



25/07/2018

Intern Project

Autonomous Rovers



Student: novkovia

Tutor: Venkatesan Muthukumar

UNIVERSITY OF NEVADA, LAS VEGAS

Table of Contents

Project	2
Intel up with Intel depth sensor	3
1) Intel Realsense presentation	3
2) Installation and start-up of the intel up	4
3) Installation of ROS and Realsense	6
Kinect sensor	9
1) Product presentation	9
2) Installation and result.....	10
RPLidar with ROS	12
1) Product presentation	12
2) Installation and result.....	13
Car control with Pixhawk and Roboclaw.....	17
1) Product and software presentation	17
2) Hardware installation	19
3) Initialization and control by RC	23
4) Installation and test software with Intel Up.....	29
Nvidia TX2 and ZED Camera	31
1) Product presentation	31
2) Installation and result.....	32
Sources	34

Project

The goal of this project is to control an autonomous rover in a room. This will scan the environment using an RC controller and then will move automatically after capturing the entire room.

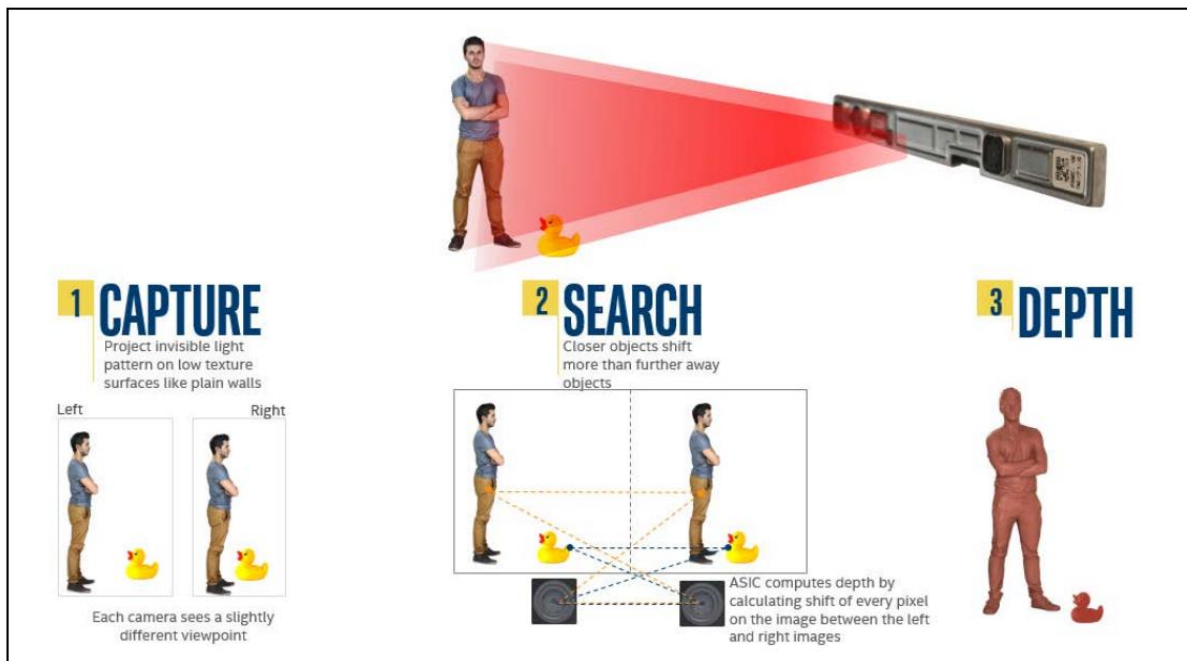
For this, we have a set of components allowing to manage and control the robot but also to capture the environment. I will describe all the components used and their operation during the different tasks.

Intel up with Intel depth sensor

1) Intel Realsense presentation

The R200 camera is a USB 3.0 device that can provide color, depth, and infrared video streams. Depth video streams are like color video streams, except each pixel has a value representing the distance away from the camera instead of color information. It consists of an infrared laser projection system, two infrared and a full HD color imaging sensors. The depth video stream is generated with stereo vision technology assisted by the Infrared laser projector and the two infrared imaging sensors. Color data is provided by the full HD color imaging sensor. The R200 module is not intended to be the primary photography solution. It has the ability to synchronize with a high resolution world facing camera for depth + photography applications.

The R200 module uses stereo vision to calculate depth. The stereo vision implementation consists of left infrared camera, right infrared camera, and an infrared laser projector. The left and right camera data is sent to the R200 ASIC. The ASIC calculates depth values for each pixel in the image. The infrared projector is used to enhance the ability of the system to calculate depth in scenes with low texture. Traditionally, scenes with low texture such as walls presented a challenge for stereo vision systems to calculate depth.



The R200 ASIC is a USB 3.0 composite device which exposes all hardware endpoints to the operating system. The ASIC is a bulk device and transmits depth and color videos streams in data bursts rather than as constant video streams. The camera module is compliant with the USB 3.0 specification. The module does not support USB 2.0 connections and does not route the USB 2.0 pins.

2) Installation and start-up of the intel up

For all of our tasks, we must use a microcontroller. We will first use the microcontroller supplied with the Realsense sensor, the intel up.



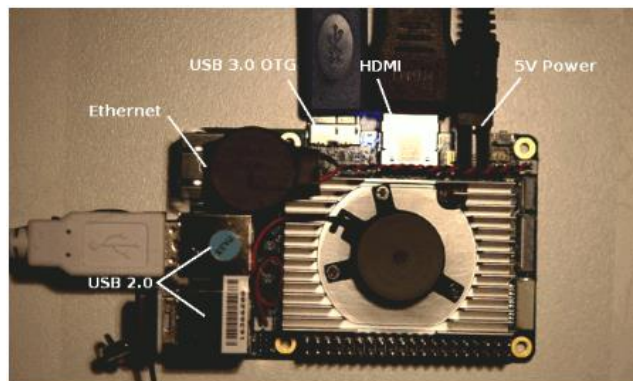
We must start by downloading install ubuntu 14.04.5 on our microcontroller. Here is the link:

<http://releases.ubuntu.com/14.04/ubuntu-14.04.5-desktop-amd64.iso>

Then is necessary to install linux on a bootable usb key using the software Rufus:

<https://rufus.akeo.ie/>

Then plugged all the connectors:



- Plug the USB keyboard and mouse into two of the four USB 2.0 ports.
- Plug the USB 3.0 OTG into the UP board's USB 3.0 port. Do not plug your RealSense camera into the adapter yet; it needs to be unplugged while the udev rules are being configured in a later step.
- Plug an HDMI cable into the UP board's HDMI port and connect the other end of the cable to your monitor.
- Plug the USB flash drive into one of the remaining USB 2.0 ports.
- Plug one end of the 5V power supply into an outlet and the other end into the UP board.

Your UP board will now boot. After a few seconds, the UP logo should appear on your display. The system will then begin booting from the Ubuntu installer on the USB flash drive. To install Ubuntu onto the UP board, select option "Install Ubuntu."

Follow the different steps to install, during the step "who are you" the data has been filled:

- Your name: robot
- Your computer's name: robot
- Pick a username: robot
- Choose a password: robot
- Confirm your password: robot
- I selected 'Log in automatically'
- I did not select 'Encrypt my home folder'

Your root password (when prompted by running "sudo") is "robot".

Now that you have Ubuntu installed, update the system to have the latest software. Do this via the command line.

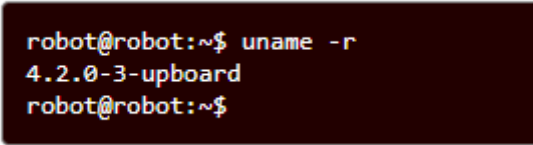
```
sudo apt-get update
sudo apt-get -y dist-upgrade
```

You can install the linux-upboard kernel:

```
sudo add-apt-repository ppa:ubilinux/up
sudo apt-get update
sudo apt-get -y install linux-upboard
sudo apt-get -y autoremove --purge 'linux-.*generic'
```

Verify you are running the linux-upboard kernel:

```
uname -r
```



```
robot@robot:~$ uname -r
4.2.0-3-upboard
robot@robot:~$
```

If the output indicates you are running a kernel with the string upboard in the name, you are ready to install ROS and Realsense package.

3) Installation of ROS and Realsense

Since the release of the upboard kernel, additional patches have been made available to enable more RealSense cameras. Prior to installing the ROS with RealSense support, your system needs to have access to the kernel source packages.

```
wget -q -O - https://bit.ly/en_krn1_src | sudo /bin/bash
```

Now that you have Ubuntu installed with kernel source packages enabled, install the ROS framework release "Indigo" including the RealSense packages.

You need to add the ROS repository to Ubuntu, update the package lists, and install the ROS base software:

```
sudo add-apt-repository http://packages.ros.org/ros/ubuntu
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net \ --recv-key
0xB01FA116
sudo apt update
```

Install ROS packages and update

```
sudo apt -y install ros-indigo-desktop-full python-rosinstall ros-indigo-
realsense-camera
sudo rosdep init
rosdep update
echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Adding setup.bash to the .bashrc file ensures that the ROS framework is available in the active environment.

The installation of the package ros-indigo-librealsense will build a new kernel image with the latest RealSense patches applied. Once installed, you will need to reboot your computer to take advantage of that kernel. As before, you can reboot via:

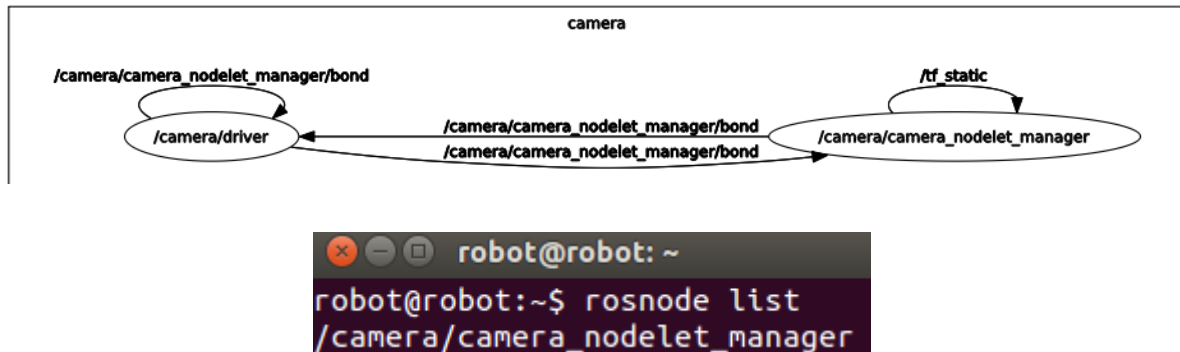
```
sudo reboot
```

If you haven't already, plug in your camera.

We can now test the camera:

```
roscore &
roscd realsense_camera
roslaunch realsense_camera r200_nodelet_rgbd.launch
```

With the camera node running in the background:



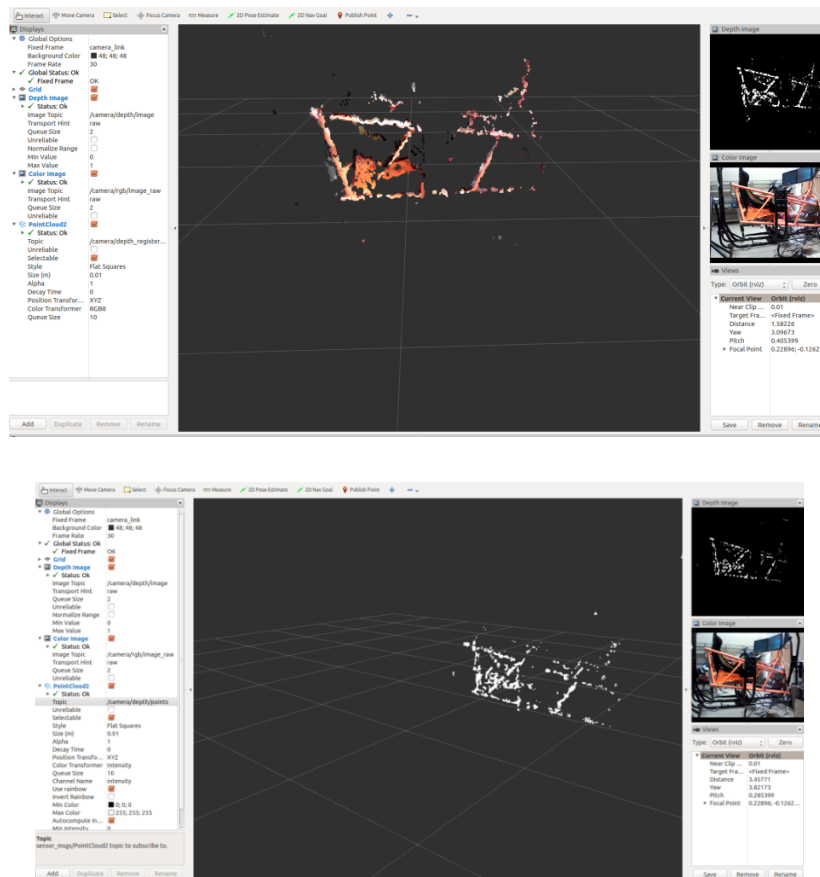
We can see that the Intel camera is visible on the nodes of ROS.

You can now continue on to running RViz.

```

roscd realsense_camera
roslaunch rviz rviz -d rviz/realsense_rgbd_pointcloud.rviz

```



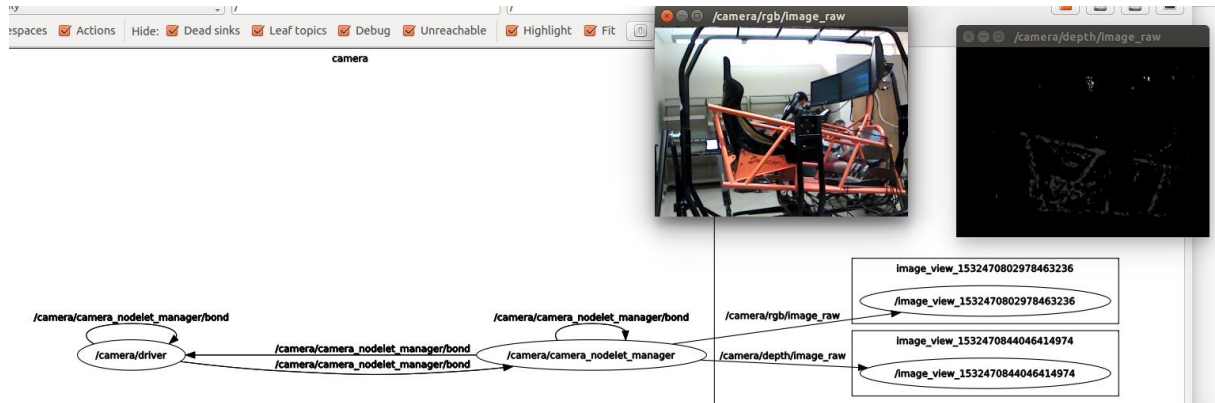
Once RViz loads, you can use your mouse to move the view around. The projection from the RealSense camera may be behind the default viewpoint, so you may need to zoom out and pan around to find the pointcloud.

For color stream:

```
roslaunch image_view image_view image:=/camera/rgb/image_raw
```

For depth stream:

```
roslaunch image_view image_view image:=/camera/depth/image_raw
```



It is observed that the Intel camera sends the information to the image view from a topic camera/rgb/image_raw and camera/depth/image_raw.

Kinect sensor

1) Product presentation

Kinect is a motion sensing input devices that was produced by Microsoft for Xbox 360 and Xbox One video game consoles and Microsoft Windows PCs. Based around a webcam-style add-on peripheral, it enabled users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands.



Kinect is one of the most popular sources of PointClouds - array of points with 3D coordinates information. It has proprietary connector - actually it's USB+12V bus.

This camera is equivalent to the Intel camera but it is used for many projects in internet. There is a lot of help and support.

2) Installation and result

We can use the Intel up card previously configured for the Intel Realsense camera. We need to install packages to use kinect on the intel up.

On RPi install freenect nodes and dependencies:

```
sudo apt install ros-kinetic-freenect-launch
```

Also install ros-kinetic-depthimage-to-laserscan node:

```
sudo apt install ros-kinetic-depthimage-to-laserscan
```

Run roscore on master:

```
roscore
```

And run example freenect launch file:

```
roslaunch freenect_launch freenect.launch
```

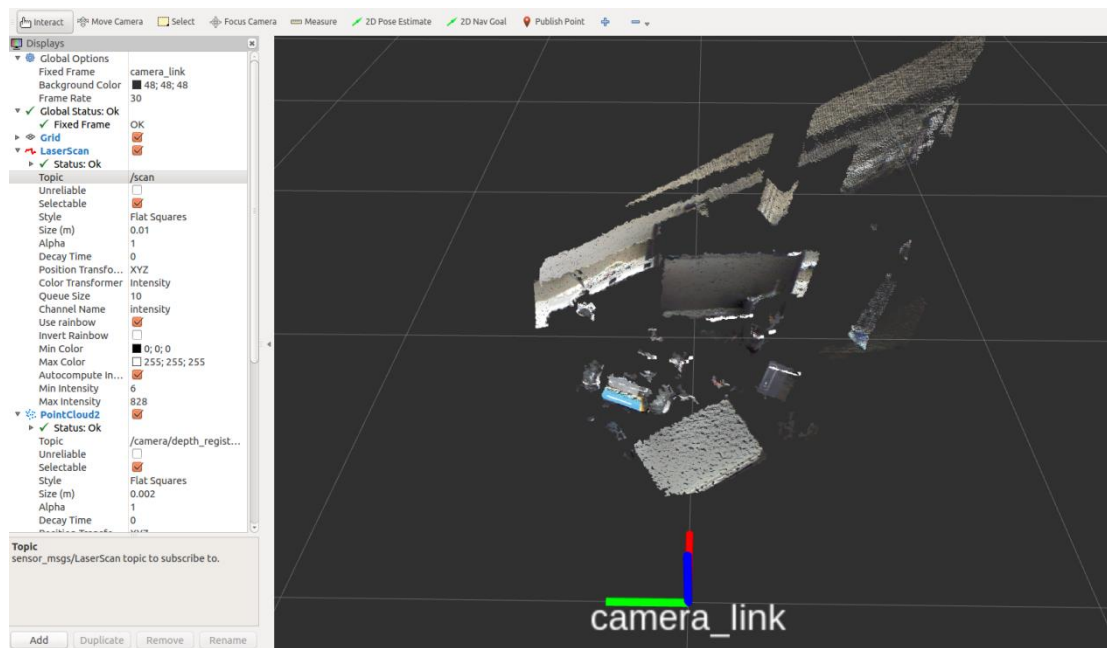
Run rviz on master and use kinect_view.rviz config:

```
Rviz
```

Rviz config that you downloaded below:

<https://gist.github.com/WinKILLER/df1a37f0406e5370abf39534bd8bbde1>

We can see pointcloud data.



To add depthimage-to-laserscan node let's create new launch file

Create file and paste node description from gist:

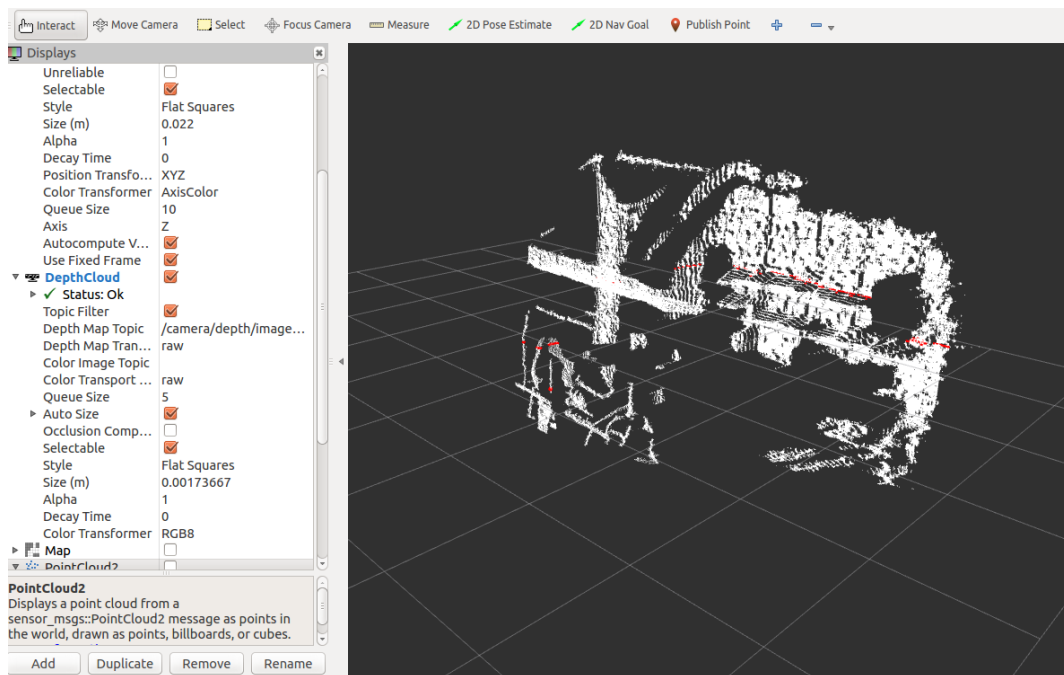
```
vim ~/rosvd_ws/src/vidsrv/launch/laserscan.launch
```

Download link:

<https://gist.github.com/WinKILLER/7a8f6aa494157f02a633efb3831ad69f>

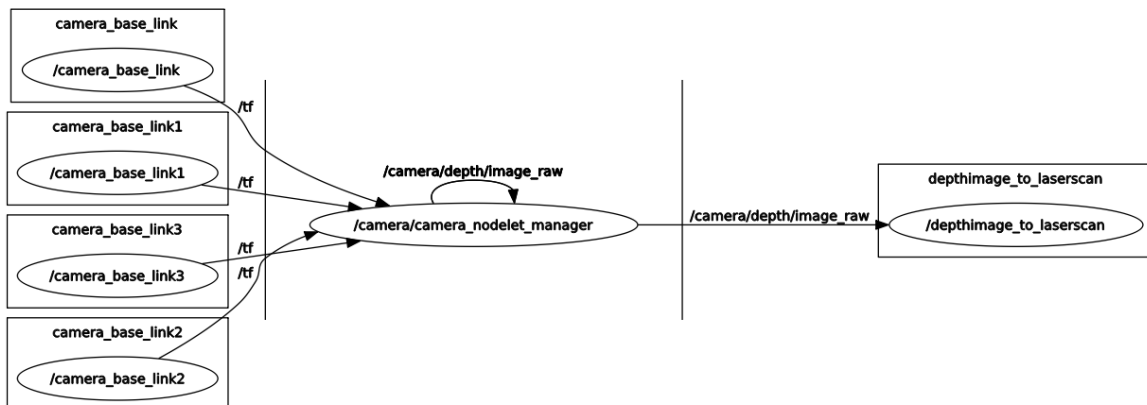
Run roscore, freenect node and new created one

Run rviz and launch kinect_view-2.rviz



Red stripe is laserscan visualization.

To visualize the operation of the Kinect with the client we will launch rqt_graph:



RPLidar with ROS

1) Product presentation

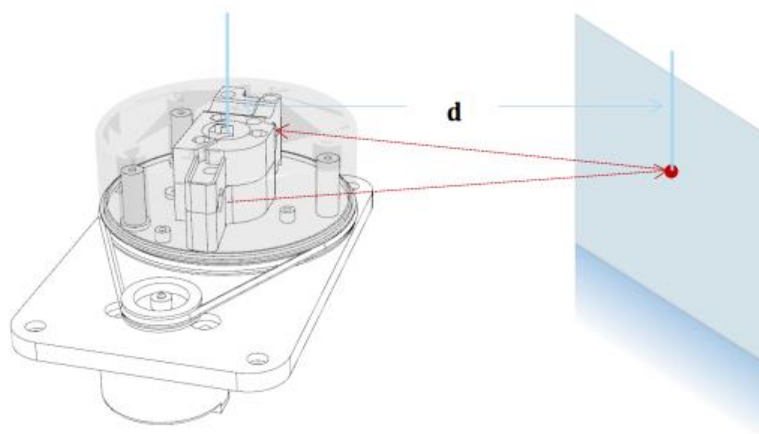
RPLIDAR A1 is a 360 degree 2D laser scanner (LIDAR) solution developed by SLAMTEC. The system can perform 360degree scan within 6meter range. The produced 2D point cloud data can be used in mapping, localization and object/environment modeling.

RPLIDAR A1 is basically a laser triangulation measurement system. It can work excellent in all kinds of indoor environment and outdoor environment without sunlight.



RPLIDAR is based on laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware developed by SLAMTEC. The system measures distance data in more than 2000 times per second and with high resolution distance output (<1% of the distance).

RPLIDAR emits modulated infrared laser signal and the laser signal is then reflected by the object to be detected. The returning signal is sampled by vision acquisition system in RPLIDAR A1 and the DSP embedded in RPLIDAR A1 start processing the sample data and output distance value and angle value between object and RPLIDAR A1 through communication interface.



2) Installation and result

To use this lidar, we need to install the different drivers provided. We can use Lidar on Windows and Linux.

For our project, we will set the lidar for use on a microcontroller equipped with ubuntu.

We must start by installing ROS on our microcontroller, here the Nvidia TX2 with ubuntu:

```

roscore http://10.18.17.128:11311/
nvidia@tegra-ubuntu:~$ roscore
... logging to /home/nvidia/.ros/log/bbe5e940-8083-11e8-88cd-00044b8d09fb/roslau
nch-tegra-ubuntu-20769.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://10.18.17.128:44614/
ros_comm version 1.12.13

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.13

NODES

auto-starting new master
process[master]: started with pid [20779]
ROS_MASTER_URI=http://10.18.17.128:11311/

```

After having tested if ROS is functional, we have to install the different drivers and sample for RPLIDAR. For this, we must follow this manipulation:

```

### Clone the ROS node for the Lidar in the catkin workspace src dir
git clone https://github.com/robopeak/rplidar_ros.git

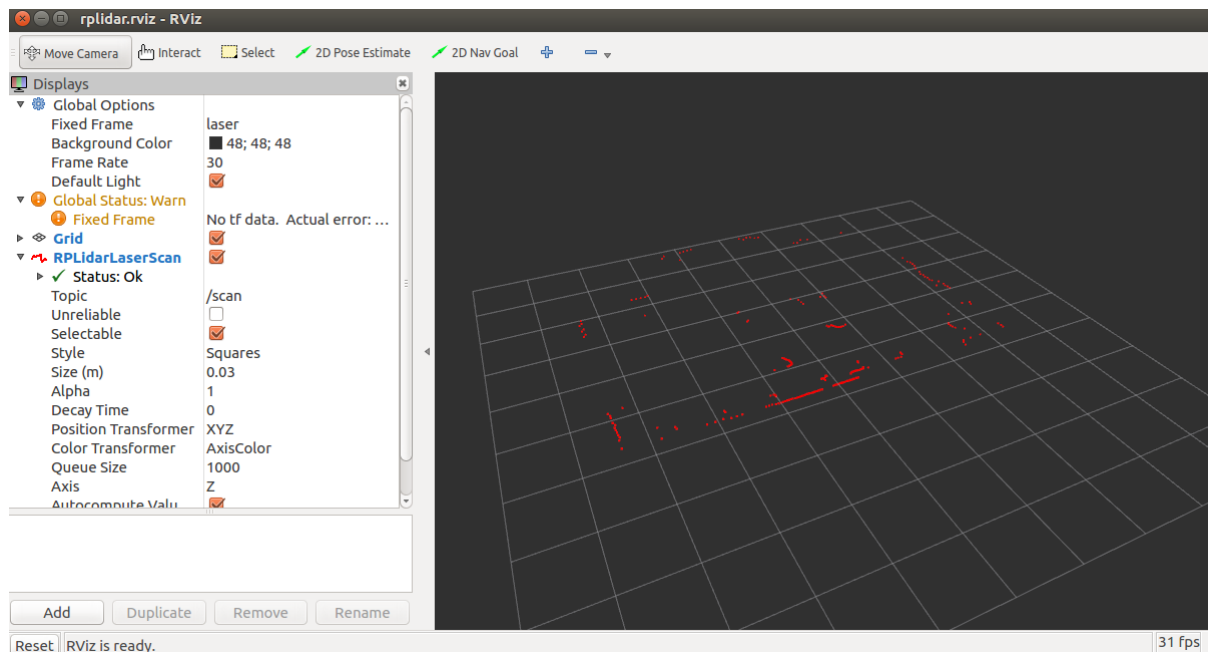
### Build with catkin
cd ~/catkin_ws/
catkin_make

### Set environment when build is complete
source devel/setup.bash

### Launch demo with rviz
roslaunch rplidar_ros view_rplidar.launch

```

Here is the result obtained after launching the Lidar roslaunch command:



We observed from the software Rviz which is a 3D visualizer for displaying sensor data and state information, the detection of the piece by the lidar.

```
nvidia@tegra-ubuntu: ~
nvidia@tegra-ubuntu:~$ rostopic list
/rosout
/rplidarNode
/rviz
nvidia@tegra-ubuntu:~$
```

We can see that the lidar is visible on the nodes of ROS.

We will now observe the results obtained at the output of our lidar, which allows the treatments to be linked at distance from the angle. For this, we will start the lidar with the following command:

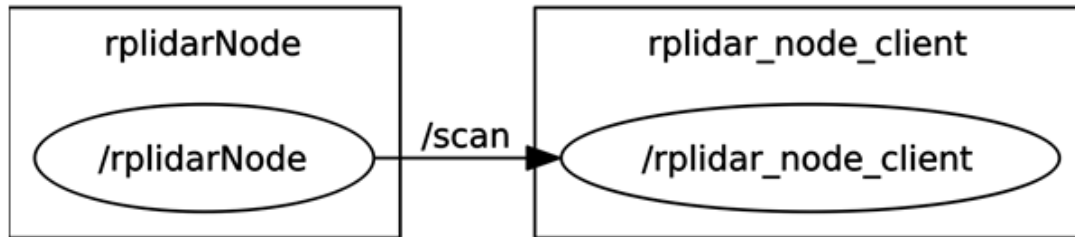
Run `roslaunch rplidar_ros rplidar.launch`

To visualize the result, we will launch a client program which allows visualizing the various values obtained on the terminal; the command is the following one:

`roslaunch rplidar_ros rplidarNodeClient`

To visualize the operation of the lidar with the client we will launch rqt_graph:

```
nvidia@tegra-ubuntu:~$ rosrn rqt_graph rqt_graph
```



It is observed that the lidar sends the information to the client from a topic /scan.

Here is a part of the results obtained at the exit of our lidar:

```
nvidia@tegra-ubuntu: ~/catkin_ws
[ INFO] [1530820035.881426349]: : [157.000031, 4.761500]
[ INFO] [1530820035.881455693]: : [158.000015, 2.461250]
[ INFO] [1530820035.881486029]: : [159.000015, 4.678000]
[ INFO] [1530820035.881516493]: : [160.000015, 4.604750]
[ INFO] [1530820035.881546029]: : [161.000000, inf]
[ INFO] [1530820035.881576461]: : [162.000031, 1.827000]
[ INFO] [1530820035.881606445]: : [163.000015, 1.870500]
```

Here is the code used to understand these results:

```
client.cpp (~/.catkin_ws/src/rplidar_ros/src) - gedit
Open [x]

/* RoboPeak LIDAR System
 * RPlidar ROS Node client test app
 *
 * Copyright 2009 - 2014 RoboPeak Team
 * http://www.robopeak.com
 */

#include "ros/ros.h"
#include "sensor_msgs/LaserScan.h"

#define RAD2DEG(x) ((x)*180./M_PI)

void scanCallback(const sensor_msgs::LaserScan::ConstPtr& scan)
{
    int count = scan->scan_time / scan->time_increment;
    ROS_INFO("I heard a laser scan %s[%d]:", scan->header.frame_id.c_str(), count);
    ROS_INFO("angle_range, %f, %f", RAD2DEG(scan->angle_min), RAD2DEG(scan->angle_max));

    for(int i = 0; i < count; i++) {
        float degree = RAD2DEG(scan->angle_min + scan->angle_increment * i);
        ROS_INFO(" : [%f, %f]", degree, scan->ranges[i]);
    }
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "rplidar_node_client");
    ros::NodeHandle n;

    ros::Subscriber sub = n.subscribe<sensor_msgs::LaserScan>("/scan", 1000, scanCallback);

    ros::spin();

    return 0;
}
```

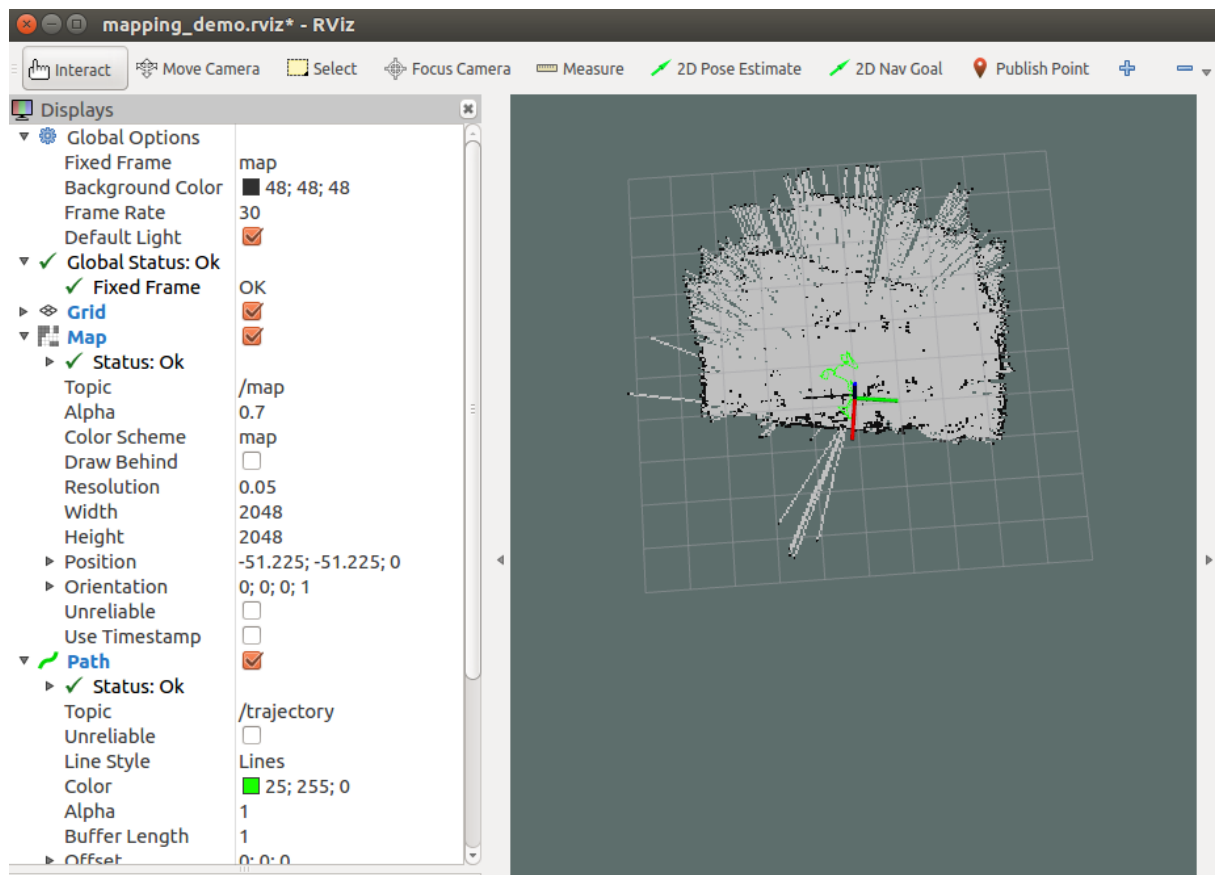

We have on the left the information related to the counter, is the number of laser dot that was captured. On the right, we have information on the angle and distance taken depending on the laser point.

In the example above, we then observe at the 160 degrees lidar position, we are at a distance of 4.6.

This information can then be transmitted to a set of components and software to perform a location mapping.

Then I tried to map the room using hector slam:

Run `roslaunch hector_slam_launch tutorial.launch`



The result is not correct; I must find a solution to have a precise geolocation RPlidar. Maybe by assembling this one with a depth sensor.

Car control with Pixhawk and Roboclaw

1) Product and software presentation

Pixhawk is an advanced autopilot system designed by the PX4 open-hardware project and manufactured by 3D Robotics. It features advanced processor and sensor technology from ST Microelectronics® and a NuttX real-time operating system, delivering performance, flexibility, and reliability for controlling any autonomous vehicle.

The benefits of the Pixhawk system include integrated multithreading, a Unix/Linux-like programming environment, completely new autopilot functions such as Lua scripting of missions and flight behavior, and a custom PX4 driver layer ensuring tight timing across all processes. These advanced capabilities ensure that there are no limitations to your autonomous vehicle.

The Pixhawk module will be accompanied by new peripheral options, including a digital airspeed sensor, support for an external multi-color LED indicator and an external magnetometer. All peripherals are automatically detected and configured with a ground control stations.



A ground station is typically a software application, running on a ground-based computer, which communicates with Rover via USB or wireless telemetry. It displays real-time data on the Rover performance and position and can serve as a “virtual cockpit”. A ground control stations can also be used to control a Rover on road or UAV in flight, uploading new mission commands and setting parameters.

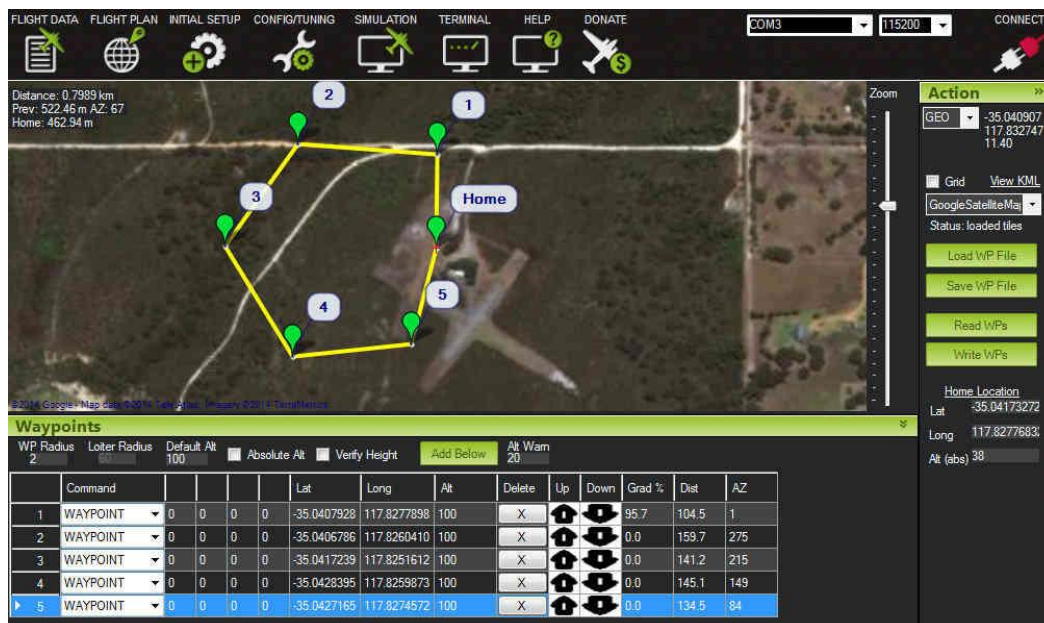
There are at least ten different ground control stations. On desktop there is (Mission Planner, APM Planner 2, MAVProxy, QGroundControl and UgCS. For Tablet/Smartphone there is Tower (DroidPlanner 3), MAVPilot, AndroPilot and SidePilot that can be used to communicate with ArduPilot (Copter, Plane, Rover...).

For our, we have to use Mission Planner software which is an open source and is compatible with Windows and Mac OS X. It allows updating the firmware of the Pixhawk controller, to make the calibrations of the accelerometer, the compass, radio control, etc.

We have access to a set of information that also implements the features related to a control station, displays several parameters concerning the rover, such as its speed, orientation, altitude, GPS position, etc.

Here is the download link of the software:

<http://ardupilot.org/planner/docs/common-install-mission-planner.html>

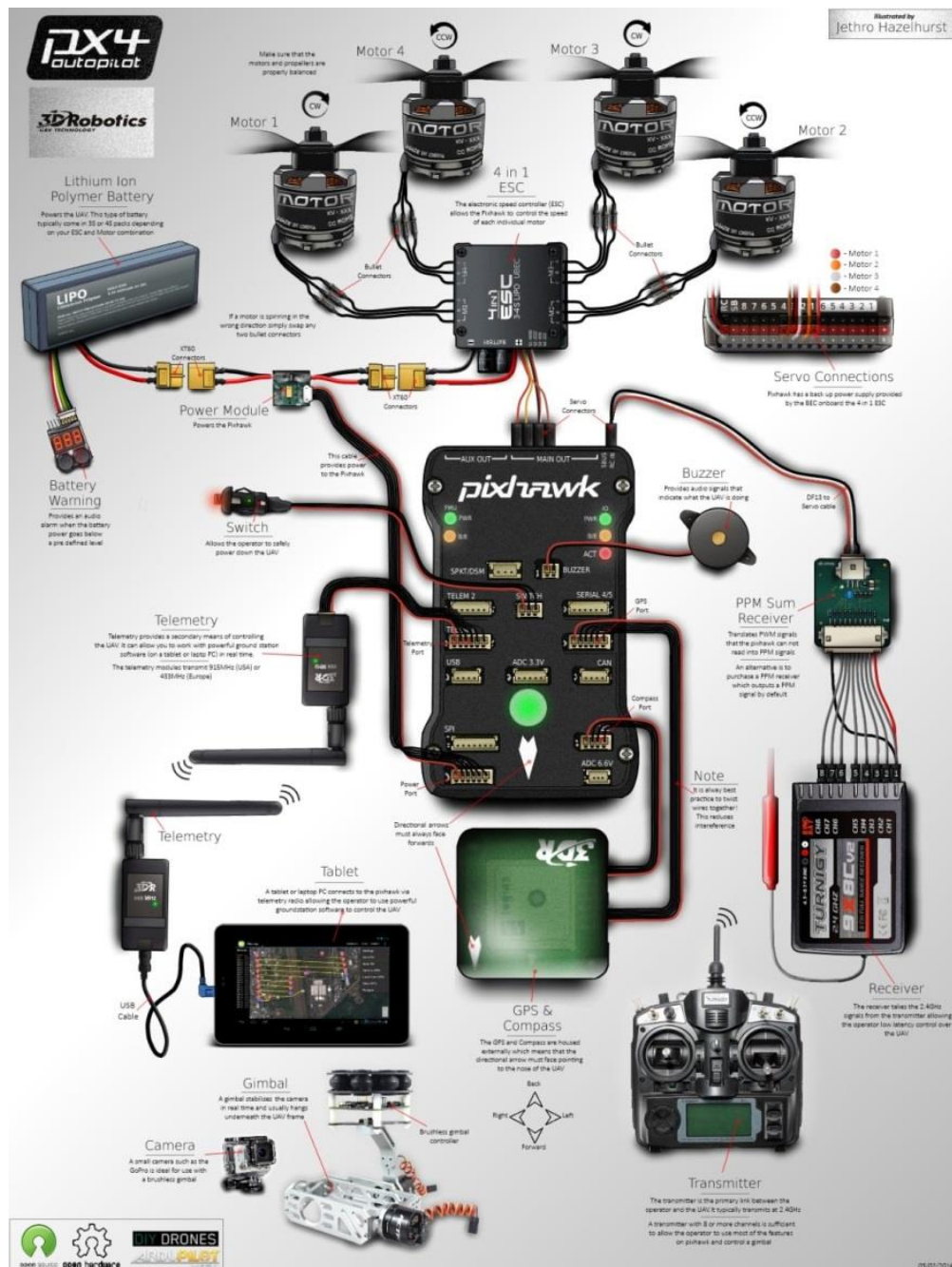


2) Hardware installation

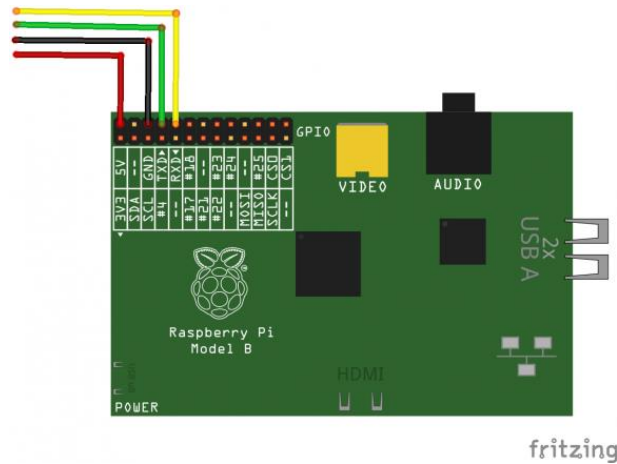
We now have the following components wired to our controller:

- Buzzer allowing knowing from different sound signals the state of our controller.
- Securities switch allowing the start of our Pixhawk from a manual pressure.
- Power module to power the controller and our entire rover.
- Radio telemetry control to have information about the car on our computer
- GPS module to give the position of our rover.
- RC receiver to control our car manually connected to the SPKT / DSM port.

Here is the wiring plan:

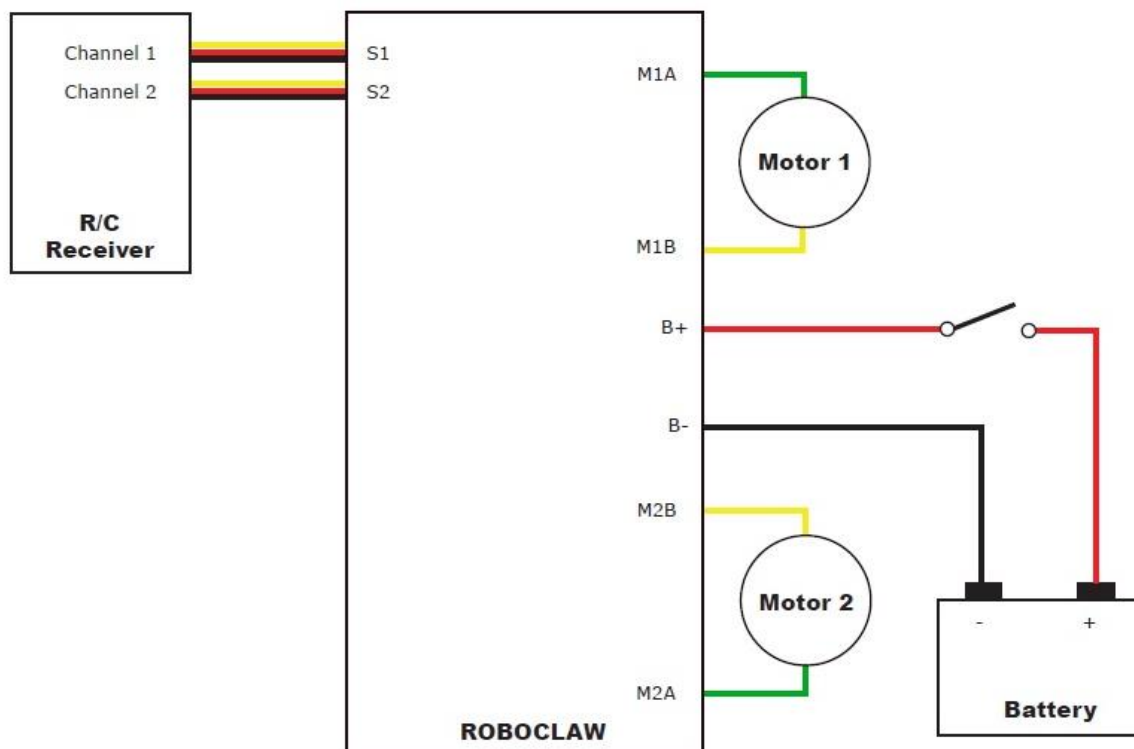


Here's how to wire the Intel Up or Raspberry (same pin) board from the Pixhawk "Telem 2" connector:



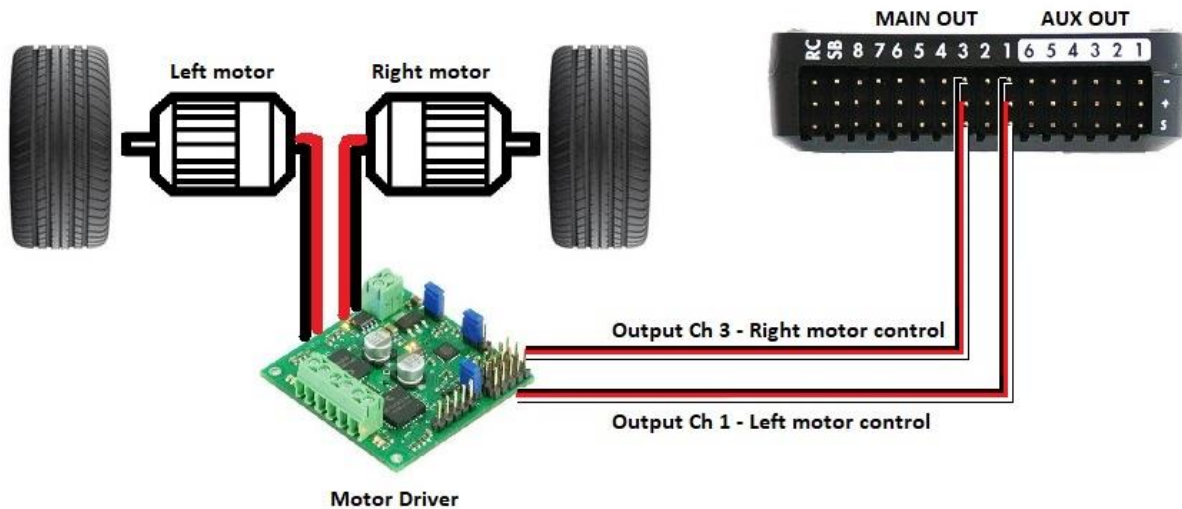
The red wire is the 5V output of the controller that connects to the Raspberry or Intel Up 5V input. In this case, it is the controller that powers the card. The black wire corresponds to the ground, and the yellow and green wires respectively correspond to the transmission and reception signal.

Here are the connections to make on the Roboclaw power board:



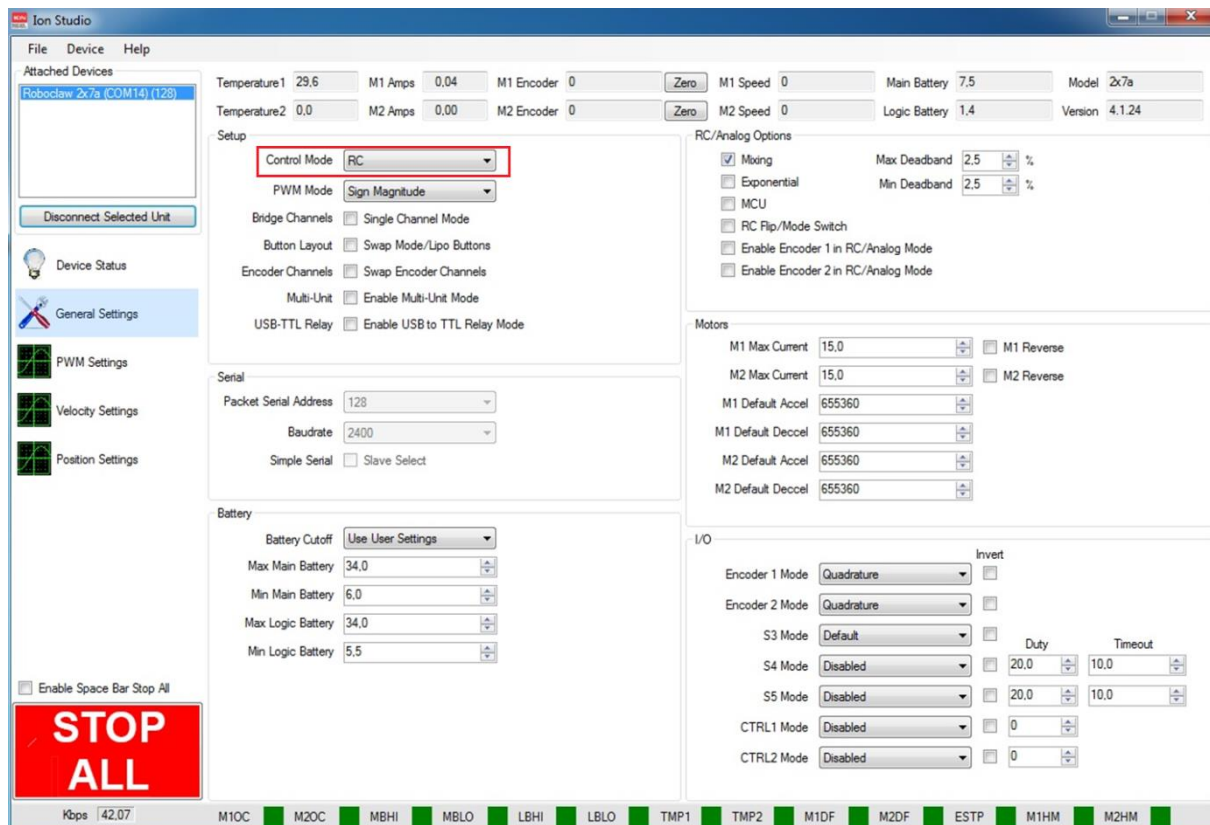
2 motors are connected to the M1 and M2 part, the power supply is connected to the B + and B- part and the output signal is on the S1 and S2 signals.

Unlike the previous image, do not connect to an RC receiver but the Pixhawk module. Connect Channel 1 to port "1" and Channel 2 to port "3" of Pixhawk.



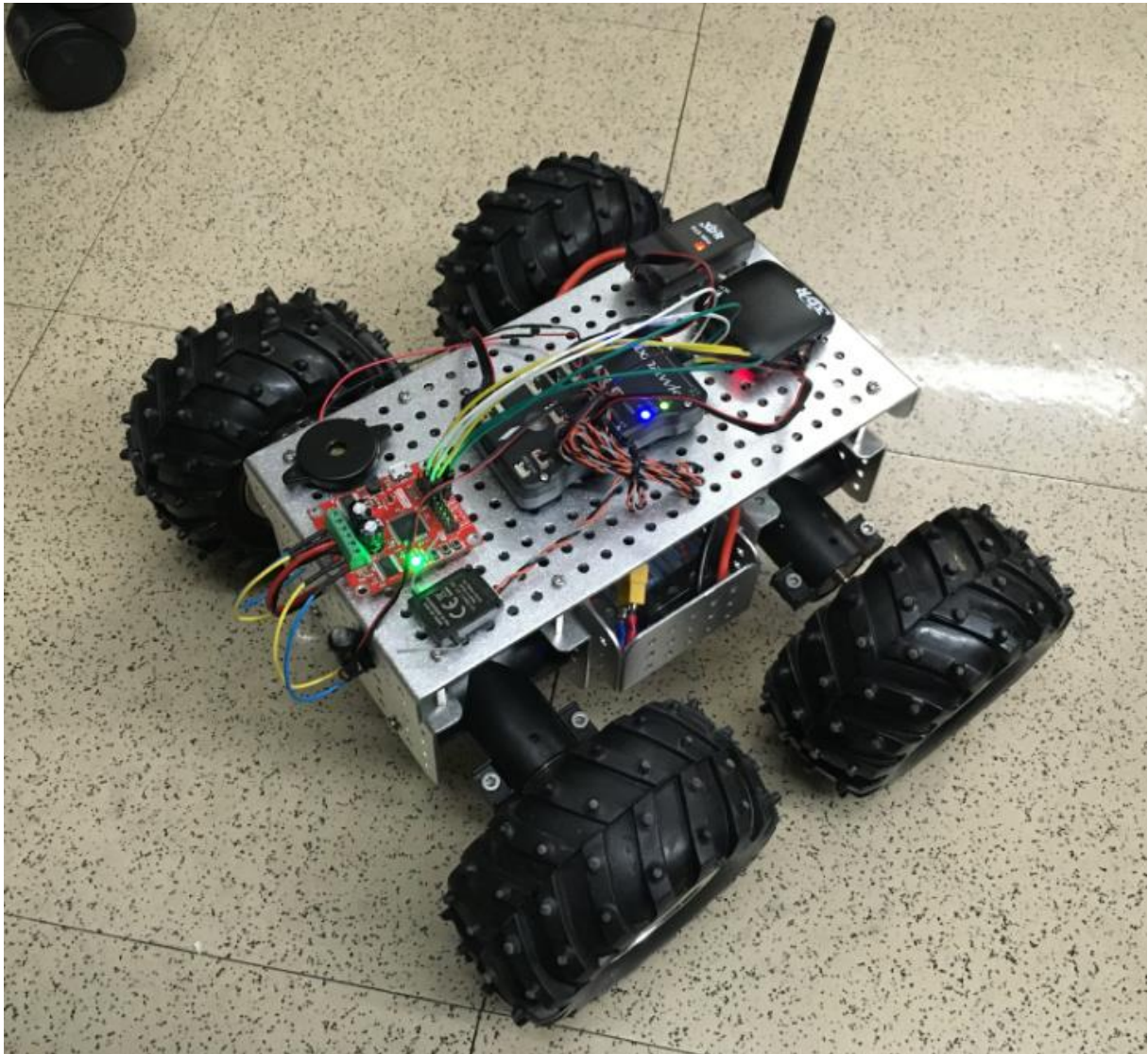
After connecting our motor and the battery, you have to connect the motor controller « Roboclaw » via USB to check if the control mode is the RC mode.

The Ion Studio software is used to configure the Roboclaw power board.



Download link: <http://www.basicmicro.com/downloads>

Here is the final result:



The powered robot and all the components are functional; we can do calibration and control RC after this point.

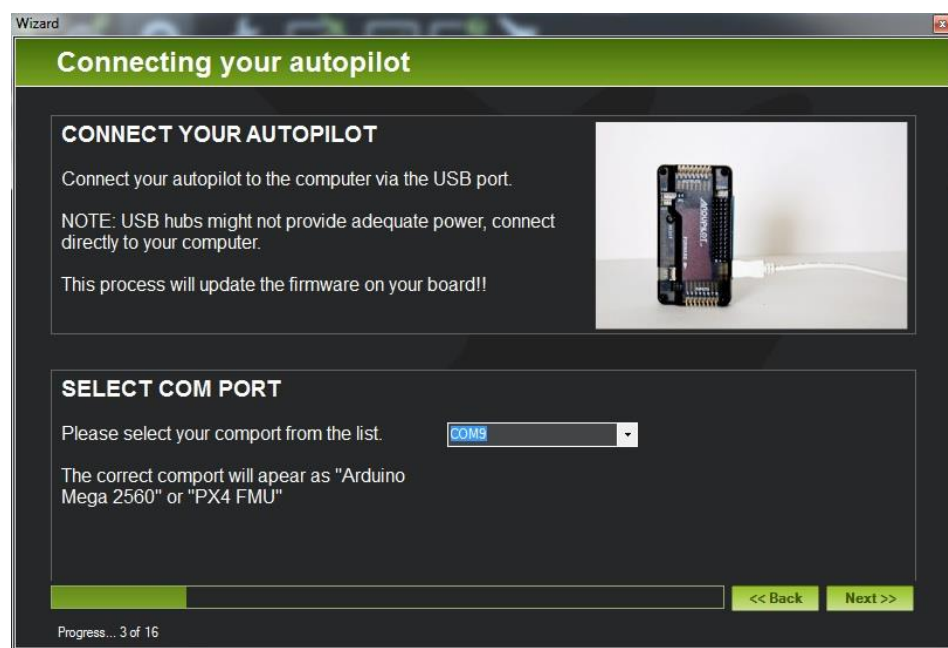
3) Initialization and control by RC

After wiring our Pixhawk with the power board and the motors, we need to perform the first installation and calibration on the rover. For this, we must follow these steps.

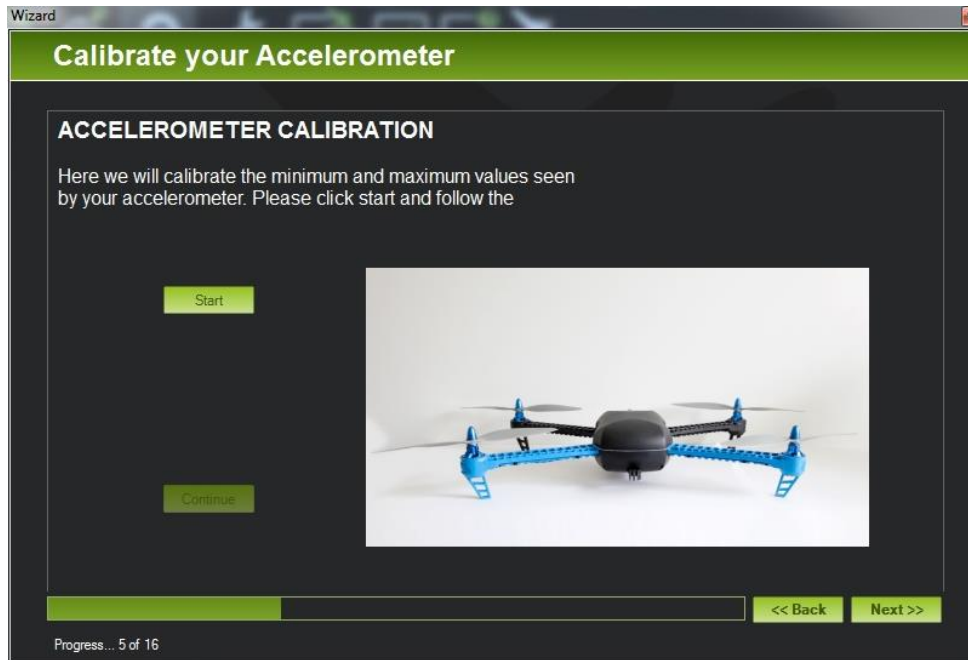
When you first start the Mission Planner software, we have a choice between different types of vehicles. In our case, you have to choose the Rover.



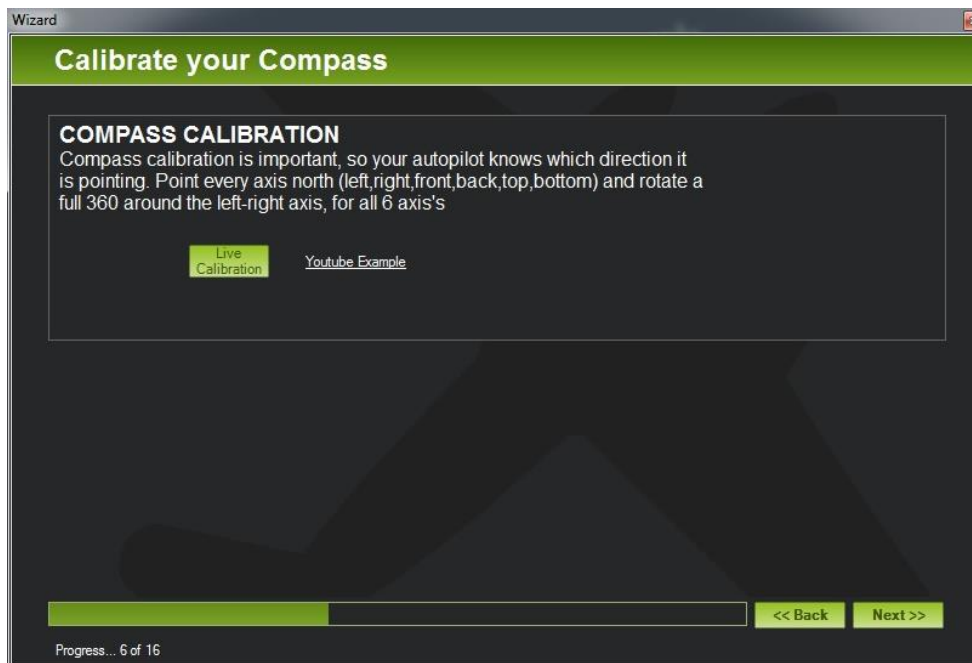
The software then asks us for the type of autopilot used. This is where you have to select the USB Com port corresponding to the Pixhawk module which is for me the Com9. An update of your autopilot will be done at this stage.

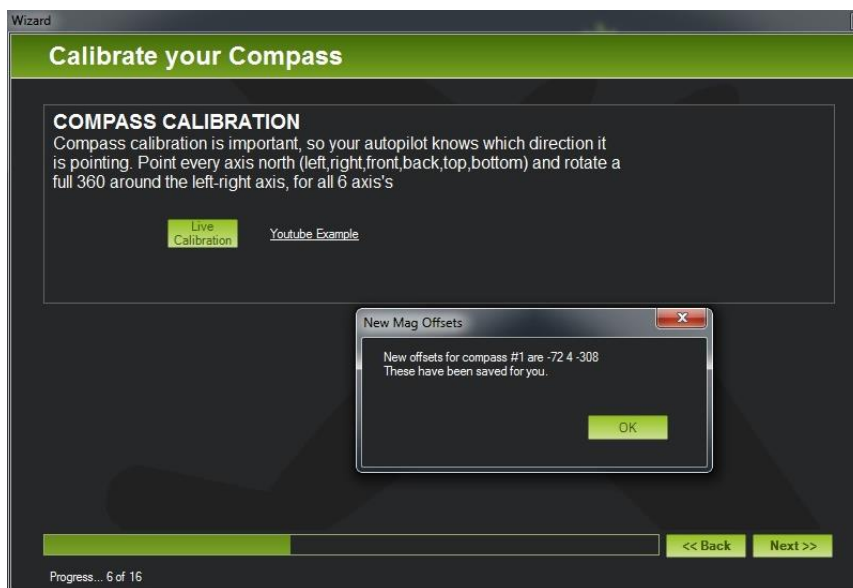
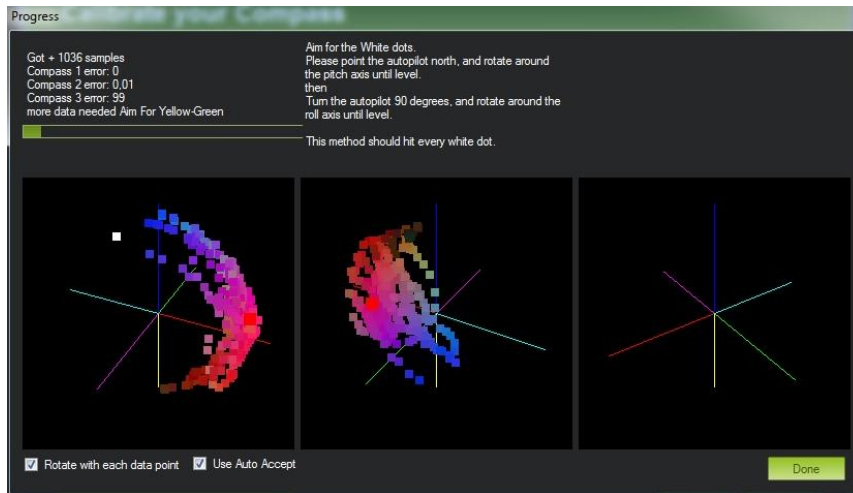


Then we asked to perform the calibration of the accelerometer by following all the images. We have different positions to respect such as placing the rover on the left side, then the right side...

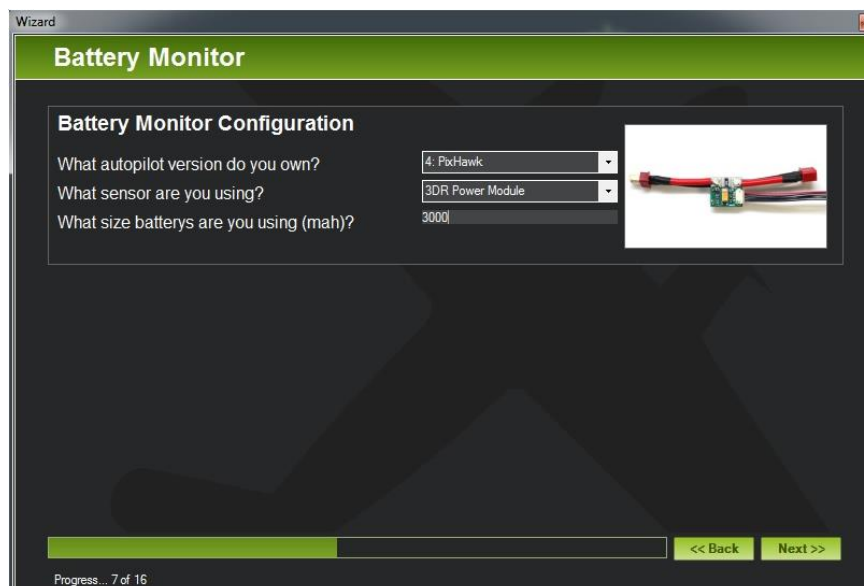


After installing the software on the Pixhawk module, we must calibrate the compass to initialize the axis of the rover.

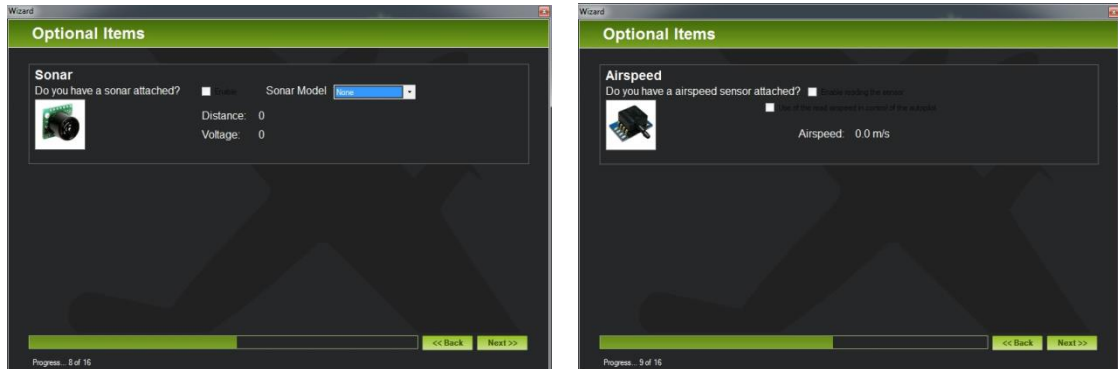




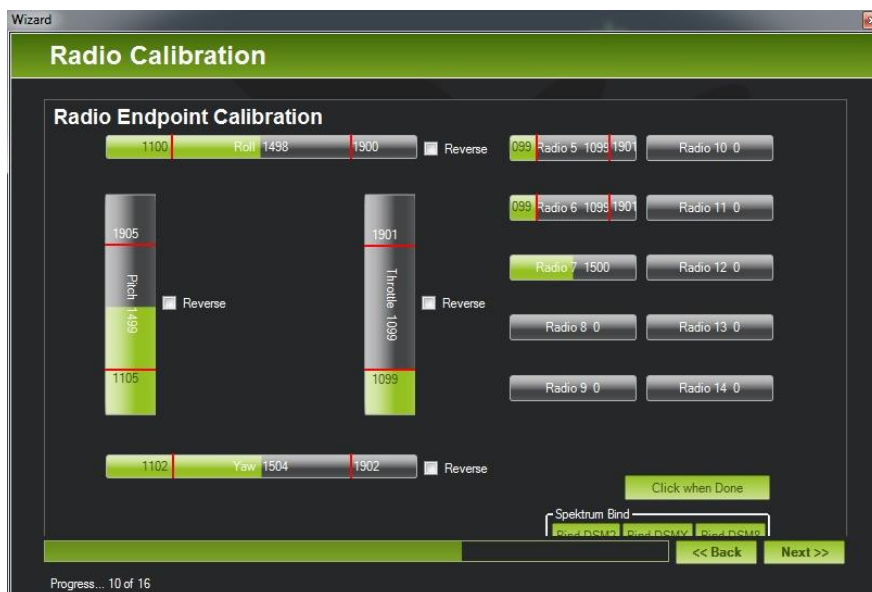
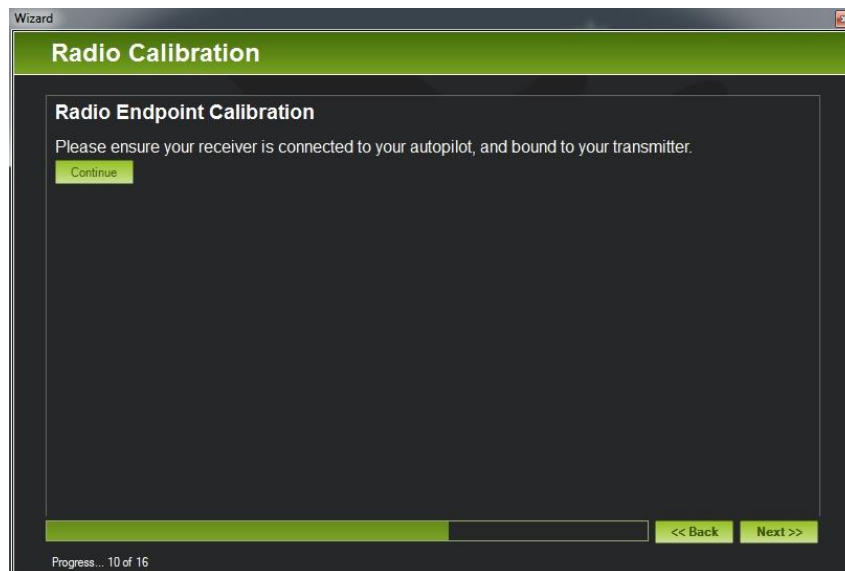
After calibrating the compass, it is necessary to define the type of battery and cable used for the rover.



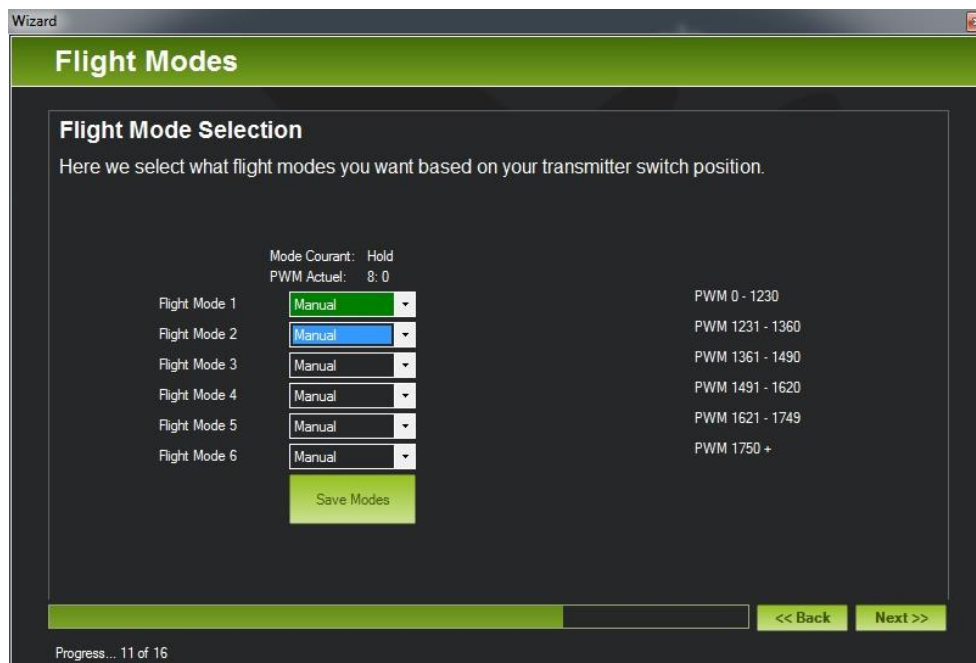
To have more precision, Mission planner asks us if we added the sonar and airspeed sensors to the Rover. Do not check if you do not have.



The next step is the calibration of your RC Remote Control.



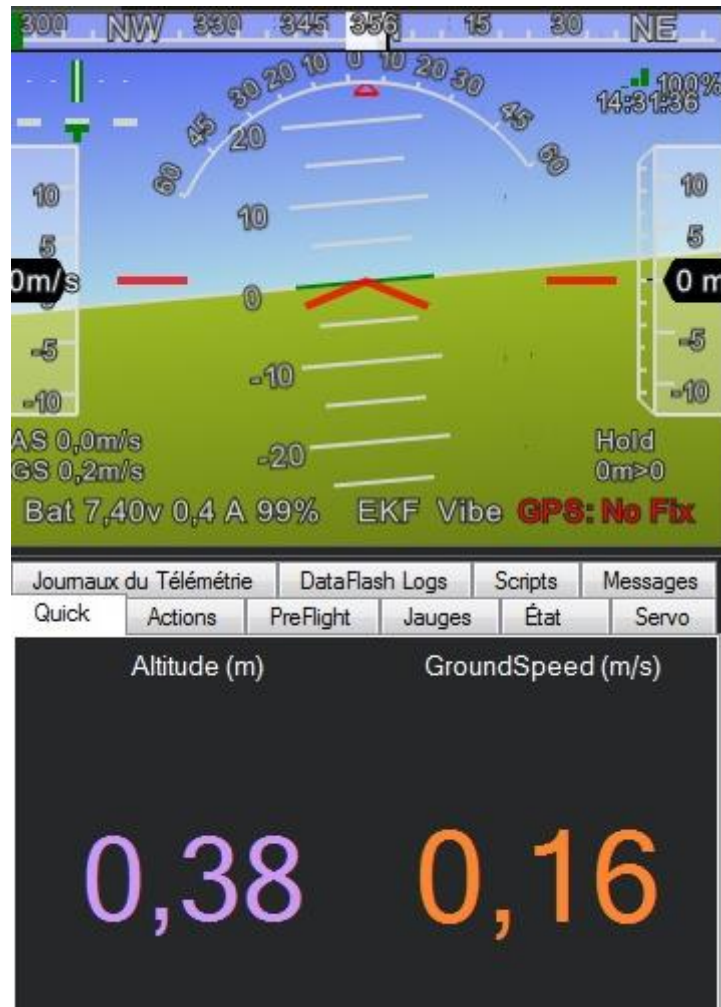
The last configuration is on driving modes. Leave the default modes.



After completing all the configurations, you must establish the link between the RC command and the motor pins that were previously connected to Pixhawk. It has RCIN3 which corresponds to the left joystick, it allows going on a straight line with our rover and RCIN1 which corresponds to the right joystick, it allows rotating the rover on left or right.



It is now possible to control the rover from the RC remote control. Do not forget to manually disarm the system with a long press on the manual switch (must see a red led fixed). During use, we can observe the different characteristics in real time of our Rover.



4) Installation and test software with Intel Up

We want to control our Rover from our Intel Up board. For this, we need to start installing the libraries that will allow the Intel Up to communicate with the flight controller.

We start with the recovery and installation of the MAVLink and MAVproxy libraries. For this we need to update the list of required libraries:

```
sudo apt-get update
```

Install the required libraries and utilities needed to install MAVLink and MAVproxy:

```
sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv
python-pip python-numpy python-dev libxml2-dev libxslt-dev python-lxml
```

Finally, get and install the MAVLink, MAVproxy libraries:

```
sudo pip install future
```

```
sudo pip install pymavlink
```

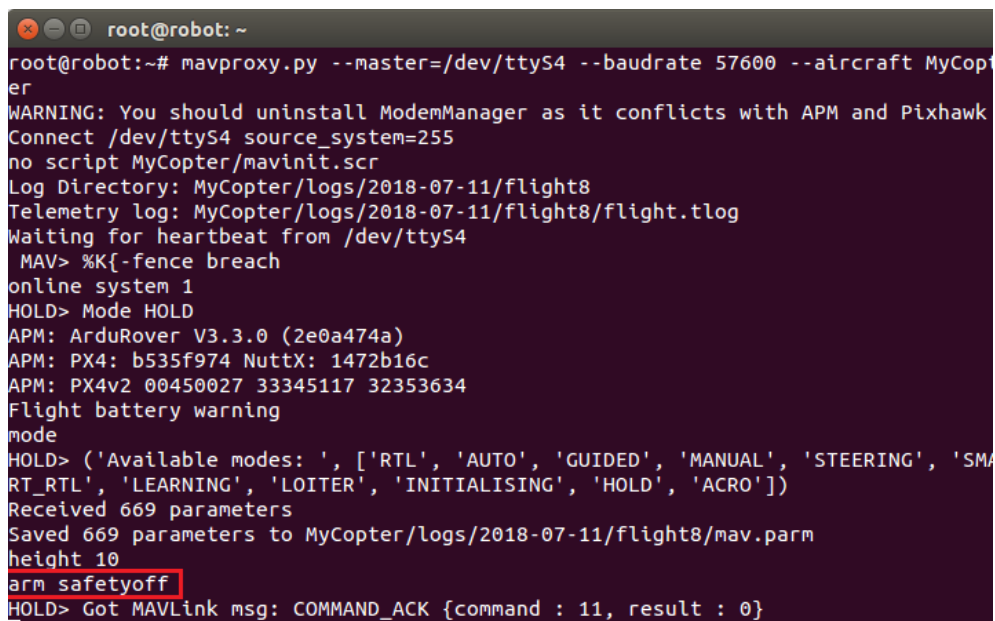
```
sudo pip install mavproxy
```

It is now possible to make a connection test between the Intel Up and the flight controller PixHawk. Enter the following command line:

```
sudo -s
```

```
mavproxy.py --master=/dev/ttyS4 --baudrate 57600 --aircraft MyRover
```

For this example, I disarmed the Pixhawk without pressing the manual security button:



```
root@robot: ~
root@robot:~# mavproxy.py --master=/dev/ttyS4 --baudrate 57600 --aircraft MyCopter
WARNING: You should uninstall ModemManager as it conflicts with APM and Pixhawk
Connect /dev/ttyS4 source_system=255
no script MyCopter/mavinit.scr
Log Directory: MyCopter/logs/2018-07-11/flight8
Telemetry log: MyCopter/logs/2018-07-11/flight8/flight.tlog
Waiting for heartbeat from /dev/ttyS4
  MAV> %K{-fence breach
online system 1
HOLD> Mode HOLD
APM: ArduRover V3.3.0 (2e0a474a)
APM: PX4: b535f974 NuttX: 1472b16c
APM: PX4v2 00450027 33345117 32353634
Flight battery warning
mode
HOLD> ('Available modes: ', ['RTL', 'AUTO', 'GUIDED', 'MANUAL', 'STEERING', 'SMART_RTL', 'LEARNING', 'LOITER', 'INITIALISING', 'HOLD', 'ACRO'])
Received 669 parameters
Saved 669 parameters to MyCopter/logs/2018-07-11/flight8/mav.parm
height 10
arm safetyoff
HOLD> Got MAVLink msg: COMMAND_ACK {command : 11, result : 0}
```


We can see that the command has been executed. The master argument is the port used to connect to the flight controller. In the case of an Intel Up on Ubuntu 14.04, we use the serial port which is /dev/ttyS4. The baud rate argument corresponds to the baud rate on the port. The last argument is the folder name that will be created to store logs and telemetry data.

The connection is functional; you will be able to pass the information from the controller to the Mission Planner software on computer.

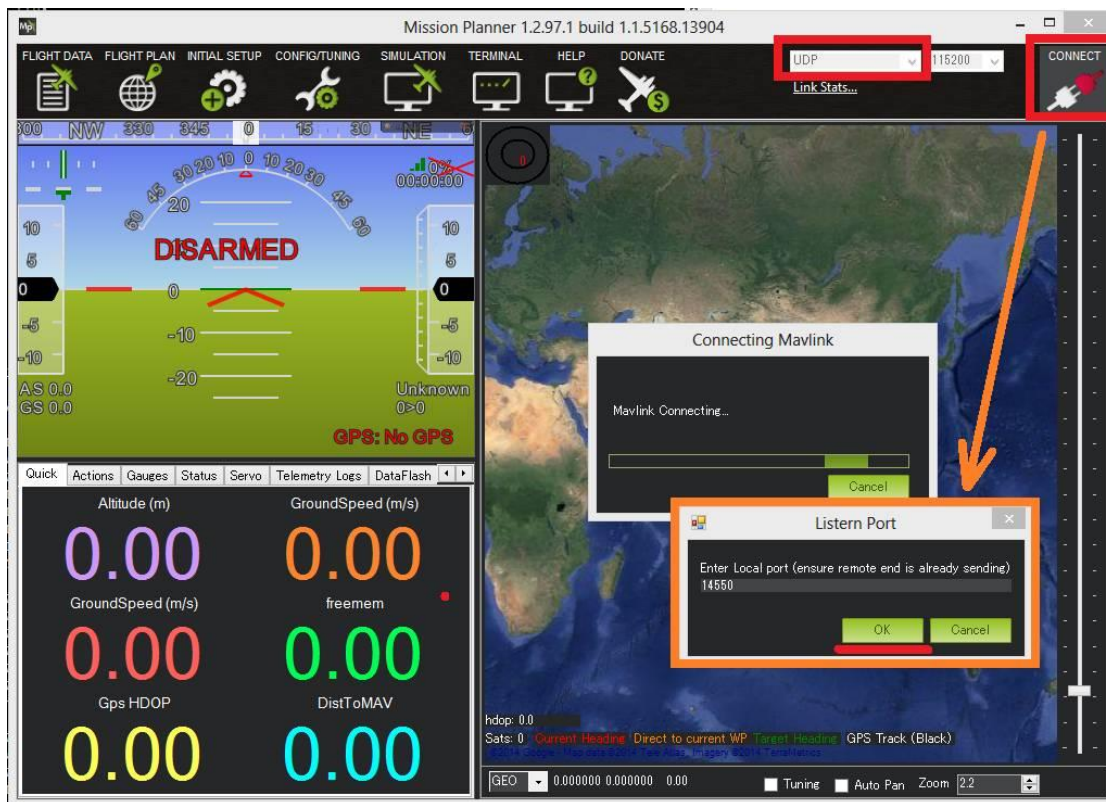
To be able to redirect the information to the computer where Mission Planner is installed, it is necessary to get its ip address with the ipconfig command.

After retrieving the IP address, log in SSH on the Intel Up connected to the flight controller Pixhawk, then type the following command:

```
mavproxy.py -master=/dev/ttyS4 -baudrate 57600 -out 192.168.1.11:14550 -aircraft MyRover
```

Here the "out" argument is the IP address of the computer where Mission Planner is installed.

On the computer, launch Mission Planner, then at the top right of the window, select from the drop-down menu "UDP", leave the default speed then click on "Connect". A popup window will ask for the UDP port number and then enter port 14550.



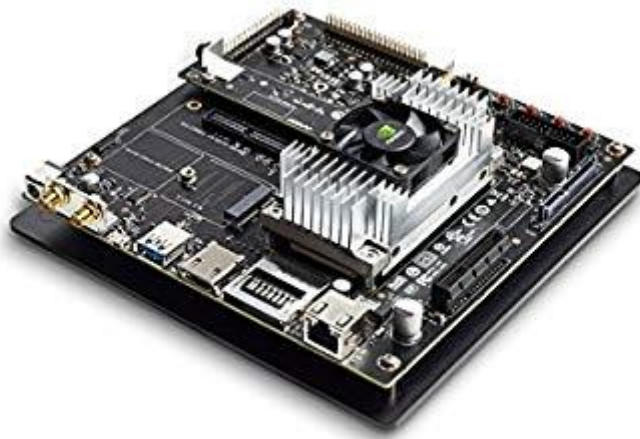
You now have access to all of the components and different actions for the rover from the computer.

Nvidia TX2 and ZED Camera

1) Product presentation

Jetson TX2 is the fastest, most power-efficient embedded AI computing device. The latest addition to the industry-leading Jetson embedded platform, this 7.5-watt supercomputer on a module brings true AI computing at the edge. It's built around an NVIDIA Pascal™-family GPU and loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth. It features a variety of standard hardware interfaces that make it easy to integrate it into a wide range of products and form factors.

The Jetson TX2 doubles the computing power and power efficiency of the earlier Jetson TX1.



The ZED Camera is a stereo depth sensor which contains two 4 megapixel imagers. For each frame, a composite of the left and right images (side by side) are sent over USB 3.0 to a host computer. The host computer then constructs a depth image using a GPU.



Stereolabs has a Github repository with a lot of code for project support, which is worth checking out.

2) Installation and result

When first booting the Nvidia Jetson, you have 2 command line to install Linux which is preloaded on the hard drive:

```
cd NVIDIA-INSTALLER
```

```
sudo ./installer.sh
```

Enter password: nvidia

After installation, you must update the jetson card by installing the development kit "Jetpack".

This action requires flashing the card, so be careful and especially not to disconnect the card during the operation to not make it unusable.

A person performs video tutorial details to install Jetpack:

<https://www.jetsonhacks.com/2017/03/21/jetpack-3-0-nvidia-jetson-tx2-development-kit/>

I advise you to follow this video for this step.

When installing the development kit is finished, you need to install ROS:

<https://www.jetsonhacks.com/2018/04/27/robot-operating-system-ros-on-nvidia-jetson-tx-development-kits/>

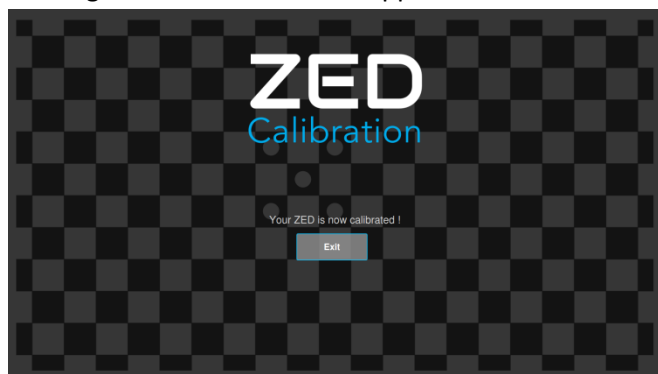
You will find above a detailed installation of ROS with a video tutorial on Jetson TX2.

Our nvidia Jetson is operational so we can install all the useful drivers to use our ZED camera.

<https://www.jetsonhacks.com/2017/04/07/stereolabs-zed-camera-nvidia-jetson-tx2/>

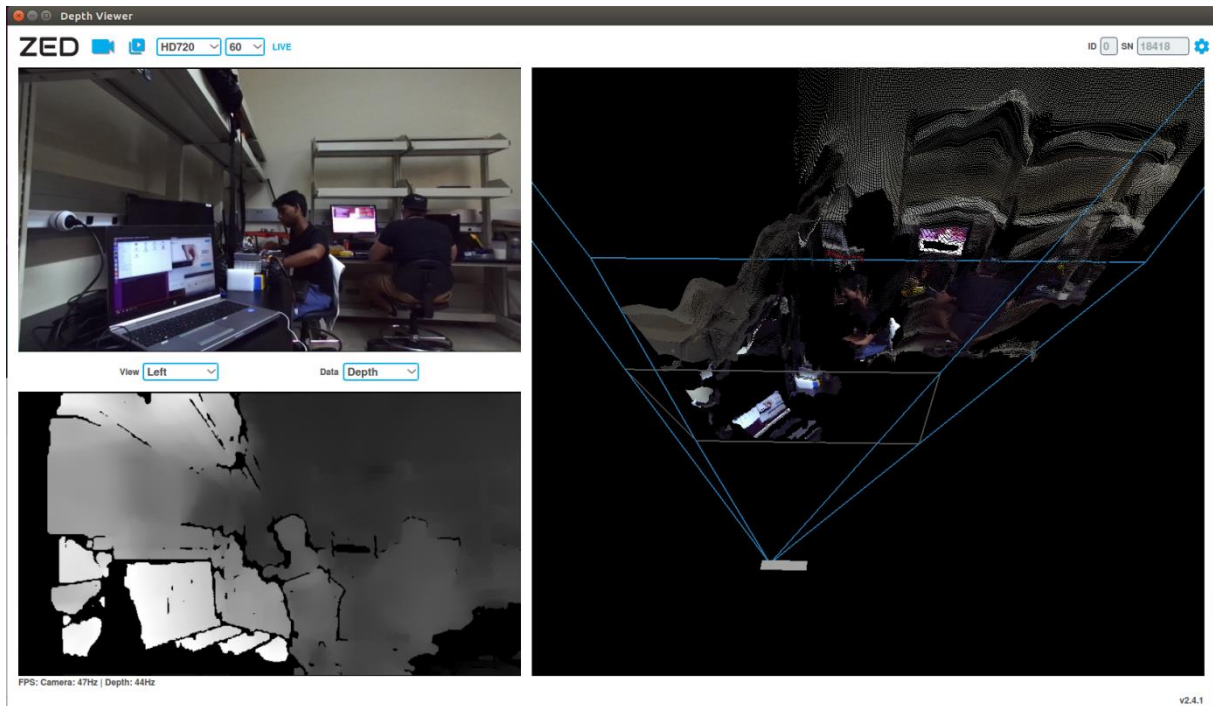
You will find above a detailed installation of Zed camera package with a video tutorial on Jetson TX2.

We have to start by calibrating the camera with the supplied calibration software:

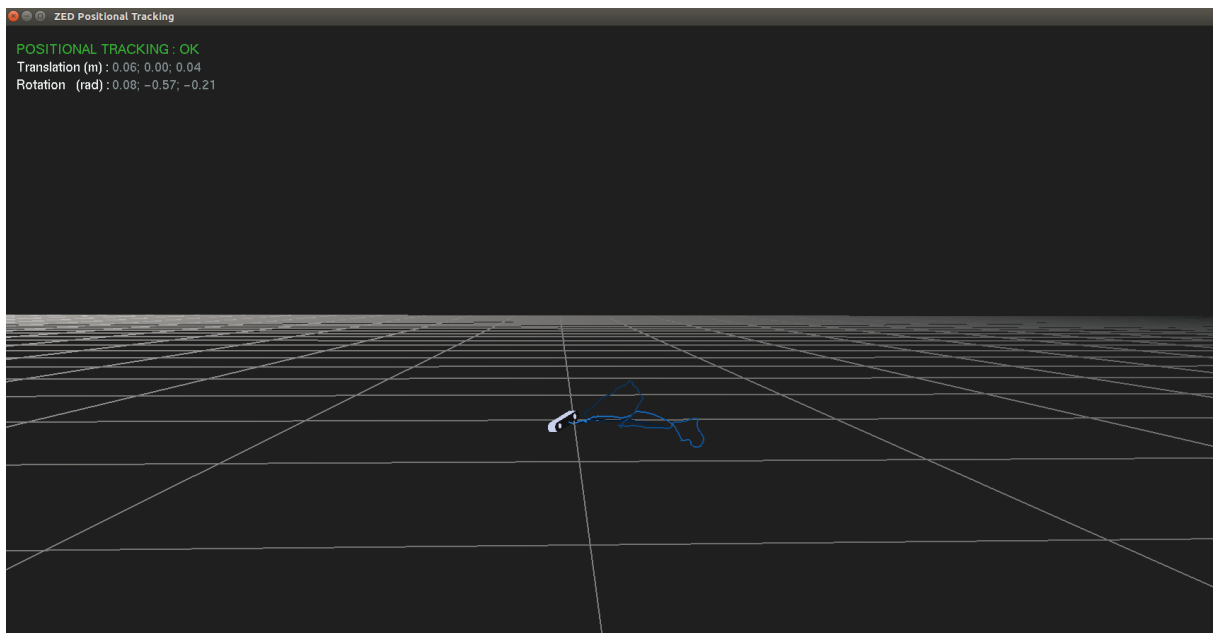


Here are the results obtained after installing packages and calibration.

We have software to map a place and define this position:



We have software to track the position of our ZED camera:



With this software and these examples, we can then use the camera to map a place and then define the position and the course of our rover.

Sources

Intel:

- <https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/realsense-camera-r200-datasheet.pdf>
- <https://software.intel.com/en-us/realsense/previous>
- <http://wiki.ros.org/RealSense>
- <https://01.org/developerjourney/installing-ubuntu-1404-lts-intel-realsense-robotic-development-kit>

Kinect:

- <http://sauravag.com/2016/10/how-to-setup-kinect-with-ros-and-rgbd-slam/>
- <https://answers.ros.org/question/196455/kinect-installation-and-setup-on-ros-updated/>
- <https://www.youtube.com/watch?v=QpNMJEAKX0>
- <http://roboticsweekends.blogspot.com/2017/12/how-to-connect-kinect-to-raspberry-pi-2.html>

Nvidia TX2:

- <https://www.jetsonhacks.com/2017/03/14/nvidia-jetson-tx2-development-kit/>
- <https://www.jetsonhacks.com/2017/03/21/jetpack-3-0-nvidia-jetson-tx2-development-kit/>
- <https://www.youtube.com/watch?v=D7lkth34rgM>
- <https://www.jetsonhacks.com/2018/04/27/robot-operating-system-ros-on-nvidia-jetson-tx-development-kits/>

ZED:

- https://www.youtube.com/watch?v=7_8XLI99dno
- <https://www.jetsonhacks.com/2017/04/07/stereolabs-zed-camera-nvidia-jetson-tx2/>

Lidar:

- <https://www.robotshop.com/media/files/pdf/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf>
- <https://blog.zhaw.ch/icclab/rplidar/>
- <http://wiki.ros.org/rplidar>
- https://github.com/robopeak/rplidar_ros
- <https://hollygood.wordpress.com/2015/12/01/ros-slam-2-hector-slam-2d%E5%9C%B0%E5%9C%96%E5%BB%BA%E7%BD%AE/>

Pixhawk:

- <http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html>
- http://downloads.basicmicro.com/docs/roboclaw_datasheet_2x7A.pdf
- <http://ardupilot.org/rover/docs/common-choosing-a-ground-station.html>
- <https://www.sqli-carrieres.com/job-news/do-it-yourself-construire-son-drone-partie-3/>
- <https://www.robotshop.com/media/files/pdf/datasheet-pixhawk.pdf>
- <https://www.youtube.com/watch?v=DGAB34fJQFc>