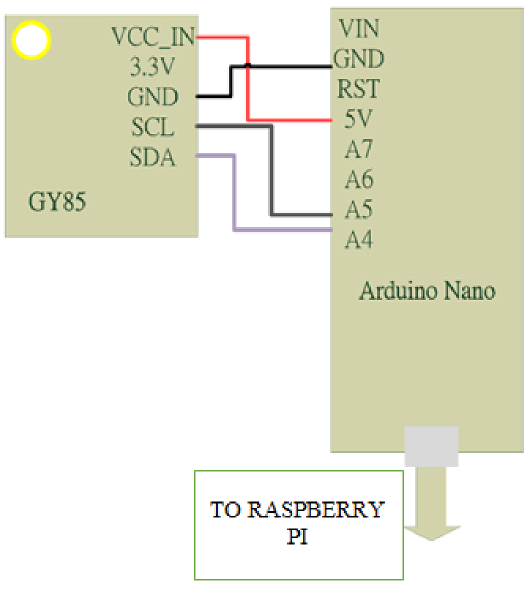


Figure 16. IMU Connection with Arduino

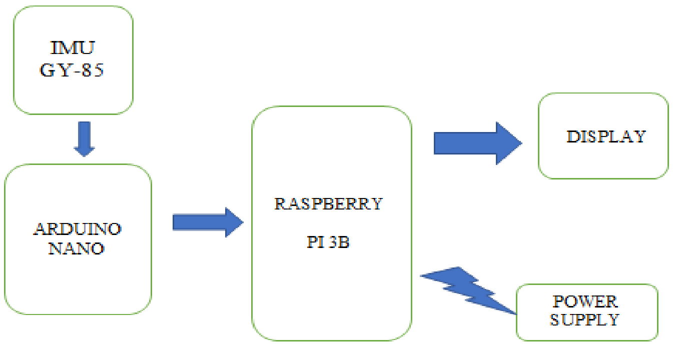


Figure 17. Raspberry Pi 3B Connection with Arduino Nano.

Step 2: Installing and Setting Up Arduino with ROS Kinetic (Raspberry Pi 3B)

Arduino Nano Communicates the data from IMU Sensor to Raspberry Pi board via serial communication. Therefore, we need to setup the serial communication between Arduino Nano and Raspberry Pi Board.

This guide will walk you through how to install and setting up an Arduino board to work with Raspberry Pi 3B having in common ROS Kinetic.

You must have a Raspberry Pi 3B with ROS kinetic installed, an Arduino Board connected via the USB port to Pi, and some Linux Knowledge

Install the Arduino IDE

To install the Arduino IDE o**n** the Ubuntu Mate operation system , use the following commands in the Linux terminal.

```
sudo apt-get update
sudo apt-get install arduino arduino-core
```

Install rosserial

After installing the Arduino IDE, the next step is installing the package that allows communication between ROS and the serial port of the Arduino board. The package is called rosserial_arduino and allows the node that will run on the Arduino to publish or subscribe to the nodes running on Raspberry Pi 3. The rosserial package contains three other packages: rosserial_msgs, rosserial_client, and rosserial_python.

```
sudo apt-get install ros-kinetic-rosserial-arduino

sudo apt-get install ros-kinetic-rosserial
```

After installing the Arduino IDE and the rosserial package, we will first check the IDE, but not before giving Administrator privileges to the current user for the Arduino Permission Checker.

The command is:

```
sudo usermod -a -G dialout your_user_here
```

To open the Arduino IDE, write the following command in the Ubuntu terminal:

```
arduino
```

Once we have checked the installation of the IDE, the next step is to close it and continue the setup operations. To close, use the Ctrl + C keys.

If you give the command *ls* in the Ubuntu terminal, you will find a new sketchbook directory. If you do not want to change your location or name, all Arduino sketches will be saved in this directory.

To write Arduino sketches for ROS, we need the ros_lib library.

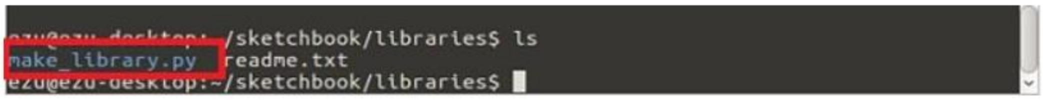
Install the ros_lib library

The link between ROS and Arduino is through the ros_lib library. This library will be used as any other Arduino library.

To install the ros_lib library, type the following commands in the Ubuntu terminal:

```
cd sketchbook/libraries

rosrun rosserial_arduino make_library.py .
```



```
ezu@ezu-desktop:~/sketchbook/libraries$ ls
make_library.py  readme.txt
ezu@ezu-desktop:~/sketchbook/libraries$
```

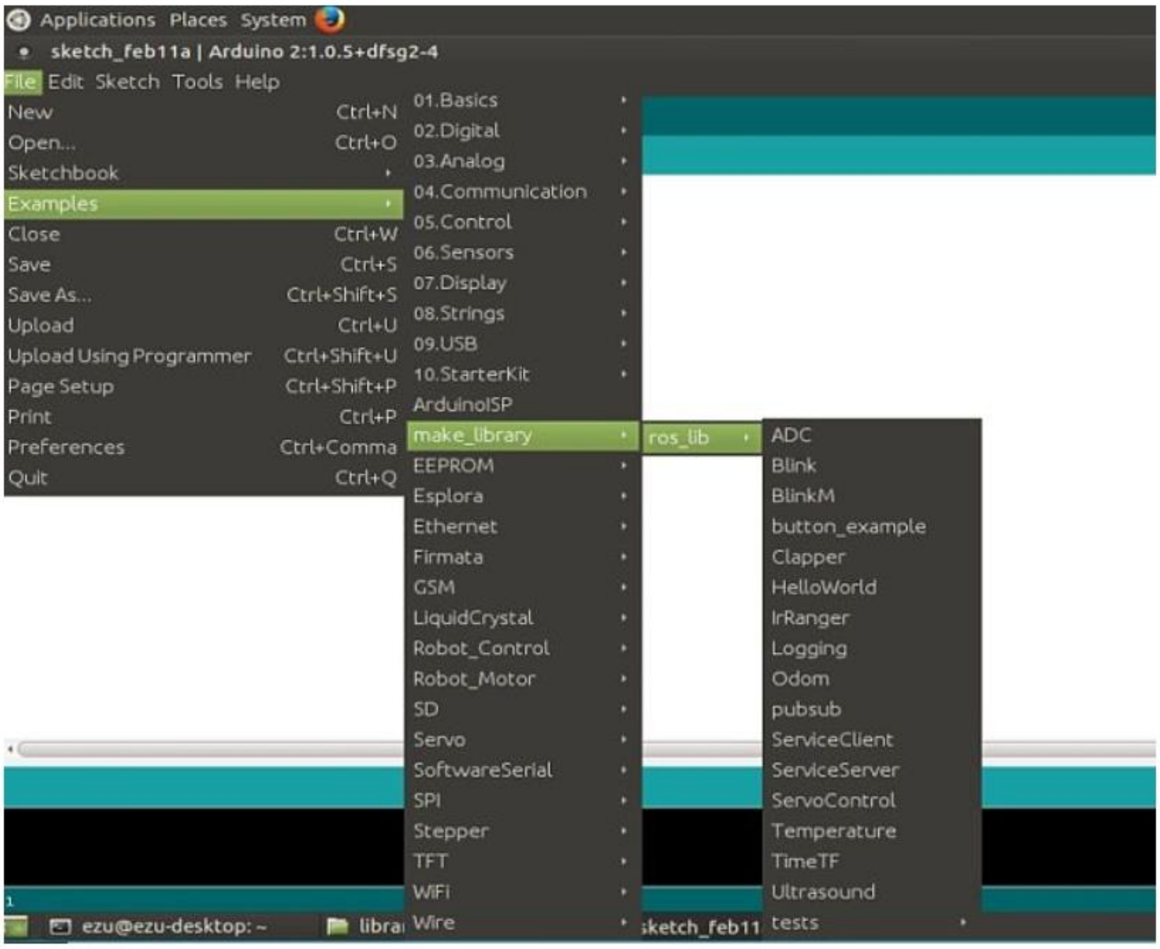
make_library.py

Check the Arduino IDE settings

To check the settings made, we will open the Arduino IDE again. To launch the application, we will use the command:

```
arduino
```

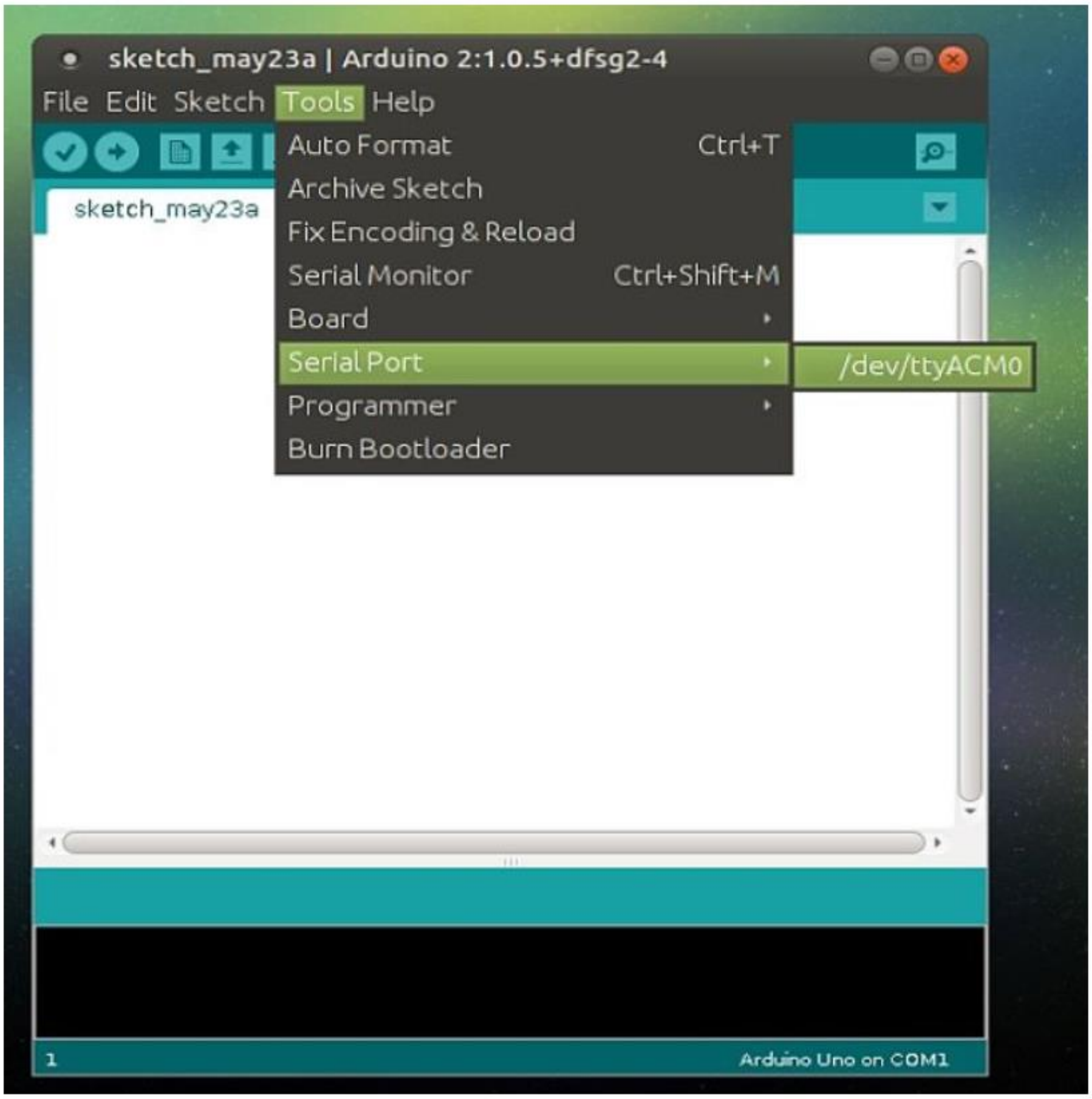
After opening the IDE, check if you have access to the ros_lib examples: **File -> Examples -> make_library -> ros_lib**



ros_lib

Check the serial port:

Tools-> Serial Ports



Arduino's serial port

Step 3: Software Installation, Install ROS razor_imu_9dof Package

Clone the ROS source repository and build it:

```
$ cd catkin_ws/src
$ git clone https://github.com/KristofRobot/razor_imu_9dof.git
$ cd ..

$ catkin_make
```

Load Firmware into IMU GY-85 Sensor Board

In this step you copy the Razor firmware from the ROS package and build it and load it into the board using the Arduino IDE.

If you haven't done so before, start the Arduino IDE. It creates a directory in your home for Arduino source code; version 1.0 creates a directory called "sketchbook" and version 1.5 creates a directory called "Arduino". This is where the Arduino IDE expects to find source code for the firmware you're going to install on the Razor.

- Copy the Razor firmware from the razor_imu_9dof/src directory to your arduino source directory.

```
$ roscd razor_imu_9dof

$ cp -r src/Razor_AHRS ~/Arduino
```

Replace ~/Arduino with ~/sketchbook if needed.

- Plug the Razor USB cable into your ROS workstation
- Restart the Arduino IDE
- Open the Razor_AHRS sketch in Arduino. Select the Razor_AHRS tab.
- In Arduino:
    - Look at the top of the Razor_AHRS.ino file, it contains useful information about the firmware.
    - There is a section labeled "USER SETUP AREA" where you can set some firmware defaults, including the board type.
        - Caution: choose the correct Razor hardware revision when compiling and uploading the firmware to the board. The setting is located in Razor_AHRS.ino under "HARDWARE OPTIONS"!
    - Go to "Tools" → "Board" and select "Arduino Pro or Pro Mini (3.3v, 8mhz) w/ATmega328". Note: in Aduino 1.5+, the board menu doesn't allow selecting the voltage/frequency; go to the Processor menu after selecting "Arduino Pro or Pro Mini" and select "ATMega 328 (3.3V, 8Mhz)"
    - Go to "Tools" → "Serial Port" and select the port used with the Razor.
    - Go to "File" and hit "Upload to I/O Board". After a short while at the bottom of the *Arduino* code window it should say "Done uploading".

Create Configuration File

Create a custom configuration file *my_razor.yaml*:

- Copy the template file *razor.yaml*:

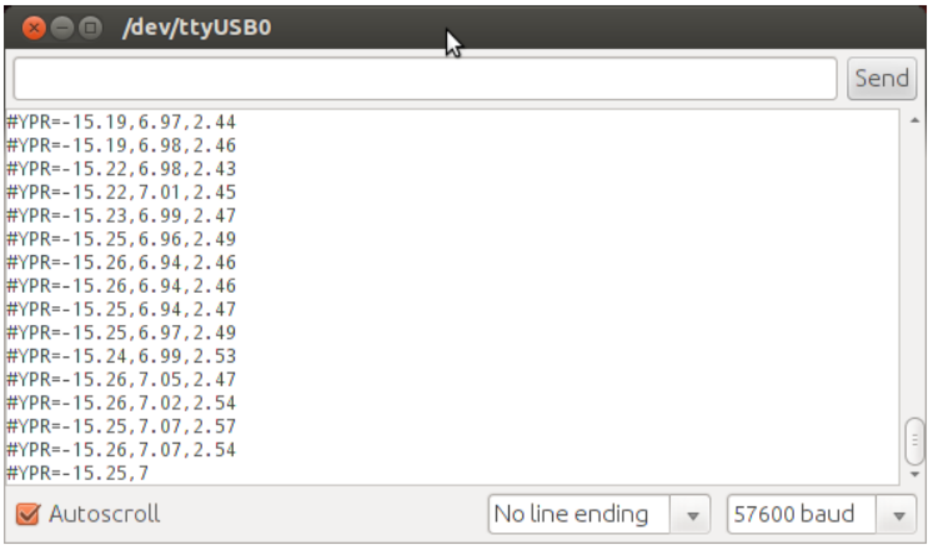
```
$ roscd razor_imu_9dof/config
$ cp razor.yaml my_razor.yaml
```

- Edit *my_razor.yaml* as needed. See further for more information on setting the calibration values.

Step 4: Testing the IMU Sensor

Serial Monitor

To test by staring at numbers, bring up the *Serial Monitor* of Arduino under "Tools" → "Serial Monitor". Set it to 57600 baud and send the string "#o1" to the AHRS. You should get output that looks like this:



```
#YPR=-15.19,6.97,2.44
#YPR=-15.19,6.98,2.46
#YPR=-15.22,6.98,2.43
#YPR=-15.22,7.01,2.45
#YPR=-15.23,6.99,2.47
#YPR=-15.25,6.96,2.49
#YPR=-15.26,6.94,2.46
#YPR=-15.26,6.94,2.46
#YPR=-15.25,6.94,2.47
#YPR=-15.25,6.97,2.49
#YPR=-15.24,6.99,2.53
#YPR=-15.26,7.05,2.47
#YPR=-15.26,7.02,2.54
#YPR=-15.25,7.07,2.57
#YPR=-15.26,7.07,2.54
#YPR=-15.25,7
```

The three numbers represent YPR (Yaw, Pitch, and Roll) in degrees.

Step 5: Echo the Imu Messages

The normal way to start publishing Imu data from the Razor is to run the launch file:

```
$ roslaunch razor_imu_9dof razor-pub.launch
```

You can now run "rostopic list" in another window and look for the /imu topic.
Run "rostopic echo /imu" and look at the Imu messages. Move the Razor around and see how the values in the Imu messages change.

Note: Tutorial on Installing and Setting Up Arduino with ROS Kinetic (Raspberry Pi 3B) is available here

And also, Official link to Install ROS razor_imu_9dof Package found here

7.5 Setting up of Movidius Neural Compute Stick with Raspberry Pi 3B for Object Detection

The Intel® Movidius® Neural Computing Stick (NCS) is a tiny, fanless, deep learning USB drive developed to learn AI programming.The NCS is powered by the low-power, high-performance Movidius ™ Visual Processing Unit (VPU) with Vision Accelerators. NCS can be referred to as a tiny GPU which enables the RPI to run computationally intensive deep learning models by just plugging in the device, without the use of any additional hardware. Pretty good results are given by RPI along with an NCS.

Step1: Preparation

we need

- Raspberry Pi 3B plus Raspbian Stretch OS
- Intel Movidius Neural Compute Stick
- Raspberry Pi Camera

Now, Setup the connection as shown in the Figure 19.

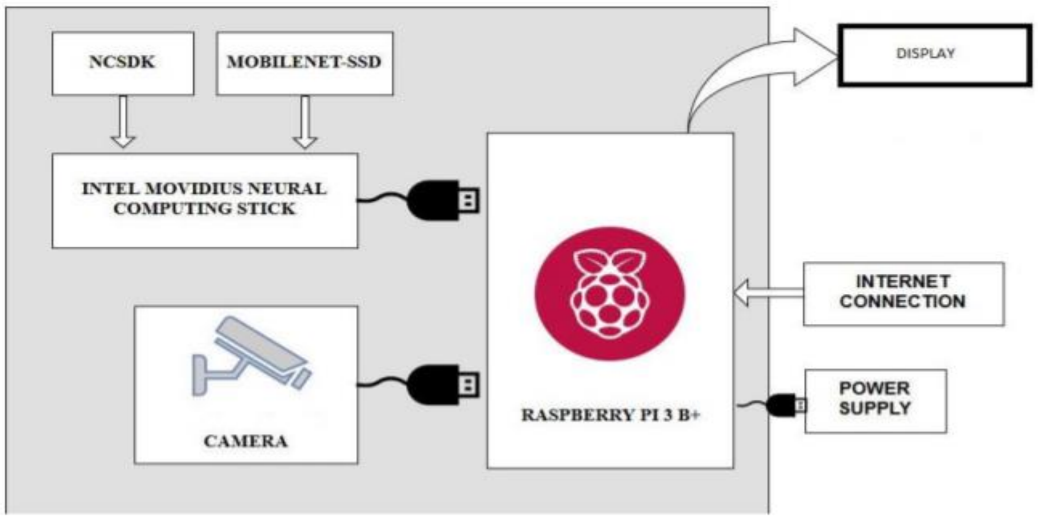


Figure 18. Connection of Rasp Camera, Neural Stick with Raspberry Pi 3B.

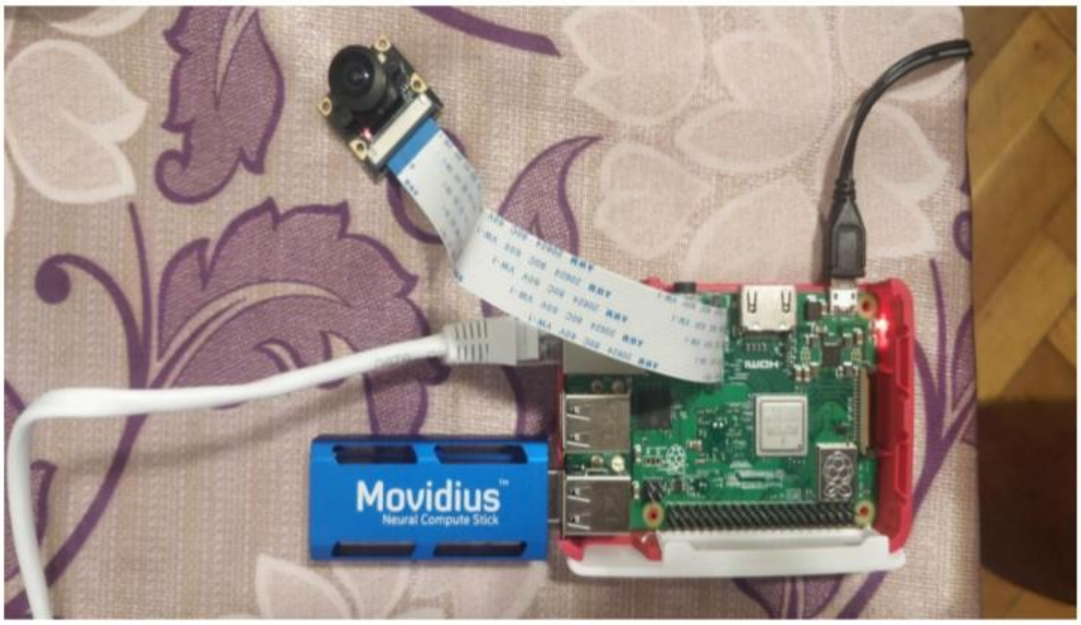


Figure 19. Hardware Setup

Step 2: Workflow and Installation of SDK on Neural Stick

This project involves more software requirements than the hardware setup. A separate development and deployment environment are necessary. It is not advisable to make the development activities also on the PI.

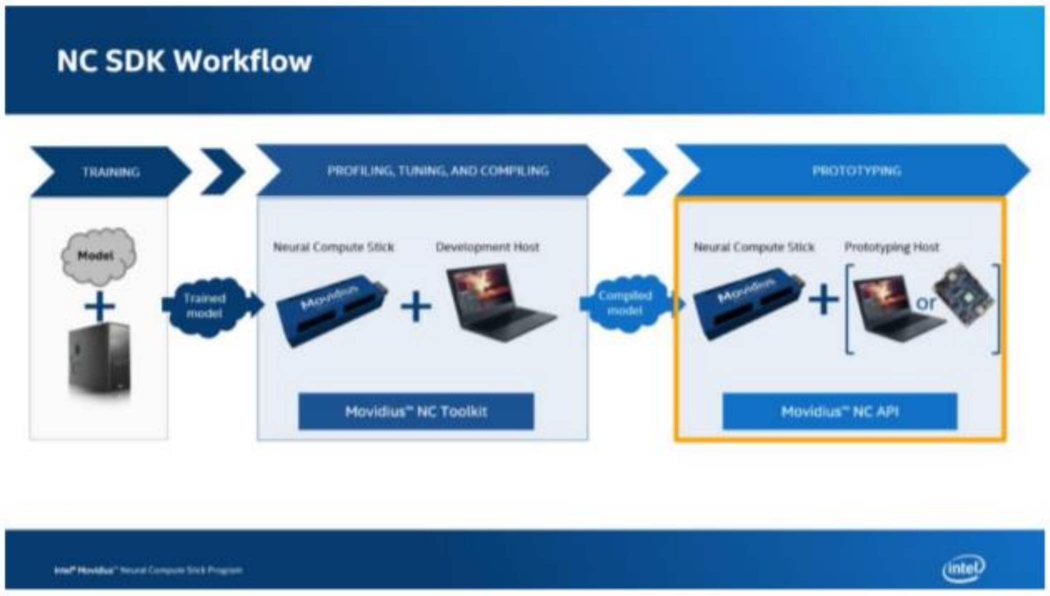


Figure 20. Workflow

Installation is carried out in 2 parts:

- Development host - It is a host PC for development, tuning, testing and compiling them onto the Compute Stick. Complete SDK has to installed.
- Deployment Environment - It is the RPI for which the API mode of installation will suffice.

PRE-REQUISITES:

• The host PC requires an Ubuntu 16.04 OS, 8 GB RAM and at least dual core CPU. More cores, the better. NCSDK is installed and is checked by compiling the examples.

• Currently, there are 2 versions of NCSDK's available in the GitHub repository – Version1 and Version2.

• Version1 is being used in this model as Version2 is not fully compatible and also it is confirmed by the NCS development team that models built on Version1 won't compile on Version2.

• Therefore, it becomes important to stick to one version throughout the compilation and deployment steps.

The NCSDK package is cloned from GitHub link and is installed which internally runs makefiles, installs Caffe, TF, OpenCV and other required dependencies. To check the NCS connection, we also install NCAPPZOO repository and run hello_ncs.py from the apps folder.

PRE TRAINED MODEL:

• Generally, a model is defined, trained and the layers are specified. Once the model is fully tweaked, we then generate a graph file.

• A graph is a file format of the model used to load onto the compute stick. • A pre trained model, Mobile SSD is chosen link.

• By using SSD, we only need to take one single shot to detect multiple objects within the image, thus making it faster for real-time applications.

Model was initially trained on MS – COCO and then fine-tuned on VOC0712.

Activation function – ReLu. • 21 Classes are present.

GRAPH FILE GENERATION:

Two files are required for this graph file generation.

• A Protxt file which defines the model structure such as input sizes, number of layers, etc.

• Weights file which has all the filter weights, layer weights, biases, etc

mvNCCompile MobileNetSSD_deploy.protxt -w MobileNetSSD_deploy.caffemodel -s-12 -o graph/graph

This command is used to compile protxt files and weights into a Graph.

DEPLOYMENT:

• The result graph file is loaded on to the deployment environment.

• The model, when deployed on RPI gives up to 7 frames per second, whereas it was 10 fps in the development environment.

A python script is deployed to get this up and running on the NCS.

The python script has the following functionalities

• To load the image, in this case, it's the live video feed from the RPI Camera via OpenCV

• load graph onto the device

• Pre-process the Image

• Load the image and return a prediction

• Post-processing of images

Generally, the output of the Graph is a list of a bunch of numbers

Output[0] – number of objects detected

Output[1] – Prediction

Output[2] – Confidence factor

Output[3] – x1 Co-ordinate

Output[4] – y1 Co-ordinate

Output[5] – x2 Co-ordinate

Output[6] – y2 Co-ordinate

All this information about the predicted image is used to draw bounding boxes and name the identified object.