Platypus Boat Documentation

Release 0.1

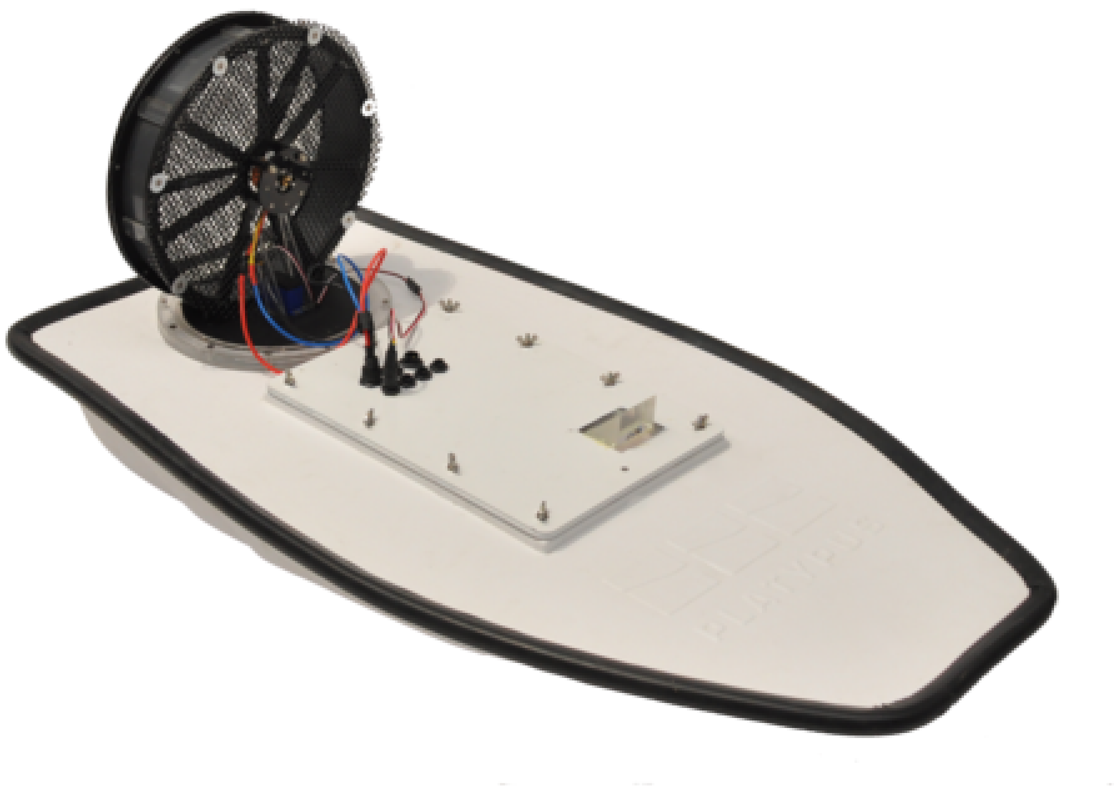
Alexandre Amory

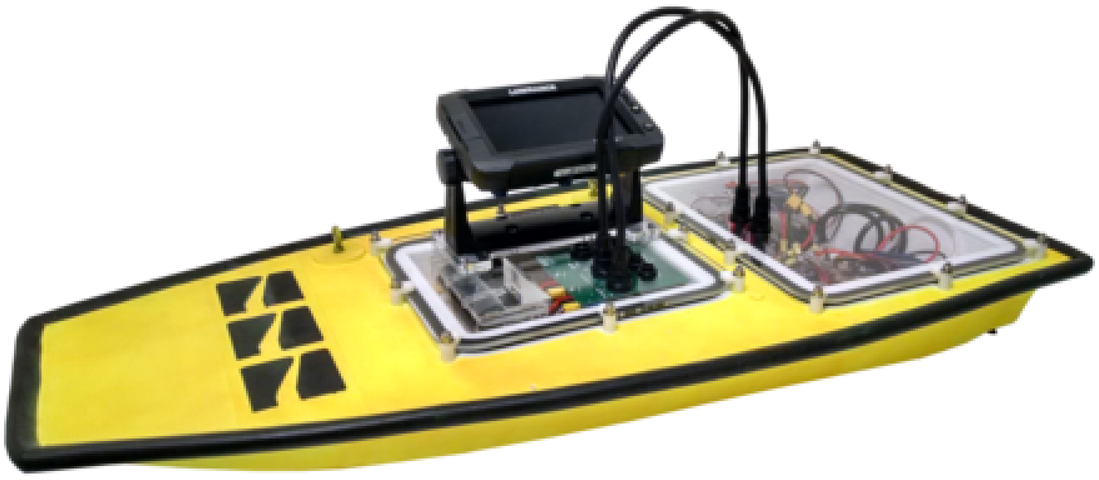
Apr 30, 2019

Contents

CHAPTER 1

About the Boats





CHAPTER 2

Autonomous Systems Laboratory - PUCRS

This document written by LSA describes the procedure to configure and use Platypus boats. LSA has three boats: two airboats and a prop boat. They are used for water monitoring and disaster mitigation applications.

CHAPTER 3

Summary

3.1 Getting Started

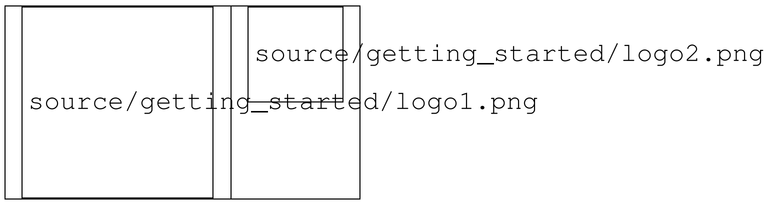
Before going into the tutorials, follow the instructions below to prepare your environment.

3.1.1 Bill of Materials for the Boats

Note: place here all the accessory/payload for the boats

Mandatory Resources

all missions require at least these parts.


source/getting_started/logo2.png
source/getting_started/logo1.png

Item name	photo	cost(US$)	link to provider
body row 1, column 1	source/getting_started/logo1.png	column 2	column 3
body row 2	source/getting_started/logo1.png		

Optional Resources

depending on the mission's goals, these other resources are also available.

source/getting_started/logo1.png

source/getting_started/logo2.png

Item name	photo	cost(US$)	link to provider
body row 1, column 1	source/getting_started/logo1.png	column 2	column 3
body row 2	source/getting_started/logo1.png		

3.2 ROS Quick Start Tutorials

In this page, we will provide you a quick start with ROS.

There are three parts of the ROS Quick Start tutorials. In the first part, you will get an understanding about ROS main concepts. Then, you will learn how to develop your first program with ROS, namely programming a publisher and subscriber using both C++ and Python. Finally, some more advanced concepts will be introduced through the development of a cleaning application with the Turtlesim simulator. At the end of ROS Quick Start tutorials, you will get a full understanding of ROS that allows you to dive deeper and develop more advanced robotics applications.

Warning:

Before starting these tutorials, it is assumed that you have a very basic knowledge about ROS environment and setting-up a ca

- Installing and Configuring Your ROS Environment.
- Navigating the ROS Filesystem.
- Creating a ROS Package.
- Building a ROS Package.

You can also start with watching the tutorials of *ROS Background*

3.2.1 ROS Background

If you are new to ROS, it is recommended to start with taking these online tutorials that will introduce you to the basic concepts of ROS.

Warning: These tutorials assume that you have already installed ROS on your computer and that you have configured your environment. You still haven't done yet, follow in the installation instructions of ROS Indigo, then you must configure your ROS environment and finally create a ROS package to create and run your programs.

Note: Although some videos are shown for ROS Hydro, they are also valid for all subsquent versions including Indigo and Jade, as they deal with the basics.

Lecture 1: Start a ROS Master Node

Tip: In this video, you will:

- Learn how to start the main node in ROS, called Master node
- Understand the default workspace
- Configure your ROS environment in the *.bashrc* script file
- Find the list of ROS nodes and ROS topics

Listen to the following video.

Lecture 2: Understanding ROS Topics, Nodes and Messages

Tip: In this video, you will:

- Work with the Turtlesim simulator to understand ROS fundamental concepts
- Understand ROS topics and its related terminal commands
- Understand ROS nodes and its related terminal commands
- Undertsand ROS messages and its related terminal commands

Listen to the following video.

Review Questions

- What is the command to find the list of all ros nodes?
- What is the command to find the list of all ros topics?
- What is the topic that tells about the position of the turtle?
- What is the topic that sends command to the turtle to make it move?
- What is the node used to send velocity commands to the turtle?
- What is the command that allows to see the type of message for a particular topic?
- Write is the command that allows to publish velocity command to the turtle with a linear velocity 1.0 and angular velocity 0.5.

Online Test

Pass the online quiz for the above review questions

3.2.2 ROS Programming

In this tutorial, you will learn how to start programming with ROS. It's funny and amazing, yet quite simple. The first thing you need to understand in ROS is how to subscribe to a ROS topic, and how to publish a message to a ROS topic as most of ROS operations are based on publishing and subscribing to topics.

Warning: It is assumed that you already know the main concepts on ROS including ROS topics, ROS nodes, ROS messages and ROS services. If not, you need to first take the first tutorial on *ROS Background*.

The code of this tutorial is available in `src/ros_basics/talker_listener/` of gaitech_edu package.

Note: The following tutorial is based on the ROS tutorial Writing a Simple Publisher and Subscriber (C++) and Python but provides more practical programming and configuration hints. Although some videos are shown for ROS Hydro, they are also valid for all subsquent versions including Indigo and Jade, as they deal with the basics.

Lecture 1: ROS Publisher and Subscribers in C++ and Python

In this video, you will learn:

- How to develop the simplest program with ROS
- How to publish a message to a ROS topic using C++ and Python
- How to subscribe a message to a ROS topic using C++ and Python
- How to configure `CMakeLists.txt` and `package.xml` to add new required packages for compilation and runtime
- How to run a program in ROS

Review Questions

- Write the C++/Python instruction that creates a new topic called chatter of type String?
- Write the C++/Python instruction that effectively publishes a message on the chatter topic?
- How to add a new exectuable in `CMakeLists.txt` and `package.xml` to be able to compile and run a C++ ROS program?
- How to declare a topic subscriber in C++/Python?
- What is the role of the subscriber callback function?
- in C++, what is the difference between `std_msgs::String::ConstPtr` and `std_msgs::String`?

Lecture 2: ROS Services in C++ and Python

Coming soon ..

3.2.3 Turtlesim Cleaning Application

In this tutorial, you will learn more advanced concepts on ROS by developing a first application on ROS using the Turtlesim simulator. The objective of the application is to emulate a cleaning application like Vaccum cleaning robots (e.g. Roomba). The robot should cover the whole area to be cleaned. For this, we will develop, step-by-step, several functions to make the robot move straight and rotate and use these functions in developing the cleaning application. You can then extend this application to make it even smarter and more comprehensive.

The code of this tutorial is available in `src/turtlesim/cleaning_app/robot_cleaner.cpp` of gaitech_edu package.

Warning: It is assumed that you already know the main concepts on ROS including ROS topics, ROS nodes, ROS messages and ROS services. If not, you need to first take the first tutorials on *ROS Background* and *ROS Programming*.

Note: The following tutorial aims at introducing necessary and fundamental concepts of ROS beyond the simple talker/listener tutorial, like navigation, motion control, distance estimation, rotated angle estimation, and some ROS packages including TF. This tutorial consists of a series of fives videos that you need to watch in order to get the main lessons and outcomes. The tutorial could be completed in 2 to 4 hours.

Warning: It would be appreciated if you already have some background knowledge on introductory robotics courses. As pre-requisite, you should already know what a 2D/3D frame is, and what a transformation between frame is, in addition to basic knowledge on 2D kinematics.

Lecture 1: Introducing the Cleaning Application

In this video, you will:

- understand the objectives and tasks of the turtlesim cleaning applications
- recognize the different functions to be developed for the clearning application

Lecture 2: Moving in a Straight Line

In this video, you will:

- develop a function to make the robot move in a straight line forward and backward
- understand how to choose the right ROS topic to publish a message for a certain functionality
- use the Twist message to send linear velocity commands to move in straight line
- control the distance traveled by the robot

Lecture 3: Rotating Left and Right

In this video, you will learn:

- understand rotation conventional assumptions
- develop a function to make the robot rotate left and right
- use the Twist message to send angular velocity commands to rotate
- set the desired orientation of the robot after rotation
- develop and use some basic functions related to rotation

Lecture 4: Go to Goal Location (PID Controller)

In this video, you will:

- understand the essential of PID controllers
- develop a PID controller to make the robot head towards a specified location

For a good introduction on PID controllers for mobile robots, it is recommended to watch Lecture 1 and Lecture of the online course on Control of Mobile Robots, provided by Georgia Institute of Technology.

Lecture 5: Grid and Spiral Cleaning Application

In this video, you will:

- use the move and rotate functions to develop the clearning applications and area coverage
- develop a new function to make the robot cover the area in spiral form

Review Questions

- What are the steps followed to develop the cleaning application?
- Explain how the Twist message is used to make the robot move stright and rotate?
- What is the equation used to make the robot move in spiral form? How this implemented in ROS?
- What is the drawback of method used to control the traveled distance and rotated angle? Explain and justify your claim.

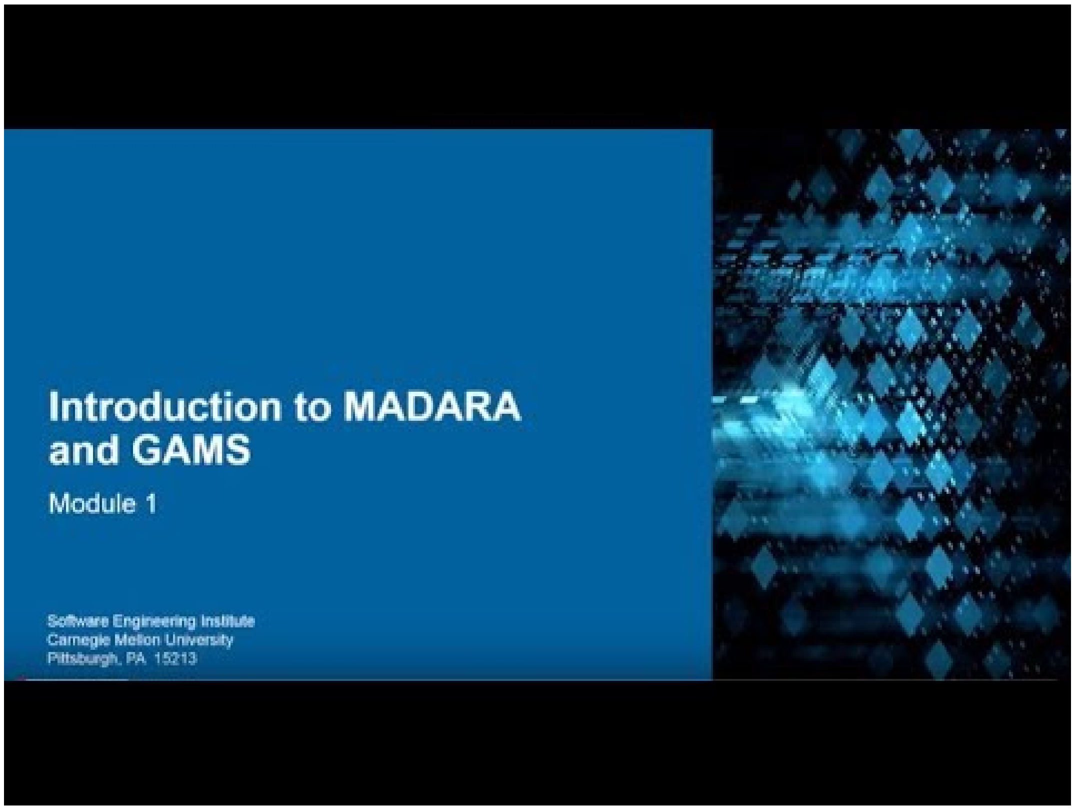
Do-It-Yourself

You are requested to extend the cleaning application by making it smarter. We want to program the robot such that it moves 1 meter, then rotates 360 degrees in place, and repeat the process until the area is cleaned. You need to use a loop to control the robot motion until the end of the mission.

3.3 GAMS/MADARA Quick Start Tutorials

In this page, we will provide you a quick start with GAMS/MADARA.

Here, we focus into practical use of those tools, to formal concepts, we advise you to read papers available into internet. Also, we can watch some videos online in folowwing link:



Introduction to MADARA and GAMS

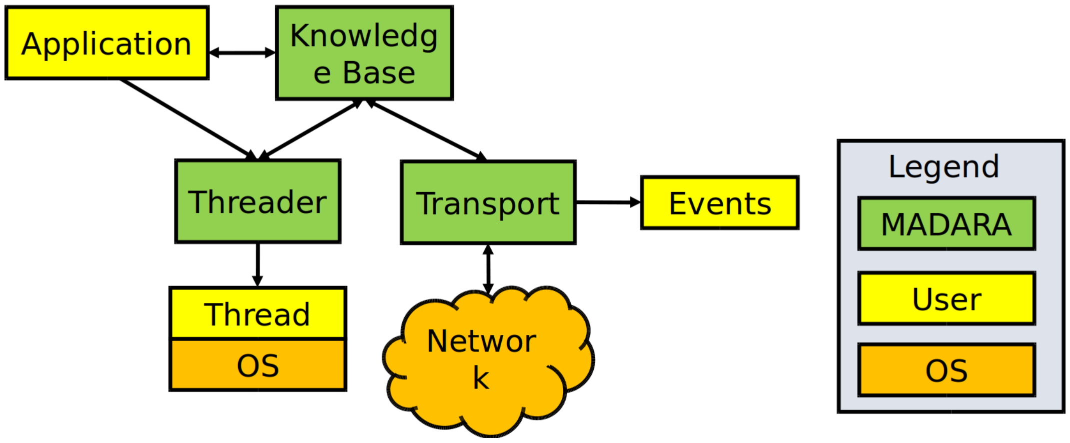
Module 1

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

3.3.1 GAMS/MADARA Background

In this page, we will provide you a quick start with GAMS/MADARA.

GAMS / MADARA combines the Group Autonomy for Mobile Systems (GAMS) project and the MADARA middleware. MADARA is a communication infrastructure between mobile robots that offers network transport services, threads and automatic sharing of knowledge between agents. The network transport is done via UDP with 3 types of messages: unicast, broadcast and multicast. GAMS provides high-level functionality, such as area coverage planning algorithms, and portable platforms for research in Artificial Intelligence with robotics and multi-agent systems. Researchers can develop algorithms in C ++ and JAVA so that the integration between heterogeneous agents is made easier, since GAMS was created on top of the MADARA project. The GAMS / Madara integrates the advantages of MADARA, a middleware for distributed systems with the convenience offered by the GAMS to develop algorithms for easy interaction between heterogeneous robots. Figure below shows the block diagram of MADARA. MADARA consists of three components: Knowledge base, threader and transport.

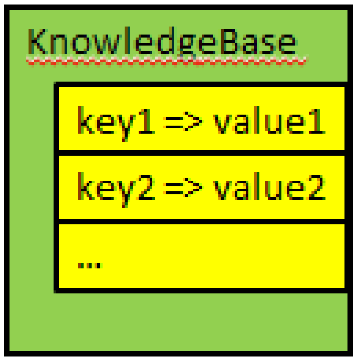


The Knowledge base is essentially the world model generated by each agent. Information about mission, location, orientation, and robot type (real or simulated) are populated on the basis of knowledges. Each agent mounts its base of knowledges according to the environment data obtained by the sensors. The data remains at the base as a mapping of all the stimuli captured by the platform.

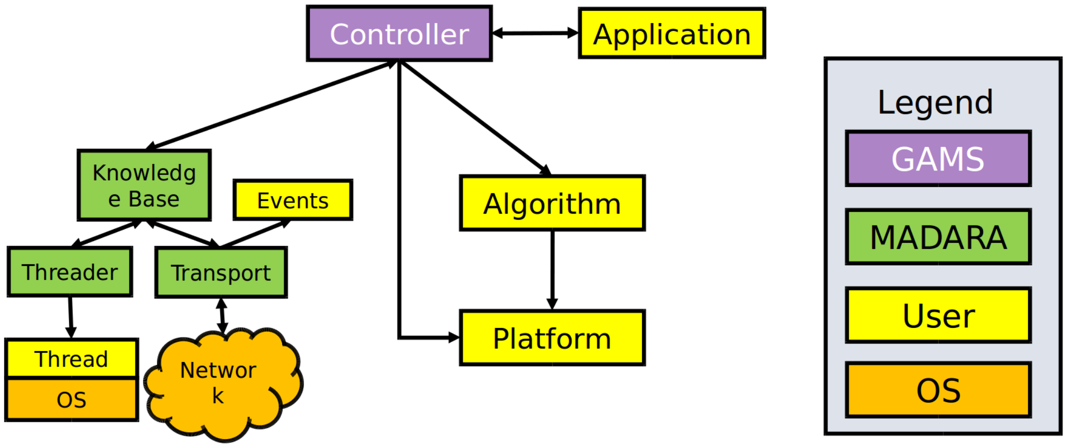
The transport block is responsible for transferring knowledges through the agent network, allowing the exchange of information between the *knowledges* of each agent. Since the threader allows the programmer to create threads similarly to libraries threads, such as threads STL, boost threads, among others. The advantage of the threader is that it comes integrated with Knowledge base, i.e., each thread contains a reference to Knowledge Base, which is shared, updated by all threads of the agent.

Thus, with the integration of threads, transport services and knowledge services, MADARA stands out by allowing the sharing of data between agents and within the agent itself. The data are distributed transparently to the programmer, i.e., that sets new values and MADARA responsibility by grouping them in UDP packets and decide the best time to send them over the network, which tends to improve use. In addition, each variable mapped on the basis of Knowledge already offers automatically critical region protection through mutexes hence the user can view and change the values in different threads without worry with racing conditions.

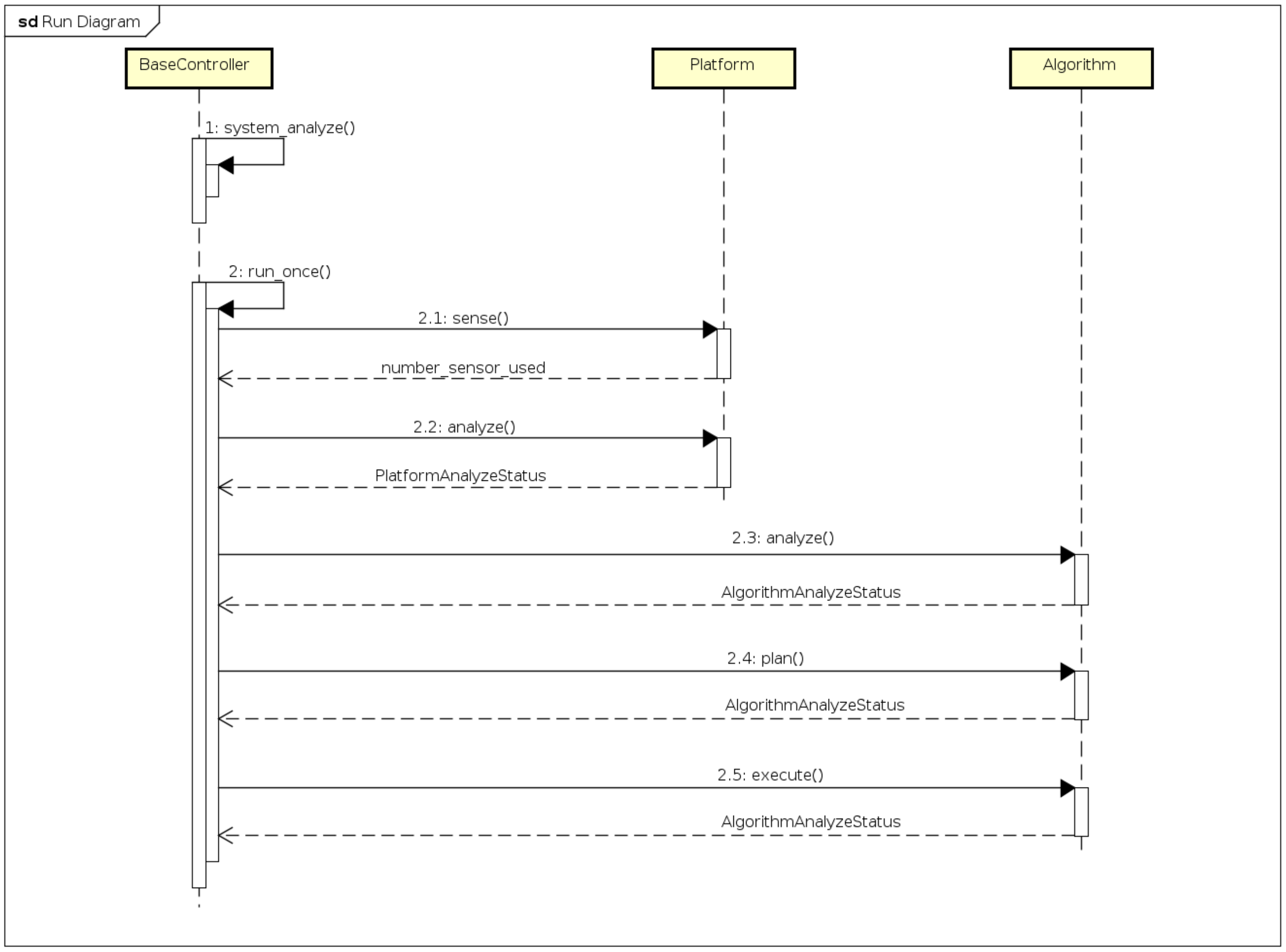
As can be seen in figure below, the variables are mapped to Knowledge Base by a tuple: key and value. Each key must be unique and allows the programmer to identify a value stored in knowledge. The programmer can invoke explicit queries via the functions call, or you can tell that a variable of the type container is related to a specific key of the Knowledge Base. In the last case, whenever the value is updated, the variable will be automatically updated. The shape of the key has meaning and helps to identify if the value is private or, in case of being sent to other agents, it allows identifying who belongs to that value. Private variable keys begin with a dot, whereas a public variable of the number 4 agent must be prefixed with `agent.4.`.



However, just set a variable is public does not guarantee that it will be delivered to other agents, you must configure the transport module, stating the type of message (unicast, multicast and Broadcast), the IP address and the port of each agent that should receive the information.



Above, you can see GAMS diagram, which is build on top of MADARA middleware. GAMS inherits all characteristics. Algorithms developed by users are executed by `Controller` by running MAPE Loop. So, the main loop of GAMS can be resumed with following sequence diagram:



where the MAPE process is mapped as:

- Monitor phase: platform sense;
- Analyze phase: platform analyze, algorithm analyze;
- Plan phase: algorithm plan;
- Execute phase: algorithm execution.

The possible values of `PlatformAnalyzeStatus` are:

- UNKNOWN = 0,
- OK = 1,
- WAITING = 2,
- DEADLOCKED = 4,
- FAILED = 8,
- MOVING = 16,
- REDUCED_SENSING_AVAILABLE = 128,
- REDUCED_MOVEMENT_AVAILABLE = 256,
- COMMUNICATION_AVAILABLE = 512,

- SENSORS_AVAILABLE = 1024,
- MOVEMENT_AVAILABLE = 2048

The possible values of `AlgorithmAnalyzeStatus` are:

- UNKNOWN = 0x00000000,
- OK = 0x00000001,
- WAITING = 0x00000002,
- DEADLOCKED = 0x00000004,
- FAILED = 0x00000008,
- FINISHED = 0x00000010

There are many algorithms to be used in GAMS/MADARA.

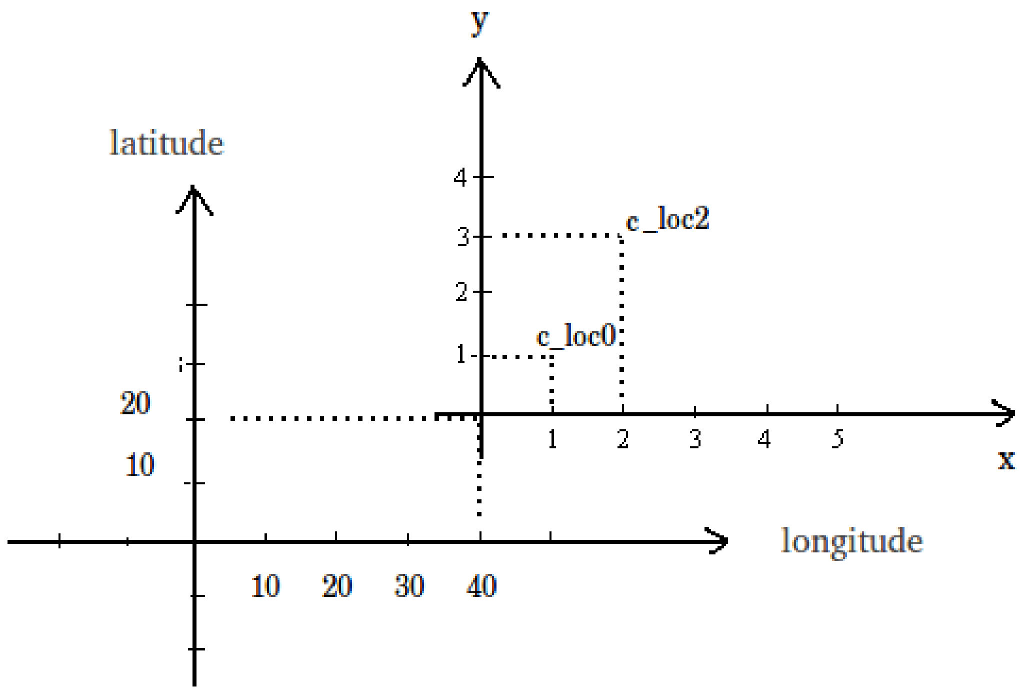
- Formation coverage
- Prioritized Region Coverage
- Minimum Time Coverage
- Serpentine Coverage
- Waypoints
- Formation Follow
- Synchronized Formations
- Convoy Shielding
- Line Defense
- Arc Defense
- Onion Defense
- Executor

Coordinate systems

GAMS supports two types of coordinate systems: GPS and cartesian. Each coordinate system can have a father, so you can create a tree of coordinate systems. For example, you can specify that an cartesian coordinate system (named `cartesian0`) is child of a GPS frame by writing the following code:

```
gams::pose::GPSFrame gps_frame;
gams::pose::Position gps_loc(gps_frame, 40, 20);
gams::pose::CartesianFrame cartesian0(gloc);
gams::pose::position c_loc0(cartesian0, 1, 1);
```

In the code above, to create a cartesian frame (named `cartesian0`) you have to define a position into the `gps_frame`. After that, you can create a point in the cartesian frame, by informing location (1,1) in relation to `gps_frame`. They will look like this:



Also you can convert between the coodinate systems. For example, to convert the position (2,0) from cartesian frame (`cartesian0`) to `gps_frame`, you should write:

```
gams::pose::Position c_loc2(cartesian0, 2, 3);
gams::pose::Position gps_loc2 = c_loc2.transform_to(gps_frame);
```

Also, you can calculate the distance between two points even if they are in different coordinate file system. The only restrition is that their file systems must be related.

```
double distance = gps_loc.distance_to(c_loc2);
```

3.3.2 Algorithms

There is many algorithms to be used in GAMS/MADARA.

- Formation coverage
- Prioritized Region Coverage
- Minimum Time Coverage
- Serpentine Coverage
- Waypoints
- Formation Follow
- Synchronized Formations
- Convoy Shielding
- Line Defense
- Arc Defense
- Onion Defense

- Executor

Waypoints

To use this kind of algorithm, you should configure some variables into knowledge system. To do that into agent 0 (zero), you can write into agent configuration file `agent_0.mf` the following lines:

```
agent.0.algorithm = "waypoints";
agent.0.algorithm.args.locations.size=4;
agent.0.algorithm.args.locations.0=[0, 0, 0];
agent.0.algorithm.args.locations.1=[0, 2, 0];
agent.0.algorithm.args.locations.2=[2, 2, 0];
agent.0.algorithm.args.locations.3=[2, 0, 0];
agent.0.algorithm.args.repeat=3;
```

So, you can set the number of waypoints with `.algorithm.args.locations.size`, number of repetitions by writing `.algorithm.args.repeat` and the amount of time (in seconds) that robot should wait after reaching one waypoint by writing `.algorithm.args.wait_time`.

Random Area Coverage

This algorithm generate random waypoints inside the region to coverage by your robot.

You can specify areas in env.mf to be coverage by agents. So, in `env.mf` you write:

```
region.0.object_type = 1;
region.0.type = 0;
region.0.priority = 0;
region.0.size = 4;
region.0.0 = [0, 0];
region.0.1 = [0, 5];
region.0.2 = [5, 5];
region.0.3 = [5, 5];
```

while, in `agent_0.mf` you specify the area to covered by the agent:

```
agent.0.algorithm="urac";
agent.0.algorithm.args.area="region.0";
```

The `object_type` can assume the following values:

- 1: region
- 2: prioritized region
- 4: search area (set of regions)

The `type` should be 0 (zero) to inform that the region polygon is convex. Nowadays, there is not other types of polygon implemented.

Snake Area Region Coverage

Limitations:

- only accepts gps frame type
- shift is constant (2.5 meters between parallel lines)

- Formation coverage
- Prioritized Region Coverage
- Minimum Time Coverage
- Serpentine Coverage
- Formation Follow
- Synchronized Formations
- Convoy Shielding
- Line Defense
- Arc Defense
- Onion Defense
- Executor

3.3.3 GAMS/MADARA Project manipulation

In this page, we will provide you a quick start with GAMS/MADARA project creation.

Creating a project

To create a new GAMS project, you should use a linux script called `gpc.pl`. So, you should run the following in a terminal window:

```
$GAMS_ROOT/scripts/projects/gpc.pl --path $PROJECT_HOME/tutorial1
```

With this command, you create a project named `tutorial1`, which has the following folder structure:

```
.
├── bin                    # store binary files
├── sim                    # store simulation files
|    ├── agent_0.mf            # stores vrep port number, starting location(lat,long,␣
↪alt) and algorithm name
|    ├── common.mf             # stores vrep configuration (ip adress, max distance of␣
↪VREPBase::move)
|    ├── env.mf                # stores vrep environment configuration (size, surface␣
↪texture/type, etc)
|    ├── run.pl                # stores running configurations (number of agents, hosts,
↪ domains, etc)
├── src                    # store your source code files
|    ├── algorithms            # store algorithms (non blocking piece of code)
|    ├── filters               #
|    ├── platforms             #
|    |    ├── threads          # store threads related to platforms
|    ├── threads               # store algorithms (blocking piece of code)
|    ├── transports            #
├── action.bat             # script to compile and run the project in Windows
├── action.sh              # script to compile and run the project in Linux
└── README.txt             # some how-to file with compile and run commands
```

Creating algorithms

You can create an algorithm (named as `talker`) into your project (called `tutorial3`) by running the following code:

```
$GAMS_ROOT/scripts/projects/gpc.pl --new-algorithm talker --path $PROJECT_HOME/
↪tutorial1
```

Creating threads (algorithms)

You can create a thread (named as `sense`) into your project (called `tutorial3`) by running the following command inside your project's folder:

```
$GAMS_ROOT/scripts/projects/gpc.pl --new-thread sense
```

Creating threads (platforms)

You can create a thread (named as `pid`) into your project (called `tutorial3`) by running the following command inside your project's folder:

```
$GAMS_ROOT/scripts/projects/gpc.pl --new-platform-thread pid
```

Adding more agents into simulation

You can add more agents into simulation by running the following command into terminal:

```
$GAMS_ROOT/scripts/projects/gpc.pl --agents 2 --randomize
```

With this command, your simulation will have 2 agents, which will generate randomized coordinates (lat, long). These values is stored into `agent_0.mf` and `agent_1.mf`,whose are located into `sim` folder.

Killing broken V-REP

Sometimes when you try to close your simulation, V-REP stops working. To fast kill it, just run the following command:

```
kill $(pgrep vrep)
```

Creating plataforms

To create plataforms in your projects, you should run the following command:

```
$GAMS_ROOT/scripts/projects/gpc.pl --new-platform my_platform --path $PROJECT_HOME/
↪tutorial1
```

DEBUG LOG LEVELS

You can define the log level that will be printed into gams terminal by configuring the file `sim/run.pl`:

```
$madara_debug = 3;
$gams_debug = 3;
```

You can define the following values to log level:

- LOG_EMERGENCY = 0,
- LOG_ALWAYS = 0,
- LOG_ERROR = 1,
- LOG_WARNING = 2,
- LOG_MAJOR = 3,
- LOG_MINOR = 4,
- LOG_TRACE = 5,
- LOG_DETAILED = 6,
- LOG_MAX = 6

3.3.4 GAMS/MADARA Programming Basics

In this page, we will provide you a quick start with GAMS/MADARA programming. You could follow the several tutorials that will introduce in a pratical way the main aspects of GAMS/MADARA.

Printing into agent terminal window

To print some message into Agent terminal window you can use the following command:

```
madara_logger_ptr_log (gams::loggers::global_logger.get (), gams::loggers::LOG_MAJOR,
↪" My message is hello folks!");
```

Creating an consumer/producer application

As your first application, we going to create an application that one agent generate values and other agent consume them. To do that, we start creating our application by running the following command:

```
$GAMS_ROOT/scripts/projects/gpc.pl --path $PROJECT_HOME/tutorial1
```

After the project creation, we should create two algorithms `producer` and `consumer`. This is done by running:

```
cd $PROJECT_HOME/tutorial1
$GAMS_ROOT/scripts/projects/gpc.pl --new-algorithm producer --path $PROJECT_HOME/
↪tutorial1
$GAMS_ROOT/scripts/projects/gpc.pl --new-algorithm consumer --path $PROJECT_HOME/
↪tutorial1
```

After running those commands, your filesystem looks like:

```
.
├── bin
├── sim
|     ├── agent_0.mf
|     ├── common.mf
|     ├── env.mf
|     ├── run.pl
├── src
|     ├── controller.cpp
|     ├── algorithms
|     |     ├── consumer.h
|     |     ├── consumer.cpp
|     |     ├── producer.h
|     |     ├── producer.cpp
|     ├── filters
|     ├── platforms
|     ├── threads
|     ├── transports
├── action.bat
├── action.sh
├── using_gams.mpb
├── using_vrep.mpb
├── using_ace.mpb
├── using_madara.mpb
├── workspace.mwc
└── README.txt
```

Now, we need add a second agent to our simulation, so we run the following command:

```
$GAMS_ROOT/scripts/projects/gpc.pl --agents 2 --randomize
```

With that, the folder `sim` will get updated by the addition of file `agent_1.mf`.

Finally, we have to configure the algorithm that each agent should run. Edit file `agent_0.mf` so the algorithm name be `producer`. Inside of file `agent_0.mf` will look like:

```
agent.0.algorithm = "producer";
```

Edit file `agent_1.mf` so the algorithm name be `consumer`:

```
agent.0.algorithm = "consumer";
```

Now, we have to declare a variable `counter` (of type `madara::knowledge::containers::Integer`) in `producer.h` and `consumer.h`.

So, your file `producer.h` will looks like:

```
class producer : public gams::algorithms::BaseAlgorithm
{

  protected:
         madara::knowledge::containers::Integer counter;

  ....
```

So, your file `consumer.h` will looks like:

```
class consumer : public gams::algorithms::BaseAlgorithm
{

  protected:
         madara::knowledge::containers::Integer counter;

  ....
```

In your `producer.cpp` we should configure the counter variable to be handled by madara::knowledge. So the file will looks like:

```
algorithms::producer::producer (
  madara::knowledge::KnowledgeBase * knowledge,
  gams::platforms::BasePlatform * platform,
  gams::variables::Sensors * sensors,
  gams::variables::Self * self,
  gams::variables::Agents * agents)
  : gams::algorithms::BaseAlgorithm (knowledge, platform, sensors, self, agents)
{
  status_.init_vars (*knowledge, "producer", self->agent.prefix);
  status_.init_variable_values ();
  counter.set_name("counter", *knowledge);
}

int algorithms::producer::plan (void)
{
        counter += 1;
        madara_logger_ptr_log (gams::loggers::global_logger.get (),␣
↪gams::loggers::LOG_MAJOR, "\n ----Incrementing the to counter: %d \n\n", counter.to_
↪integer());

  return 0;
}
```

In your `consumer.cpp`, we should relate the counter variable with madara::knowledge, so consumer will have access to updates made by producer.

```
algorithms::consumer::consumer (
  madara::knowledge::KnowledgeBase * knowledge,
  gams::platforms::BasePlatform * platform,
  gams::variables::Sensors * sensors,
  gams::variables::Self * self,
  gams::variables::Agents * agents)
  : gams::algorithms::BaseAlgorithm (knowledge, platform, sensors, self, agents)
{
  status_.init_vars (*knowledge, "consumer", self->agent.prefix);
  status_.init_variable_values ();
  counter.set_name("counter", *knowledge);
}

int algorithms::consumer::plan (void)
{
        madara_logger_ptr_log (gams::loggers::global_logger.get (),␣
↪gams::loggers::LOG_MAJOR, "\n ----Now the counter is: %d \n\n", counter.to_
↪integer());
  return 0;
}
```

After that, we should configure `controller.cpp` to share knowledge between the agents. You have to write the following commands:

```
// perform main logic of program
int main (int argc, char ** argv)
{
  settings.type = madara::transport::MULTICAST;
  settings.hosts.push_back("127.0.0.1:19906");
  settings.hosts.push_back("127.0.0.1:19907");

  ...
```

in above lines, you are telling to MADARA that knowledge should be shared into two hosts (agents): the host located into IP 127.0.0.1 and at port 19906; and the host located into 127.0.0.1:19907. You can get port number from agent file configuration located into folder `sim` (i.e. `sim/agent_0.mf` and `sim/agent_1.mf`).

We should compile and run the project by executing the following commands into bash:

```
$ ./action compile-vrep
$ ./action vrep run
```

You can get full source of this project by running the following command:

```
git clone https://github.com/marceloparavisi/gams_tutorials.git
```

3.3.5 GAMS/MADARA Compilation

In this page, we will provide you a quick start to compile GAMS/MADARA for different systems, such as:

GAMS/MADARA Compilation

In this page, we will provide you a quick start to compile GAMS/MADARA for Linux Desktop, more specifically all scripts was tested with Linux Ubuntu 16.04.

It is expected that all dependencies be resolved automatically by scripts. You have to be aware and look for errors of download failures, which the script can't handle.

To download GAMS/MADARA, just run:

```
$ export GAMS_ROOT=$HOME/gams
$ export CORES=4
$ git clone -b master --single-branch https://github.com/jredmondson/gams $GAMS_ROOT
```

After that, you should compile the following one of the following commands.

Build C++ with Tests

```
$ $GAMS_ROOT/scripts/linux/base_build.sh prereqs ace madara gams vrep tests
```

Build C++ with Tests and Java support

```
$ $GAMS_ROOT/scripts/linux/base_build.sh prereqs ace madara gams vrep tests java
```

Build C++ with Tests and Android support

```
$ $GAMS_ROOT/scripts/linux/base_build.sh prereqs ace madara gams vrep tests android
```

Build C++ with ROS support

```
$ $GAMS_ROOT/scripts/linux/base_build.sh prereqs ace madara gams ros
```

When the compilation finishes, you should see in folder you called the above script the following folders:

```
.
├── ace
├── gams
├── madara
└── vrep
```

The script below show several environment variables that must be configured to properly run and compile your gams/madara projects.

```
$ export PROJECT_HOME=$HOME/gamsProjects
$ export ACE_ROOT=$HOME/ace/ACE_wrappers
$ export MADARA_ROOT=$HOME/madara
$ export GAMS_ROOT=$HOME/gams
$ export VREP_ROOT=$HOME/vrep
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ACE_ROOT/lib:$MADARA_ROOT/lib:$GAMS_ROOT/
↪lib:$VREP_ROOT
$ export PATH=$PATH:$ACE_ROOT/bin:$MADARA_ROOT/bin:$GAMS_ROOT/bin:
$ export CORES=4
$ export NDK=$HOME/android_arm_tools
$ export LOCAL_CROSS_PREFIX=$NDK/bin/arm-linux-androideabi-
$ export SYSROOT=$NDK/sysroot
$ export PATH=$PATH:$VREP_ROOT
```

NOTE: Do not combine several arguments into `base_build.sh`, otherwise you will suffer with anomalous compilation effects.

JAVA SUPPORT

To allow the java compilation, you should install JAVA JDK (suggested JAVA 8) and configure the environment variable named `JAVA_HOME`. To do that, just run the following commands:

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-set-default
$ export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

If you get an error in `JavaPlatform`, you should edit `javaPlatForm.cpp` and `JavaPlatform.h` so the signature of JavaPlatform::move look likes this in `JavaPlatform.h`:

```
int move (const pose::Position & position, double epsilon = 0.1) override;
```

and looks this in `JavaPlatform.cpp`:

```
int gams::platforms::JavaPlatform::move (const pose::Position & position, double␣
↪epsilon)
```

ANDROID SUPPORT

If you are going to use Android in your aplications, you should download Android NDK. Extract NDK files into a folder and run the following shell script command inside of that folder.

```
$ ./build/tools/make-standalone-toolchain.sh --toolchain=arm-linux-androideabi-4.9 --
↪arch=arm --platform=android-14 --install-dir=../android_arm_tools
```

After the process is over, `../android_arm_tools` should be automatically created. After that, verify if the following environment variables are pointing to correct paths: `NDK`, `LOCAL_CROSS_PREFIX`, `SYSROOT`.

```
$ export NDK=\$HOME/bin/android_arm_tools
$ export LOCAL_CROSS_PREFIX=\$NDK/bin/arm-linux-androideabi-
$ export SYSROOT=\$NDK/sysroot
```

Minor fixes: UniformRandomEdgeCoverage

You should configure the variable `initialized_` to be true in the method `generate_new_position`:

```
void
gams::algorithms::area_coverage::UniformRandomEdgeCoverage::generate_new_position␣
↪(void)
{
  if (platform_ && *platform_->get_platform_status ()->movement_available)
  {
        .......
    initialized_ = true;
  }
}
```

minor fixes: AlgorithmFactoryRepository.cpp

You should change the following code from:

```
add (aliases, new area_coverage::PriorityWeightedRandomAreaCoverageFactory ());

add (aliases, new area_coverage::SnakeAreaCoverageFactory ());
```

to:

```
add (aliases, new area_coverage::PriorityWeightedRandomAreaCoverageFactory ());

  aliases.resize (2);
  aliases[0] = "snake";
  aliases[1] = "sac";

  add (aliases, new area_coverage::SnakeAreaCoverageFactory ());
```

GAMS/MADARA Compilation

In this page, we will provide you a quick start to compile GAMS/MADARA for different systems: Linux Desktop, Android, Linux RPi.

GAMS/MADARA Compilation

In this page, we will provide you a quick start to compile GAMS/MADARA for different systems: Linux Desktop, Android, Linux RPi.

3.3.6 GAMS/MADARA Simulation

In this page, we will provide you a quick start with GAMS/MADARA.

There are three parts of the ROS Quick Start tutorials. In the first part, you will get an understanding about ROS main concepts. Then, you will learn how to develop your first program with ROS, namely programming a publisher and subscriber using both C++ and Python. Finally, some more advanced concepts will be introduced through the development of a cleaning application with the Turtlesim simulator. At the end of ROS Quick Start tutorials, you will get a full understanding of ROS that allows you to dive deeper and develop more advanced robotics applications.

3.4 Raspberry Pi

Warning: @ To be done by Renan, Amory

3.4.1 Image Handling

This section presents how to burn a Linux image into a SDcard and also to backup the SDCard. It has foucus on RPi, but it should work for any Linux based embedded computer.

Download OS

Oficial Ubuntu MATE 16.04 for Raspberry Pi 3 https://ubuntu-mate.org/raspberry-pi/

Ubuntu MATE 16.04 with ROS Kinetic for Raspberry Pi 3 http://www.german-robot.com/2016/05/26/raspberry-pi-sd-card-image/

Oficial Ubuntu MATE 16.04 for Odroid XU4 https://wiki.odroid.com/odroid-xu4/os_images/linux/start https://odroid.in/ubuntu_16.04lts/

Unzip the Image file and go on to the next part.

Write the Image using Windows

Use Win32DiskImager for writing and reading Image files.

Write the Image using Linux

Execute `lsblk` or `df -l` to find out the mouting palce for the SDCard. It should be `/dev/sdX`, most probably `/dev/sdb` if your computer has only one disk.

There are several programs to burn the SDcard. dd is most well known but there are newer options such as `ddrescue` or `dcfldd`.

```
$ sudo ddrescue -D --force MyImage.img /dev/sdx
$ sudo dcfldd of=/dev/sdb if=~/MyImage.img
```

Force a synchronise of any outstanding input or output, then the card will be safe to remove.

```
$ sudo sync
```

That's it.

Expand the Image Size to Match the SDCard Size

For Raspberry Pi, execute

```
$ sudo raspi-config
```

and select 'Expand Filesystem'.

or, in the command line

```
$ sudo raspi-config --expand-rootfs
$ sudo reboot
```

For other embedded computers (e.g. ODroid), execute:

Note: describe here how to expand the Image on Odroid

Backing Up an Image File

Once your embedded computer is fully configured, it is a good ideia to:

- save a script with all the procedure to build the current image (packages installed, files configured, etc)
- backup the SDCard using minimal size, i.e. shrinking the Image file

Reading the Image file

Shutdown the embedded computer, take the SDCard to a Linux PC computer and proceed with the following steps.

Open a Terminal instance and enter the following Linux command where the SDCard is mounted;

```
$ df -h
```

Example:

```
$ df -h
Filesystem                    Size  Used Avail Use% Mounted on
udev                          7,8G     0  7,8G   0% /dev
tmpfs                         1,6G   50M  1,6G   4% /run
/dev/sda1                      50G   20G   28G  42% /
tmpfs                         7,8G  3,4M  7,8G   1% /dev/shm
tmpfs                         5,0M  4,0K  5,0M   1% /run/lock
tmpfs                         7,8G     0  7,8G   0% /sys/fs/cgroup
/dev/sda4                     813G  132G  640G  18% /home
tmpfs                         1,6G  124K  1,6G   1% /run/user/1000
```

(continues on next page)

(continued from previous page)

```
/dev/sdb2                       58G  6,7G   51G  12% /media/ale/PI_ROOT
/dev/sdb1                       63M   21M   43M  34% /media/ale/PI_BOOT
```

The last two are important: `/dev/sdb1` and `/dev/sdb2`.

Next we *unmount*, the Raspberry Pi SDcard:

```
$ sudo umount /dev/sdb1 /dev/sdb2
```

Now we make a backup copy of the Raspberry Pi image.

```
$ sudo dcfldd if=/dev/sdb of=~/MyImage.img
```

If `dcfldd` is not installed, then install it and reexecute the last command.

```
$ sudo apt-get update
$ sudo apt-get install dcfldd
```

Next we use the sync command to force a synchronise of any outstanding input or output

```
$ sudo sync
```

Let's take a look at the backed up image file. The file size should match the SDCard size.

```
$ ls -lsah ~/MyImage.img
```

That's all !

Shrinking the Image file

Let us assume the you used a 64GB SDcard to build your system. When you back it up, it will result in a 64GB image file, redardless the actual amount of space used in the SDcard. It will not be possible to directly use this image file in a, for example, 16GB SDCard. Before it, you need to shirink the image file.

This process is not exactly simple. It involves several steps as described here. Fortunalty, there are some scripts that perform these steps automatically. I personaly suggest this script. To run it you need:

- A Linux PC computer or a Linux VM for windows users
- Take the SDcard from the embedded computer and mount it on the Linux PC computer
- download the PiShrink script and follow the instructions

```
$ wget https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh
```

```
$ chmod +x ../pishrink.sh
$ sudo ../pishrink.sh image.img
[sudo] password for ale:
Creating new /etc/rc.local
e2fsck 1.42.13 (17-May-2015)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
PI_ROOT: 289654/3795104 files (0.2% non-contiguous), 2014409/15251456 blocks
```

(continues on next page)

(continued from previous page)

```
resize2fs 1.42.13 (17-May-2015)
resize2fs 1.42.13 (17-May-2015)
Resizing the filesystem on /dev/loop1 to 2226004 (4k) blocks.
Begin pass 2 (max = 369264)
Relocating blocks             XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Begin pass 3 (max = 466)
Scanning inode table          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Begin pass 4 (max = 22681)
Updating inode references     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
The filesystem on /dev/loop1 is now 2226004 (4k) blocks long.

Shrunk image.img from 59G to 8,6G
```

Generating checksum

Finally, it is recommended to generate a checksum file for the image file. This is usefull to check whether the file was correcpted during some data transfer.

```
$ md5sum image.img > image.md5
```

It results in a text file like this one. Save this file with the image file.

```
$ cat image.md5
75e87507e672de53241df4d724a0aac4  image.img
```

3.4.2 Setting Up the OS

- which basics packages to install
- how to setup the wireless
- main depedencies to intall
- setup automatic login
- how to enable the rpi pins and protocols (i2c, gpio, pwm, spi, etc)

3.4.3 Installing ROS

About ROS

- links to papers and manuals or other tutorials
-

How to install the driver and its depedencies

- where/how to download
- how to configure it
- how to install its depedencies
- how to install software depedencies and additional required nodes

- provide a script to install it all at once

Known limitations

describe here any known limitation of the software so that the next student is aware of it.

How to test it

- basic testing to see if the is procedure working on the RPi

3.4.4 Installing GAMS/MADARA to the Raspberry Pi

Warning: @ To be done by Marcelo e Davi

About GAMS/MADARA

- links to papers and manuals or other tutorials
-

How to install the driver and its depedencies

- make sure ros was installed first
- where/how to download gams/madara
- where/how to download gams/madara and ROS interface
- how to configures it
- how to install its depedencies
- how to install software depedencies and drivers the required
- provide a script to install it all at once

GAMS/MADARA working with ROS

- Marcelo e Davi describe the interface between them, limitations, how to compile it, how to test it.

Known limitations

describe here any known limitation of the software so that the next student is aware of it.

How to test it

- basic testing to see if the is procedure working on the RPi

3.4.5 Hooking Up Peripherals to the Raspberry Pi

This section shows how to add the following peripherals to the RPi board

Installing Xbee to the Raspberry Pi

Warning: @ To be done by Renan

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing the Adafruit GPS to the Raspberry Pi

Warning: @ To be done by Renan

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing the Emlid GPS-RTK to the Raspberry Pi

Warning: @ To be done by Renan

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing the IMU to the Raspberry Pi

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing the Raspicam to the Raspberry Pi

Warning: @ To be done by Renan

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing the Atlas Water Probes to the Raspberry Pi

Warning: @ To be done by ????

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing Lowrance sidescanner to the Raspberry Pi

Warning: @ To be done by ?????

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Installing the Water Sampling Device to the Raspberry Pi

Warning: @ To be done by ????

About the sensor

- where to buy, how much
- link to datasheet of the models available at LSA

How to physically connect it to the RPi

- describe power requirements
- bill of materials if required (ftdi, cables, etc)
- show fritzing schematics to connect the sensors, power, other boards, etc

How to install the driver and its depedencies

- how to install software depedencies and drivers the required

Known limitations of the sensor

describe here any known limitation of the sensor or its drivers so the next student is aware of it.

How to test it

- basic testing to see if the sensor is working on the RPi

Configuring the Ubiquity Bullet M2HP Radio

About

Bullet M2HP (Datasheet) is a high power WiFi radio adapter by Ubiquity.

Features:

- 100+ Mbps throughput
- Weatherproof design
- Compatible with high-gain antennas (N-type connector)

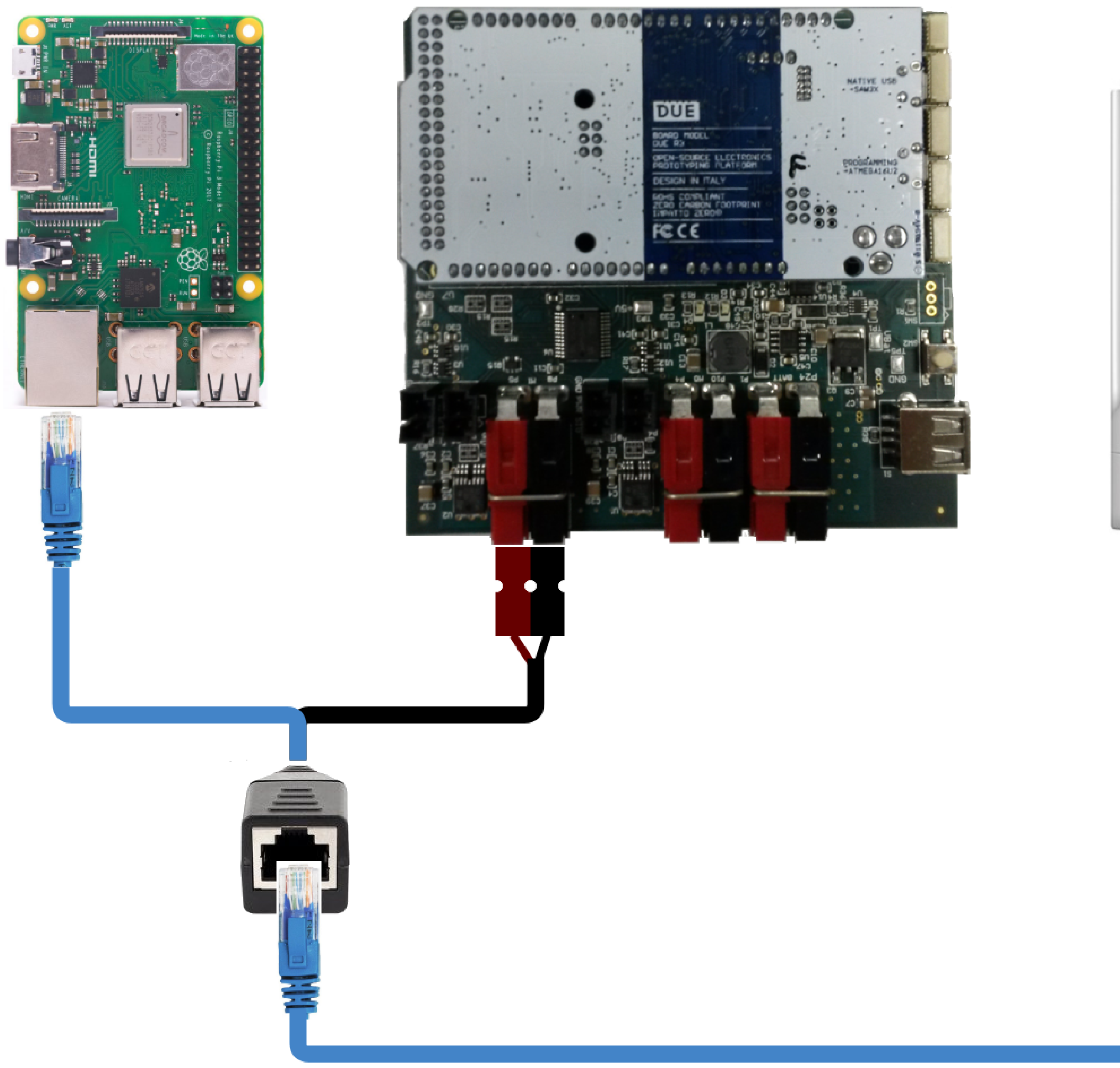
- Communication range up to 50 km (depending on the antenna)
- Power-over-Ethernet (PoE)

In the case of the Platypus boat, the Bullet is used as an access point (hotspot), generating a long-range WiFi (WLAN) network for communcation between the boat and the on-shore devices. It connects to the Raspberry Pi via its Ethernet port (LAN).

Powering up the Bullet M2HP

As the radio is powered through PoE, it comes with a PoE adapter as seen in the image below (exact model may differ):

After plugging the adapter to a power outlet, the PoE port connects to the Bullet, while the LAN port connects to the Raspberry Pi's Ethernet port. In the Platypus boat, the connection is the same, except for the power which comes from the boat's electronics board, as shown in the diagram below:



Accessing the Bullet M2HP configuration page

From a factory reset, the Bullet can be accessed through its standard IP (`192.168.1.20`). However, its default settings are WiFi station and bridge mode. As such, it does not generate a WiFi network nor runs a DHCP server, meaning that a physical cable connection and manual IP setting are necessary. The computer/Raspberry must also be operating in the same IP range as the Bullet, thus it must be a `192.168.1.x` IP.

For this initial setup, it is recommended to connect the Bullet to a conventional computer, as the configuration is done via browser (or to a Raspberry running a graphical interface OS). In Linux, we can manually connect to the bullet through the ifconfig command. After connecting the LAN adapter port into the computer, run:

```
$ sudo ifconfig
```

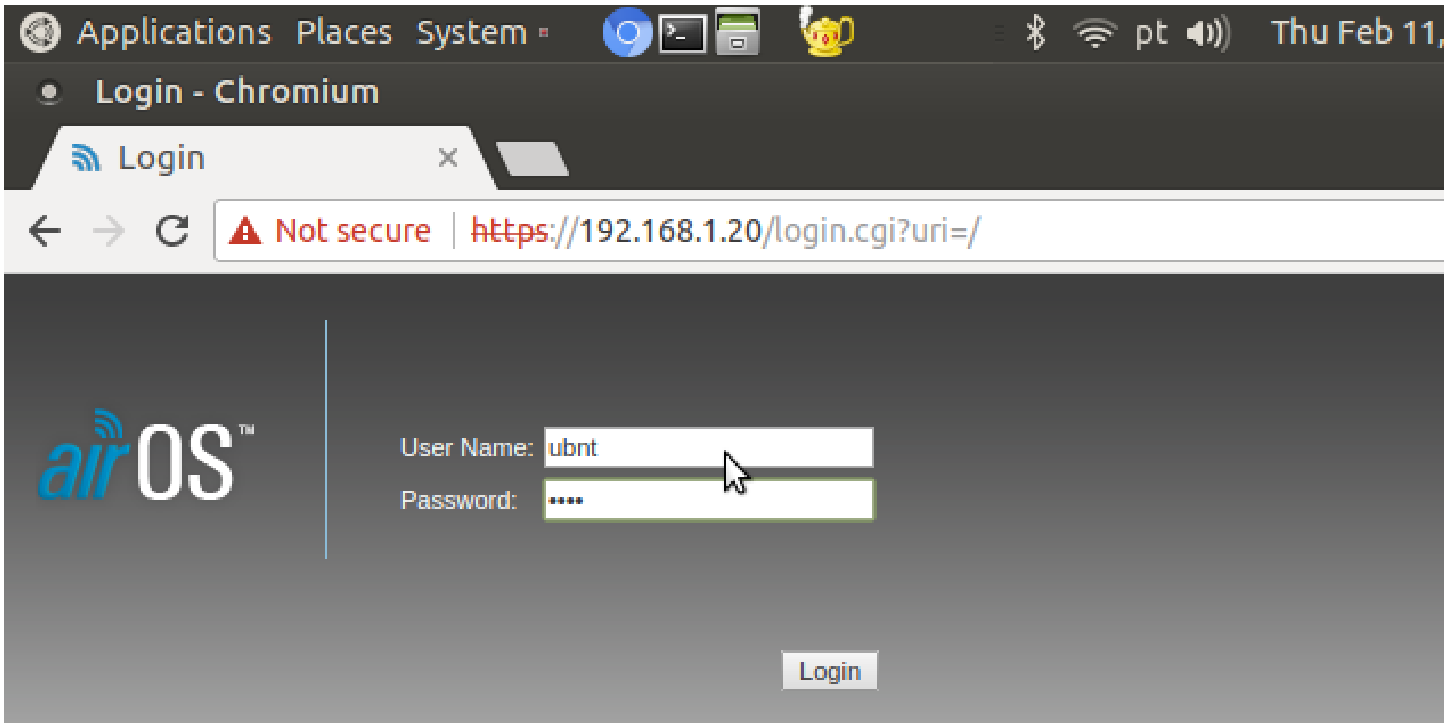
to find out which interface is responsible for Ethernet (e.g., eth0, enp0s3). To manually assign an IP and connect, run:

```
$ sudo ifconfig <name_of_interface> 192.168.1.x netmask 255.255.255.0
```

where x is a number from 1 to 254, excluding 20. For example:

```
$ sudo ifconfig eth0 192.168.1.7 netmask 255.255.255.0
```

The Bullet configuration page can the be accessed in a browser by the address http://192.168.1.20. A untrusted connection or security certificate warning may appear, which can be safely ignored. A screen like the one below will appear:



User Name: ubnt

To connect and configure the M2 Bullet, use the credentials:

```
login: ubnt
password: ubnt
```

Configuring the Bullet M2HP as an Access Point

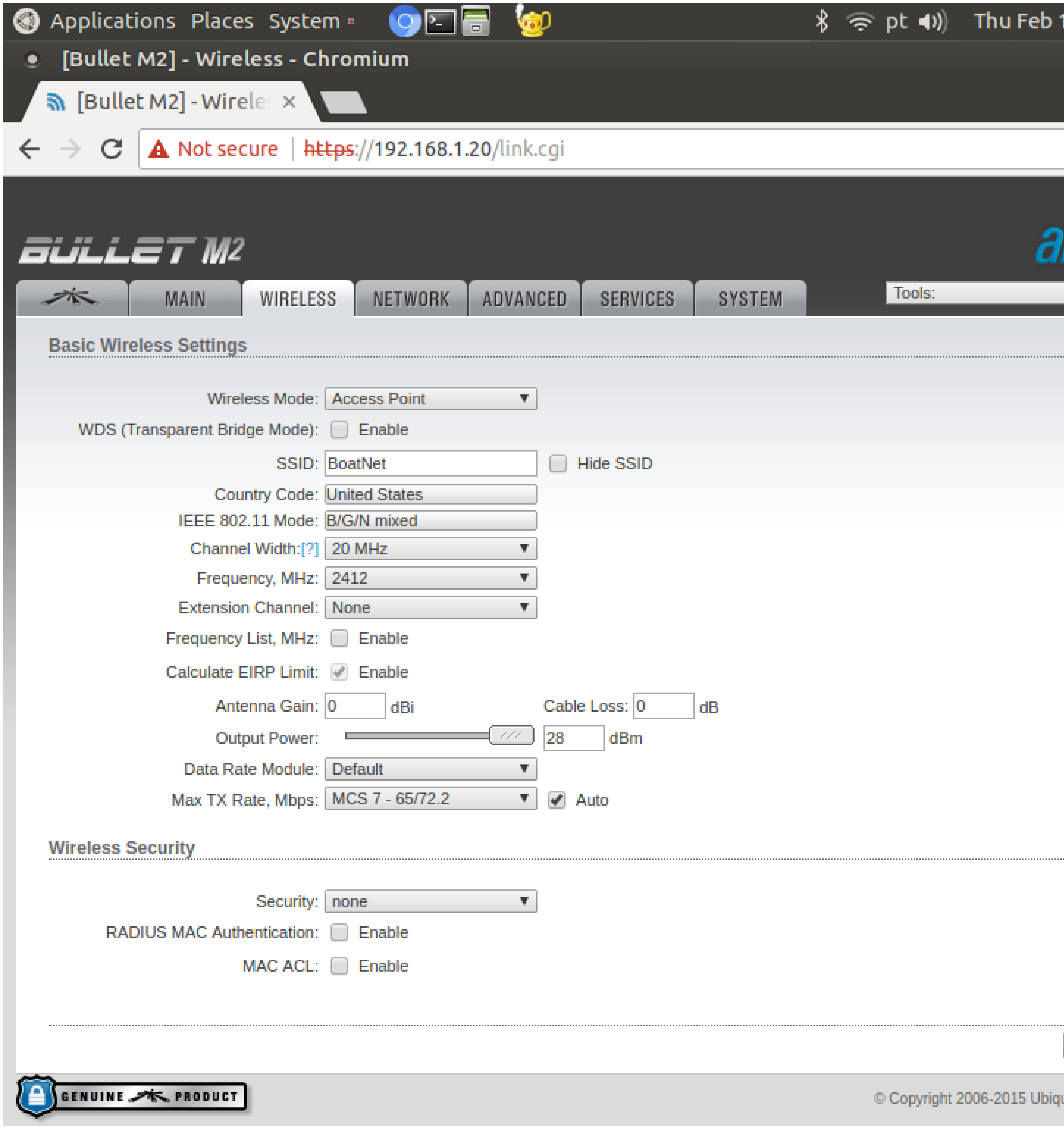
To configure the Bullet as an access point, change the following configurations in the Wireless tab:

```
Wireless mode: Access Point
SSID: <WiFi network name>
Channel Width: 20 MHz
Frequency, MHz: 2412
```

(continues on next page)

(continued from previous page)

```
Output power: Max
Security: None (you may setup WPA-PSK or WPA2-PSK if prefered)
```


Applications Places System
pt Thu Feb
[Bullet M2] - Wireless - Chromium
[Bullet M2] - Wirele
Not secure
https://192.168.1.20/link.cgi
BULLET M2
MAIN
WIRELESS
NETWORK
ADVANCED
SERVICES
SYSTEM
Tools:
GENUINE PRODUCT
© Copyright 2006-2015 Ubiq

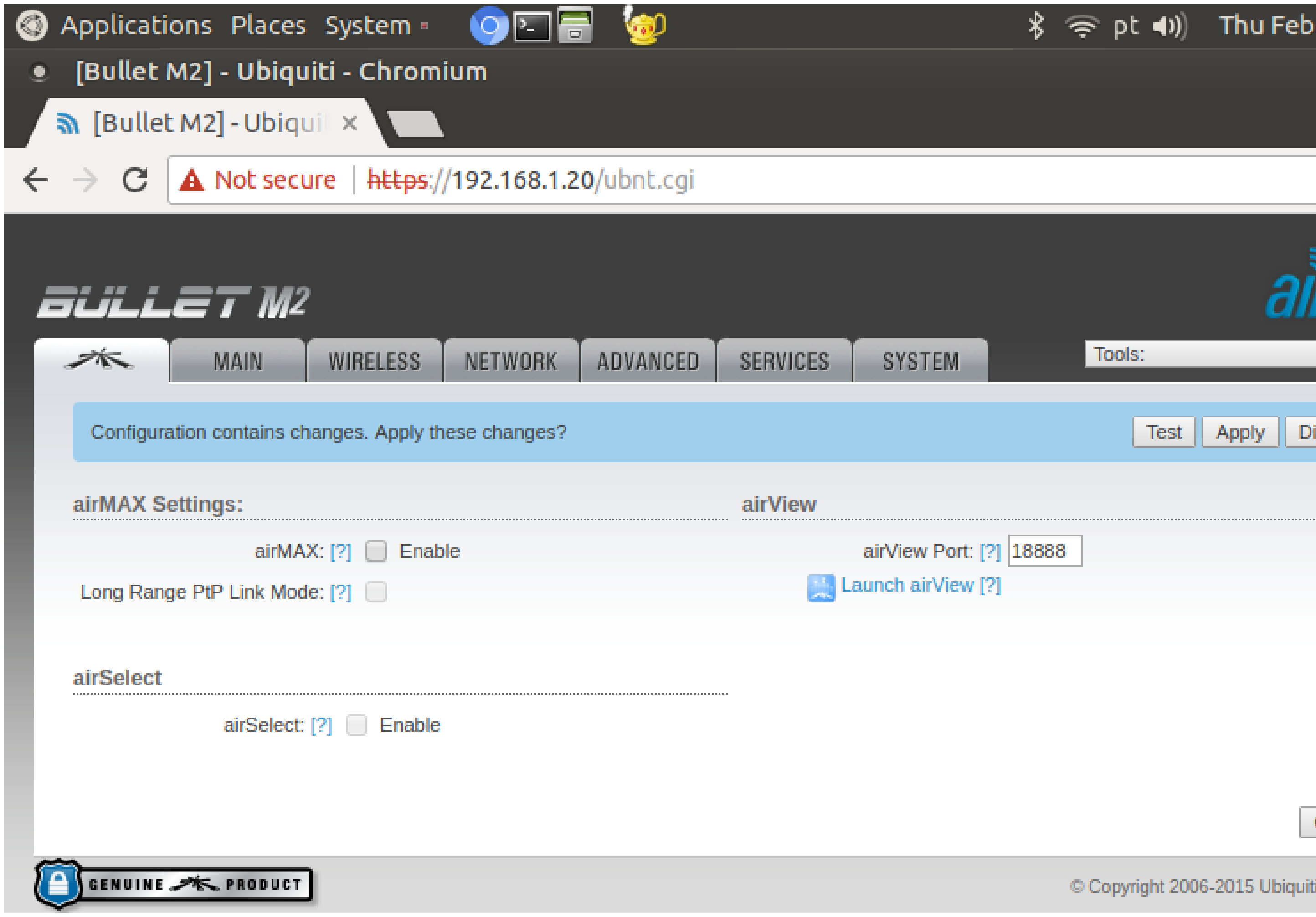
Basic Wireless Settings

Wireless Mode:	Access Point
WDS (Transparent Bridge Mode):	Enable
SSID:	BoatNet Hide SSID
Country Code:	United States
IEEE 802.11 Mode:	B/G/N mixed
Channel Width:[?]	20 MHz
Frequency, MHz:	2412
Extension Channel:	None
Frequency List, MHz:	Enable
Calculate EIRP Limit:	✓ Enable
Antenna Gain:	0 dBi Cable Loss: 0 dB
Output Power:	28 dBm
Data Rate Module:	Default
Max TX Rate, Mbps:	MCS 7 - 65/72.2 ✓ Auto

Wireless Security

Security:	none
RADIUS MAC Authentication:	Enable
MAC ACL:	Enable

Click "Change" but **do not click "Apply" yet**. In the leftmost tab, uncheck the "AirMax" box and click "Change".


Applications Places System
[Bullet M2] - Ubiquiti - Chromium
Not secure https://192.168.1.20/ubnt.cgi
BULLET M2
MAIN WIRELESS NETWORK ADVANCED SERVICES SYSTEM
Tools:
Test Apply
GENUINE PRODUCT
© Copyright 2006-2015 Ubiquiti

Configuration contains changes. Apply these changes?

airMAX Settings:

airMAX: [?] Enable

Long Range PtP Link Mode: [?]

airView

airView Port: [?] 18888

Launch airView [?]

airSelect

airSelect: [?] Enable

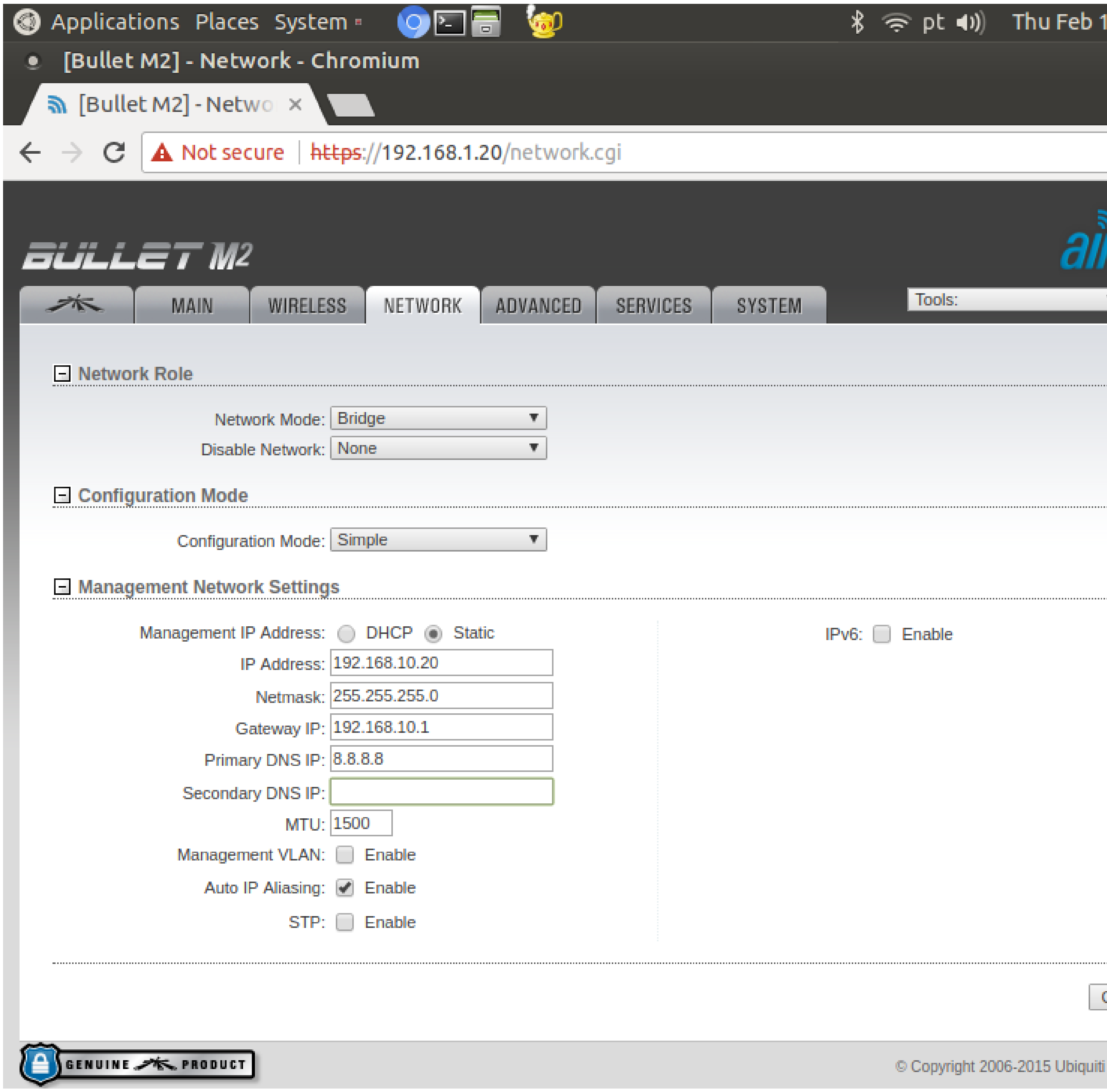
This is sufficient to configure the radio as an Access Point, and it will generate a WiFi network with the specified SSID. However, there are two possible configurations regarding the Bullet's network role: *Bridge* and *Router* mode.

Note: The typical configuration is to use the Bullet in *Router* mode with a DHCP server, as to automatically address an IP to each connecting device. If this is the case, this configuration can be performed in the "Network" tab and the changes can be applied with the "Apply" button. After a short connection drop, you may connect to the access point's WiFi network and access it with the IP specified in the "Network" tab. As the Router mode isolates the LAN and WLAN interfaces, connection between wireless devices and the Raspberry's Ethernet becomes impossible. Thus, in the case of the Platypus boat, **the radio must be configured in Bridge mode**.

Configuring the Bullet M2HP in Bridge mode

In the `Network` tab, modify the following fields to configure the radio in bridge mode:

```
Network mode: Bridge
Management IP Address: Static
IP Address: <IPV4 IP> (e.g., 192.168.10.20)
Netmask: 255.255.255.0
Gateway IP: <First IP in the same range as IP Address> (e.g., 192.168.10.1)
Primary DNS: 8.8.8.8
```


Applications Places System
[Bullet M2] - Network - Chromium
Not secure https://192.168.1.20/network.cgi
BULLET M2
MAIN WIRELESS NETWORK ADVANCED SERVICES SYSTEM
Tools:
GENUINE PRODUCT
© Copyright 2006-2015 Ubiquiti

Network Role

Network Mode: Bridge

Disable Network: None

Configuration Mode

Configuration Mode: Simple

Management Network Settings

Management IP Address: DHCP [✓ Static]

IPv6: Enable

IP Address: 192.168.10.20

Netmask: 255.255.255.0

Gateway IP: 192.168.10.1

Primary DNS IP: 8.8.8.8

Secondary DNS IP:

MTU: 1500

Management VLAN: Enable

Auto IP Aliasing: [✓ Enable]

STP: Enable

Click `Change` and then `Apply`. After a short connection drop, the radio will generate a WiFi network with the SSID specified in the `Wireless` tab.

Testing the connection between WiFi and the Raspberry's Ethernet

As there is no DHCP server, the Raspberry and WiFi devices must be manually connected to the Bullet. Once again, this can be done with the `ifconfig` command in Linux. For the Raspberry, the interface used is the Ethernet (e.g., `eth0`). After connecting the LAN cable on the Raspberry's Ethernet port, run:

```
$ sudo ifconfig <name of interface> <IP in the same range as the Bullet> netmask 255.
↪255.255.0
```

For example:

```
$ sudo ifconfig eth0 192.168.10.210 netmask 255.255.255.0
```

Setting IP by `ifconfig` will assign the IP temporarily to the port. In order to fix a static IP to the Raspberry Pi, the `/etc/dhcpcd.conf` must be edited, adding the following lines to the end of the file:

```
interface eth0
static ip_address=192.168.10.210/24
static routers=192.168.10.1
static domain_name_servers=192.168.10.1
```

Configuring Android Devices

An IP must also be manually set for the WiFi devices. In Android 7.0, this can be done as follows:

- Settings -> Connections -> Wi-Fi
- Touch the Bullet's WiFi network
- Check "Show advanced options"
- IP settings -> Static
- IP address -> IP in the same range as the Bullet (e.g., 192.168.10.207)
- Gateway -> Same as specified in the "Network" tab (e.g., 192.168.10.1)
- DNS 1 -> 8.8.8.8

To test the connection, run the "ping" command on the Raspberry:

```
$ ping <IP of connected WiFi device>
```

or on the WiFi device (if available):

```
$ ping <IP of Raspberry>
```

If successful, there should be a response such as:

```
PING 192.168.10.207 (192.168.10.207) 56(84) bytes of data.
64 bytes from 192.168.10.207: icmp_seq=1 ttl=64 time=7.03 ms
64 bytes from 192.168.10.207: icmp_seq=2 ttl=64 time=17.1 ms
...
```

3.4.6 Video Streaming Tutorials

Video Streaming with RaspberryPi Using VLC

Tip:

In this tutorial, you will:

- Learn how to configure your Raspberry Pi for video streaming
- Know the commands needed for simple video streaming through the VLC media tool

Tip:

This demonstration was tested on:

- VLC 2.2.4 on a Windows 8.1 64-bit Computer
- 2017/1/11 Raspbian Jessi on a RBpi 2 Model B V1.1 using Pi Camera rev 1.3
- Note: Pi Camera V2.1 was also tested successfully

This tutorial will introduce to you to your Raspberry Pi Camera Module to view a video stream from your Pi setup, the server using Raspbian, to a different computer, a client using Windows, in your home network

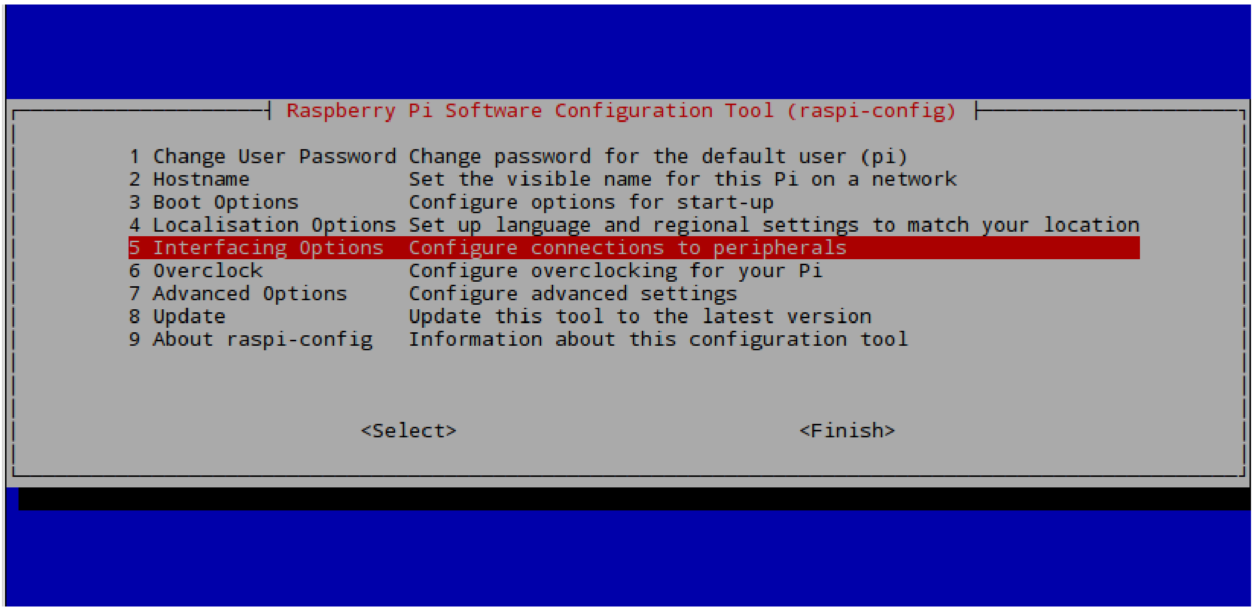
Configuring your RaspberryPi

Firstly, on your Pi's terminal, Update and Upgrade the environment so it can be up to date. This helps in reducing future problems. Don't forget to ENABLE your Raspberry Pi Camera using 'raspi-config'.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

```
$ sudo raspi-config
```

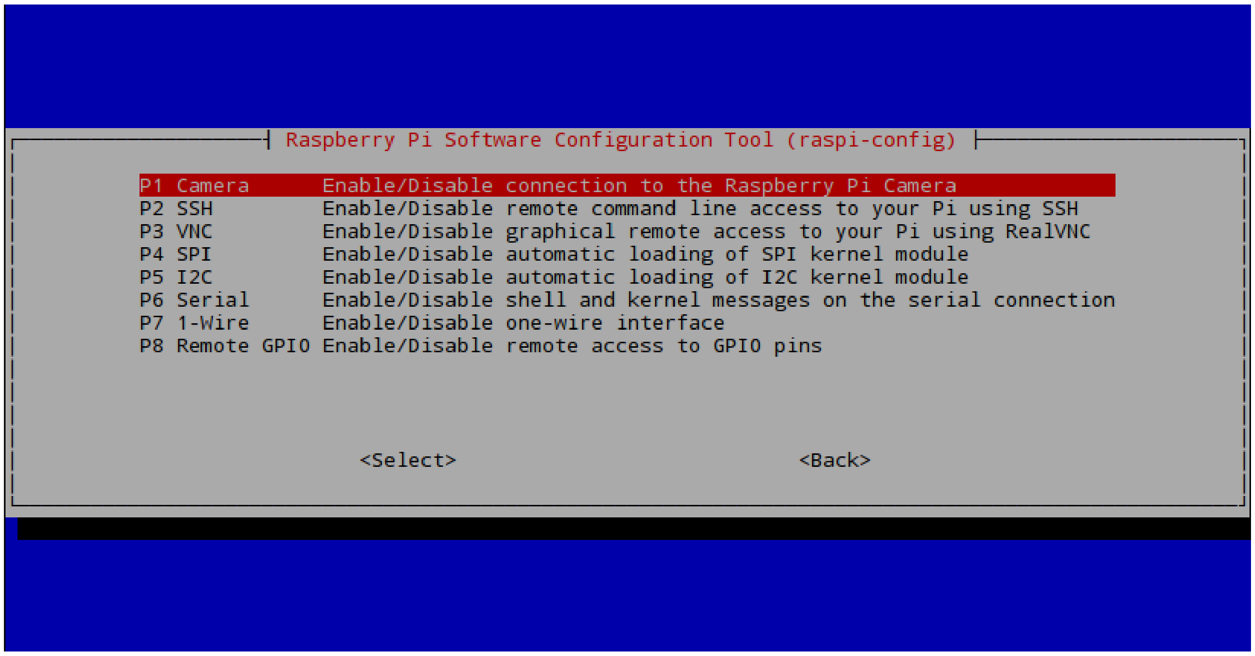
A blue BIOS-like screen will appear, go into the Enable Camera Options and enable the camera.



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

                  <Select>                            <Finish>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

                  <Select>                            <Back>
```

Note: Depending on your version of Raspbian, the Enable setting may not first appear on the main list. You will have to go under the settings in the blue screen to locate the enable option.

It is also advised now to see what is the IP Address of your Pi. Type in the following to locate the IP as you will need it in the VLC program for your Windows machine.

```
$ ifconfig
```

If you are using a wireless connection,
the IP you want is located in the `lo` section under `inet addr:x.x.x.x`
If you are using ethernet,

it will be under `eth0` in `inet addr:x.x.x.x`

Getting VLC

On your Client PC that is running Windows, download the VLC software media tool on here through the VLC's Website

Now on your Pi's terminal, download and install the VLC for Raspbian.

```
$ sudo apt-get install vlc
```

Note: Make sure that your Pi is up-to-date and also now has VLC and that your PC has VLC installed, before going to the next step

Initiating the Stream

Once installed, you may now start the video streaming by typing the folloing in your Pi's Terminal.

```
$ raspivid -o - -t 0 -hf -w 800 -h 400 -fps 24 |cvlc -vvv stream:///dev/stdin --sout '
↪#standard{access=http,mux=ts,dst=:8160}' :demux=h264
```

- `-o` Specifies the output filename. the '-' beside denotes no filename
- `-t` is the duration of the recoding, 0 being infinity
- `-hf` is Horizontal Flip
- `-w` and -h is the resolution for Width and Height
- `-fps` is Frames per Second
- The rest means that on `port 8160`, data will be sent through http using h264 as stdout using VLC

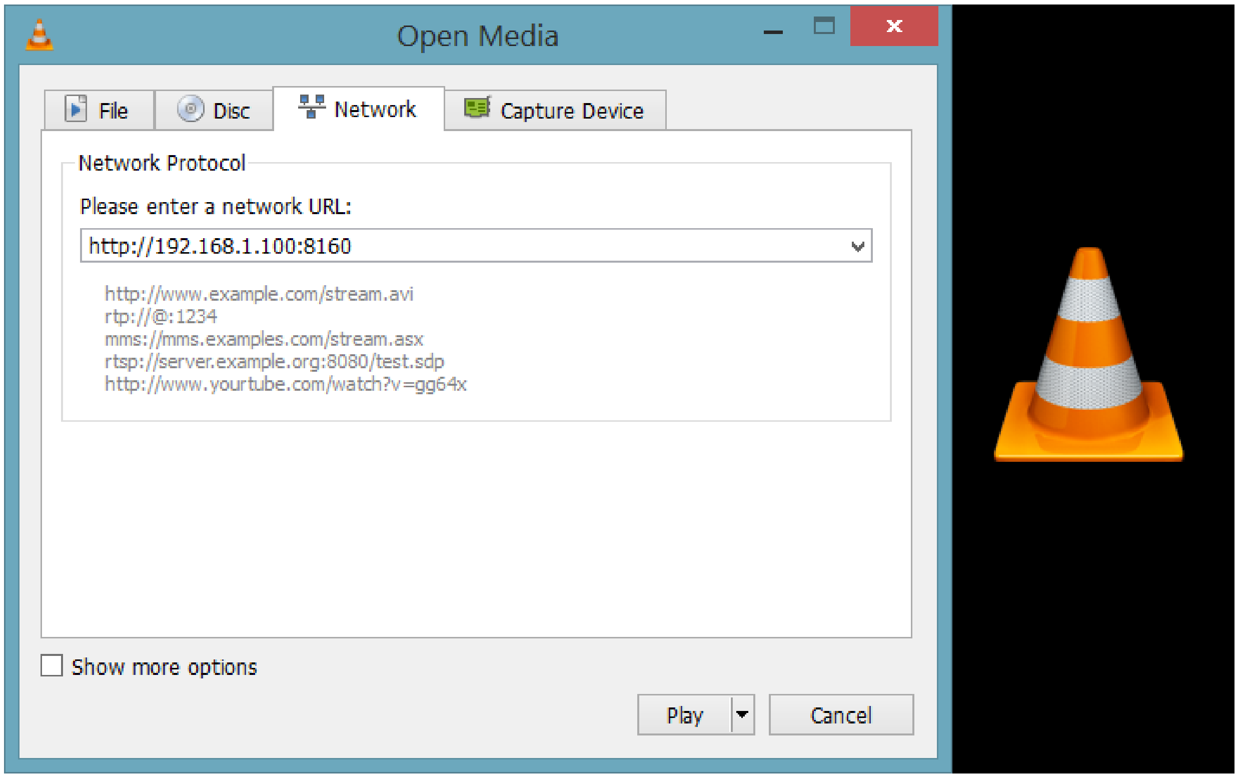
Once entered, the Pi Camera will turn on and start recording and simultaneously send it over http. This is now the time to go to your Windows machine and watch your streaming footage.

Note: You may want to experiment and change settings like `-w`, `-h`, and `-fps`.

Open the VLC program on your Windows Machine.

And under `Media` > `Open Network Stream` > `Network` > `Please enter a network URL:`

Type in the IP address that you got from `ifconfig` like so;



Please enter a network URL:

http://192.168.1.100:8160

http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x

```
http://x.x.x.x:8160
```

As we have specified the port to be 8160 in our terminal on the Pi

Once entered, VLC will automatically start playing the stream from the Pi over your network.

Conclusion

Note: As you can see from the stream that the video quality is not that ground breaking but is acceptable, and the latency is the biggest issue of this streaming method.

Video Demonstration

Note: The Monitor on the left displays real time from the Raspberry directly, whereas the Laptop is displaying the VLC stream.

Raspberry Pi camera module streaming video to another computer. This video tutorial shows the overview of this written tutorial.

END

Video Streaming with RapsberryPI Using gStreamer

Tip:

In this tutorial, you will:

- Learn how to configure your Raspberry Pi for video streaming through the gStreamer Method
- Know the commands needed for simple video streaming through gStreamer

Note: This demonstration uses a Linux based environment (Ubuntu) as the client side, NOT a Windows PC like the other methods.

Tip:

This demonstration was tested on:

- Google Chrome Version 56.0.2924.87 on Ubuntu 14.04 64-bit
- 2017/1/11 Raspbian Jessi on a RBpi 2 Model B V1.1 using Pi Camera rev 1.3
- Note: Pi Camera V2.1 was also tested successfully

This tutorial will introduce to you to your Raspberry Pi Camera Module to view a video stream from your Pi setup, the server using Raspbian, to a different computer, a client using Ubuntu, in your home network

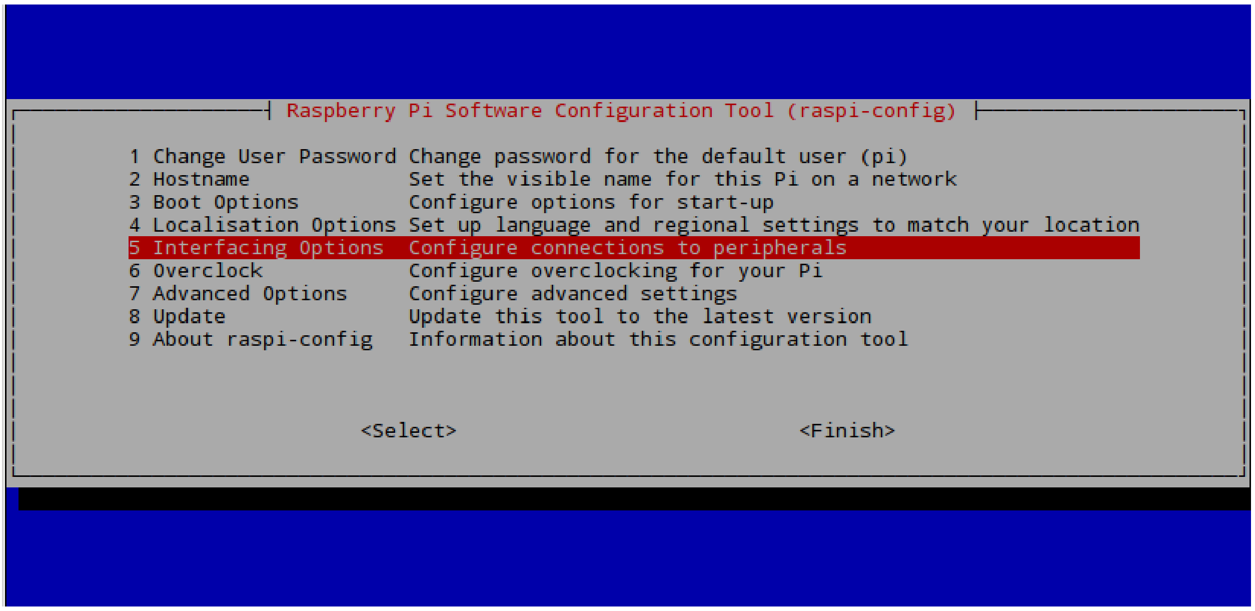
Configuring your RaspberryPi

Firstly, on your Pi's terminal, Update and Upgrade the environment so it can be up to date. This helps in reducing future problems. Don't forget to ENABLE your Raspberry Pi Camera using 'raspi-config'.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

```
$ sudo raspi-config
```

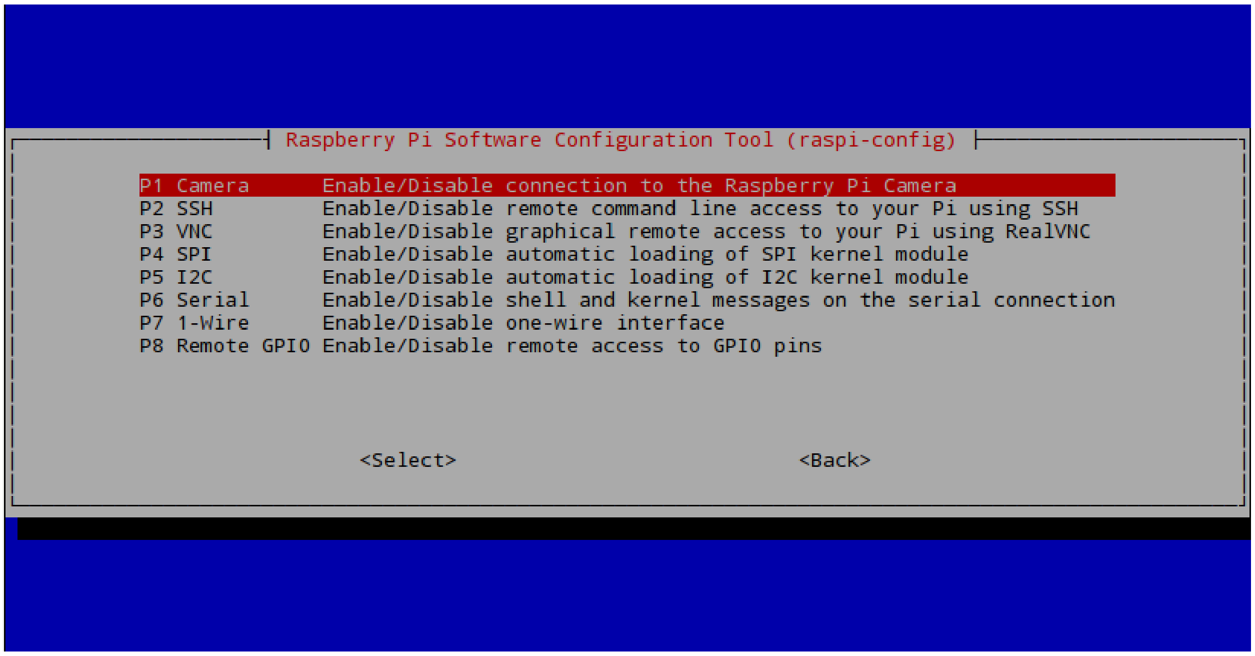
A blue BIOS-like screen will appear, go into the Enable Camera Options and enable the camera.



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

                  <Select>                          <Finish>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

                  <Select>                          <Back>
```

Note: Depending on your version of Raspbian, the Enable setting may not first appear on the main list. You will have to go under the settings in the blue screen to locate the enable option.

It is also advised now to see what is the IP Address of your Pi. Type in the following to locate the IP as you will need it in the Browser for your Windows machine.

```
$ ifconfig
```

If you are using a wireless connection,
the IP you want is located in the `lo` section under `inet addr:x.x.x.x`
If you are using ethernet,
it will be under `eth0` in `inet addr:x.x.x.x`

Getting gStreamer

Now we will get into the main focus of this tutorial, gStreamer. gStreamer is a multimedia tool that connects a sequence of elements through a pipeline.

We will now get gStreamer for both the Pi and your Ubuntu

```
$ sudo add-apt-repository ppa:gstreamer-developers/ppa
$ sudo apt-get update
$ sudo apt-get install gstreamer1.0*
```

Initiating the Video Stream

After the installation, to begin the video stream, we can type in the Pi:

```
$ raspivid -fps 26 -h 450 -w 600 -vf -n -t 0 -b 200000 -o - | gst-launch-1.0 -v fdsrc␣
↪! h264parse ! rtph264pay config-interval=1 pt=96! gdppay ! tcpserversink host=x.x.x.
↪x port=5000
```

..NOTE::

- You can remove `-n` so you can start a preview on your Pi, `-n` disables the preview
- `-b` is for the bitrate

Please note that the host here must be changed to YOUR host IP from the `ifconfig` above. That will initiate the stream from the Pi side.

On your client with Linux, also install gStreamer, and then type in the terminal

```
$ gst-launch-0.10 -v tcpclientsrc host=x.x.x.x port=5000 ! gdpdepay ! rtph264depay !␣
↪ffdec_h264 ! ffmpegcolorspace ! autovideosink sync=false
```

Please note that the `host` here must be changed to YOUR host IP from the `ifconfig` above. Now you will see the stream from the Pi server.

Note: As you can see, the quality and latency is ground breaking this time compared to the VLC and mjpgStreamer methods.

Video Demonstration

Note: The Monitor on the left displays real time from the Raspberry directly, whereas the Laptop is displaying the gStreamer stream.

Wirelessly streaming a video from a Raspberry to a remote laptop. This video tutorial shows the overview of this written tutorial.

END 3

Video Streaming with RapsberryPI Using mjpgStreamer

Tip:

In this tutorial, you will:

- Learn how to configure your Raspberry Pi for video streaming through the mjpgStreamer Method
- Know the commands needed for simple video streaming through mjpgStreamer
- Acquire the dependencies needed for mjpgStreamer

Tip:

This demonstration was tested on:

- Google Chrome Version 56.0.2924.87 on a Windows 8.1 64-bit Computer
- 2017/1/11 Raspbian Jessi on a RBpi 2 Model B V1.1 using Pi Camera rev 1.3
- Note: Pi Camera V2.1 was also tested successfully

This tutorial will introduce to you to your Raspberry Pi Camera Module to view a video stream from your Pi setup, the server using Raspbian, to a different computer, a client using Windows, in your home network

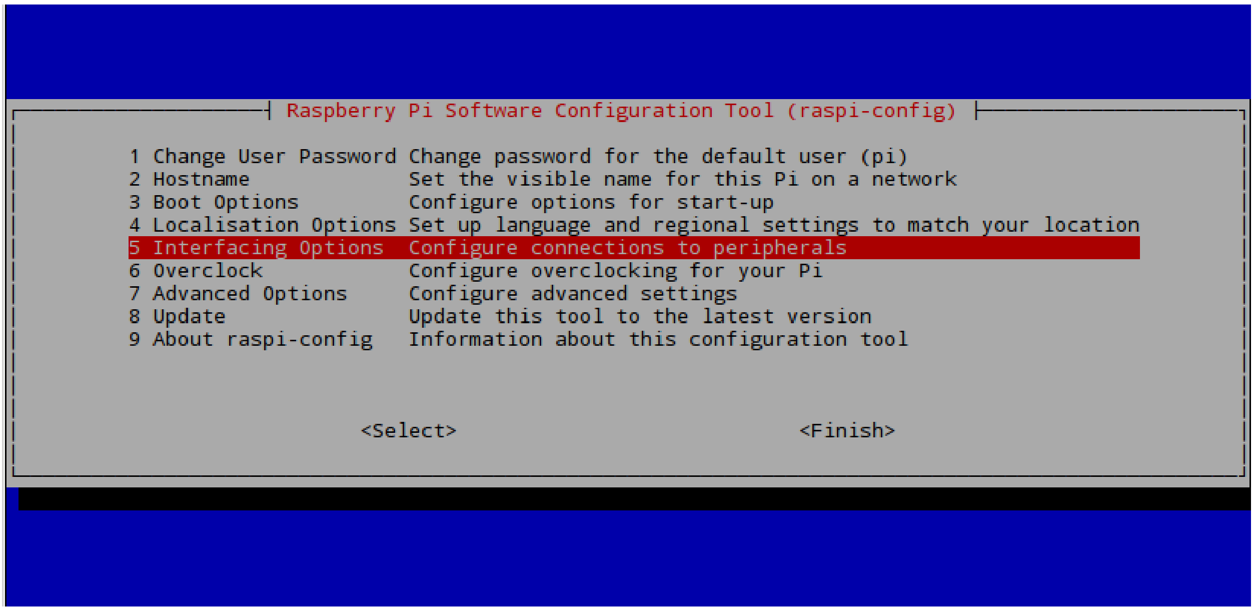
Configuring your RaspberryPi

Firstly, on your Pi's terminal, Update and Upgrade the environment so it can be up to date. This helps in reducing future problems. Don't forget to ENABLE your Raspberry Pi Camera using 'raspi-config'.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

```
$ sudo raspi-config
```

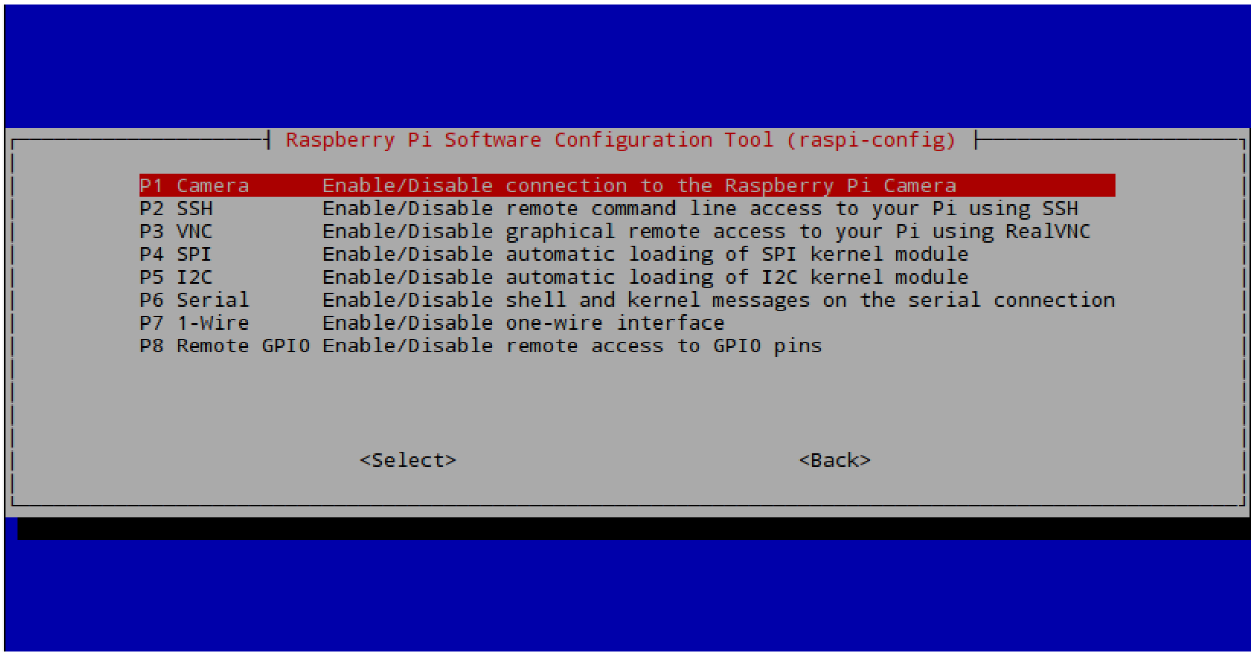
A blue BIOS-like screen will appear, go into the Enable Camera Options and enable the camera.



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

<Select>                                <Finish>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

<Select>                                <Back>
```

Note: Depending on your version of Raspbian, the Enable setting may not first appear on the main list. You will have to go under the settings in the blue screen to locate the enable option.

It is also advised now to see what is the IP Address of your Pi. Type in the following to locate the IP as you will need it in the Browser for your Windows machine.

```
$ ifconfig
```

If you are using a wireless connection,
the IP you want is located in the `lo` section under `inet addr:x.x.x.x`
If you are using ethernet,

it will be under `eth0` in `inet addr:x.x.x.x`

Getting mjpgStreamer

We will now install mjpgStreamer on our Pi, the main focus of this method To do this, we will go to the mjpgStreamer website which will automatically start the download.

We will need to decompress the file, this process will also install it at the same time.

```
$ tar -zxvf mjpg-streamer.tar.gz
```

Press Enter, and you should see a new directory called `mjpg-streamer`

Note: You can check for directories in the terminal by typing in `ls`

Getting mjpgStreamer's Dependencies

Now we need mjpgStreamer's dependancies to make it fully functional.

```
$ sudo apt-get install libjpeg8-dev
$ sudo apt-get install imagemagick
```

After this is done, go into the mjpg-streamer directory inside the already existing mjpg-streamer. **Yes, type it twice**. And then type `make` which will build the system and compile it

```
$ cd mjpg-streamer
$ cd mjpg-streamer
$ make
```

In order to start the capture, we must make a temporary file that will save the image taken by raspistill, and then it will get updated many times every second. So in `~/mjpg-streamer/mjpg-streamer $` type in:

```
$ mkdir /tmp/stream
```

We can now initiate the stream by typing in

```
$ LD_LIBRARY_PATH=./ ./mjpg_streamer -i "input_file.so -f /tmp/stream -n pic.jpg" -o
↪"output_http.so -w ./www"
```

Open a new terminal window and type

```
$ raspistill -w 640 -h 480 -q 5 -o /tmp/stream/pic.jpg -tl 1 -t 9999999 -th 0:0:0
```

- `-w` and -h is resolution
- `-q` is quality
- `-o` is the Specified output filename
- `-tl` is the time interval between each snap shot (here is 1 millisecond)
- `-t` is the camera's ON time in seconds, 9999999 is 115 Days
- `-th` Set thumbnail parameters (x:y:quality)

Now, on your client computer, open your preferred browser and type in your `IP` and `port#` which will be 8080 by default.

```
x.x.x.x:8080
```

A website will display showing you the mjpgStreamer Demo Page and a congratulation message. Go to the stream section in the menu to see the live footage from your Pi.

Note: As you can see from the stream that the video quality is not that ground breaking but is acceptable, although a little worse than the VLC method, however the latency is a so much better than in the VLC method.

Video Demonstration

Note: The Monitor on the left displays real time from the Raspberry directly, whereas the Laptop is displaying the mjpgSteamer stream.

Raspberry Pi Camera Stream Web Video. This video tutorial shows the overview of this written tutorial.

END 2

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. This tutorial will introduce to you the Raspberry Pi Camera Module to view a video stream from your Pi setup and show you how to start video streaming through several tools.

3.4.7 Thorough Tests for the Board

- describe here how one can test the features of the board

Possible Faults

- describe here usual fault and how to solve it
- describe where to buy replacement parts

3.5 ODROID XU4

Warning: @ To be done by Roger and Amory

3.5.1 Download Image

Oficial Ubuntu MATE 16.04 for Odroid XU4:

- https://wiki.odroid.com/odroid-xu4/os_images/linux/start
- https://odroid.in/ubuntu_16.04lts/

Unzip the Image file and go on to the next part.

3.5.2 Write the Image to the SDCard

The procedure is the same compared to procedure for Raspberry Pi

- *Write the Image using Linux*
- *Write the Image using Windows*

3.5.3 Write the Image to the eMMC memory

One of the nice features of Odroid XU4 is that it has the eMMC memory module. According to them, the eMMC 5.0 storage is ~7x faster than the MicroSD Class-10 card in read tests. When using it, one can realise that the boot is clearly faster than SD cards.

There are two ways to load eMMC memory, depending whether the eMMC Module Reader is available or not.

Using eMMC Module Reader

This method requires the eMMC Module Reader

Then, the procedure to write the Image is the same for SDCards.

Using Odroid and SDCard

If the Reader is not available, one can follow these steps:

- Load the Image on a SDCard as described above
- Set the memory Switch to select SDCard
- Plug the eMMC module
- Turn the system on and boot the system
- Plug in the USB3 interface a external drive with the Image file
- Find out the mounting point for the eMMC module, which is most probably `/dev/mmcblk1`

- Go to the directory with the Image file and
- Execute: `sudo dcfldd of=/dev/mmcblk1 if=./MyImage.img`

Done, the eMMC module was loaded without using the Reader. Now, to test boot with the eMMC, follow these steps:

- Shutdown the computer
- Remove the SDCard and make sure the eMMC is connected
- Set the memory Switch to select eMMC
- Turn the system on and boot the system

You will see that the time for the boot is shorter and the computer is faster. It will boot twice when you boot it for the first time. The reason is that it automatically expand the file system in the 1st boot. However, it is transparant to the user.

3.5.4 Backing up Image from the eMMC memory

A similar procedure as described in the previous section can be used for backing up the Image in the eMMC module. Follow these steps:

- Load the Image on a SDCard as described above
- Set the memory Switch to select SDCard
- Turn the system on and boot the system
- Plug in the USB3 interface an external drive where the Image file will be saved
- Find out the mounting point for the eMMC module, which is most probably `/dev/mmcblk1`
- Go to the directory where the Image file will be saved and
- Execute: `sudo dcfldd if=/dev/mmcblk1 of=./MyImage.img`

Done! Now it is recommended to *shrink the Image file*. For more information about loading Image, refer to *Raspberry Pi 3 procedure*.

3.5.5 Setting Up ROS

There is no special procedure to install ROS dor Odroid. Just follow the same procedure for *Raspberry Pi 3*, using Ubuntu Mate 16.04.

3.5.6 Setting Up GAMS/Madara

There is no special procedure to install GAMS/Madara dor Odroid. Just follow the same procedure for *Raspberry Pi 3*, using Ubuntu Mate 16.04.

3.5.7 Setting Up Peripherals

GPS

describe here how to install GPS drivers.

oCam-1MGN-U Global Shutter

describe here how to install the oCam-1MGN-U Global Shutter camera.

Wifi

The recommended wifi dongle is called Wifi module 3 because it has deattachable antenna

It works out of the box for Ubuntu Mate. No further installation or configuration is required.

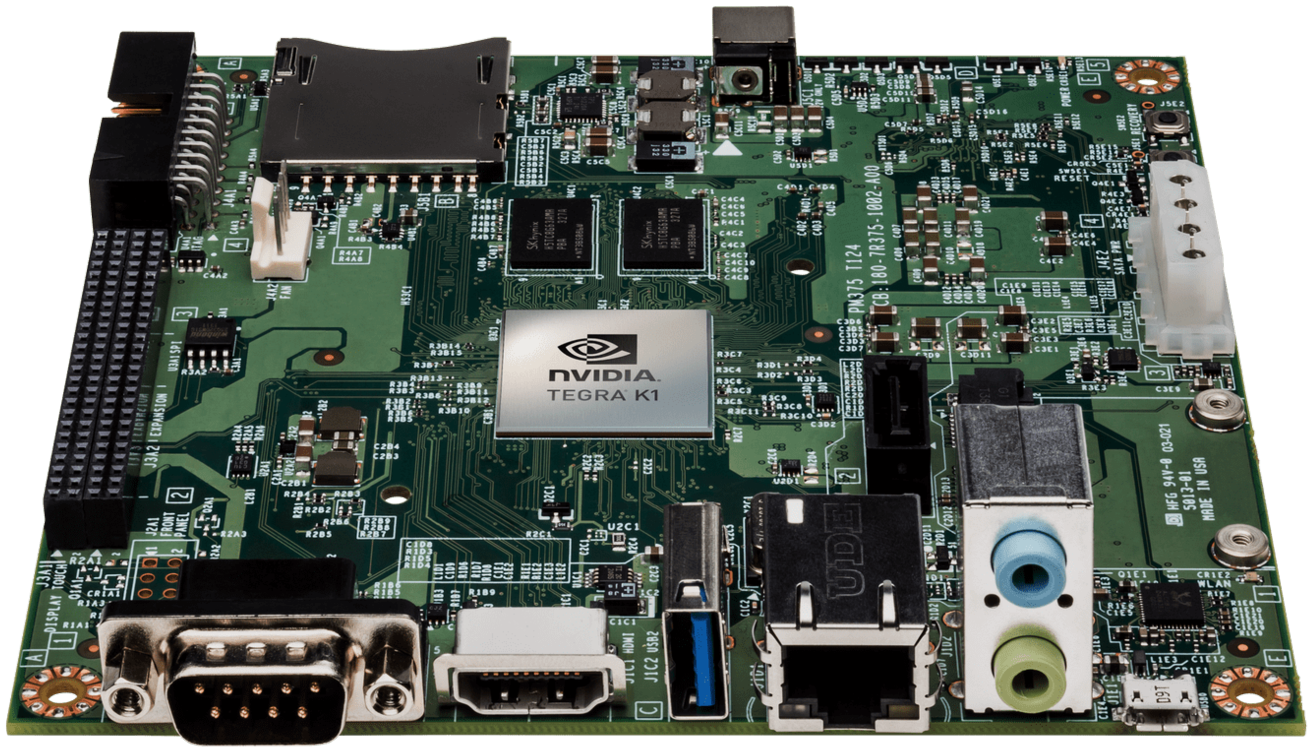
USB IO Board

describe here how to install and use the USB IO Board.

3.6 NVIDIA Jetsons Boards

3.6.1 Jetson TK1 Module

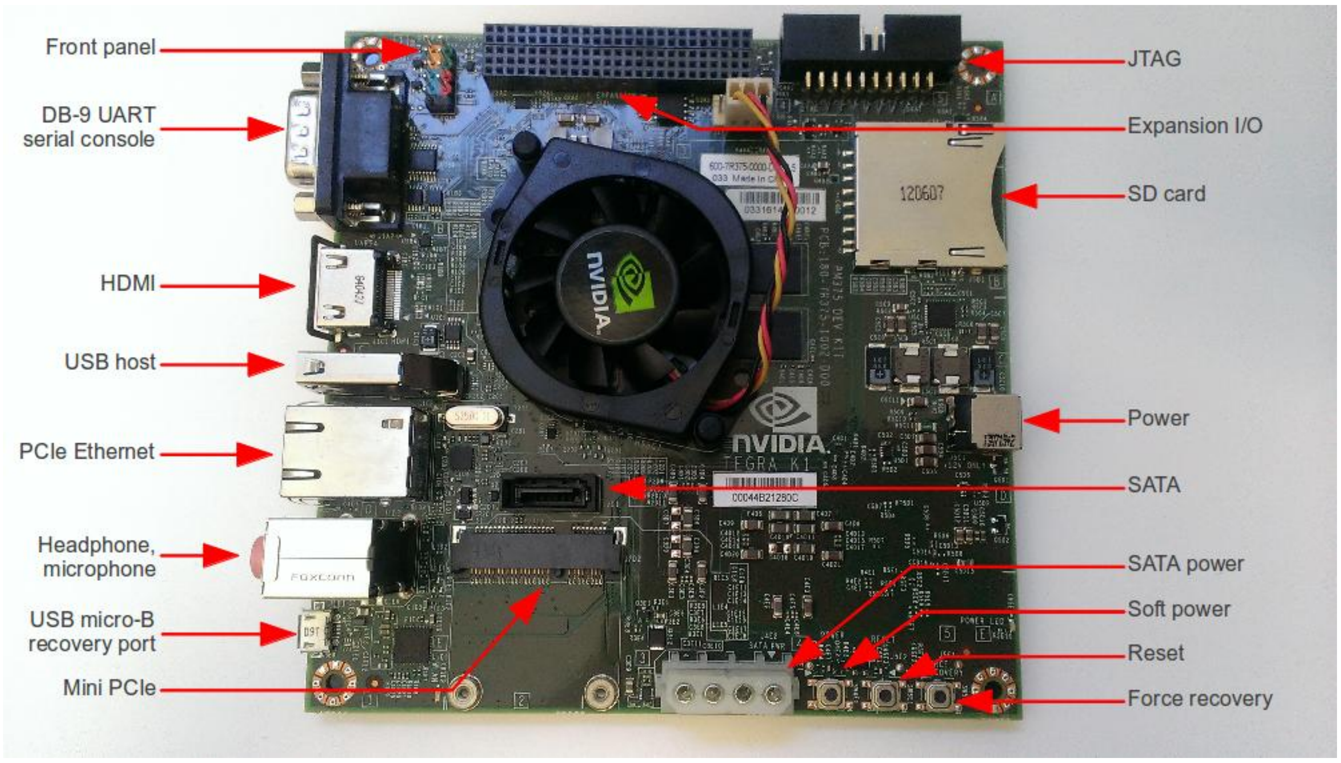
NVIDIA Tegra K1 (TK1) is NVIDIA's first mobile processor to have the same advanced features & architecture as a modern desktop GPU while still using the low power draw of a mobile chip. Therefore Tegra K1 allows embedded devices to use the exact same CUDA code that would also run on a desktop GPU (used by over 100,000 developers), with similar levels of GPU-accelerated performance as a desktop[1].



[1] Jetson TK1 Wiki

Specifications

- **Tegra K1 SOC**
  - NVIDIA Kepler GPU with 192 CUDA Cores
  - NVIDIA 4-Plus-1™ Quad-Core ARM® Cortex™-A15 CPU
- 2 GB x16 Memory with 64-bit Width
- 16 GB 4.51 eMMC Memory
- 1 Half Mini-PCIE Slot
- 1 Full-Size SD/MMC Connector
- 1 Full-Size HDMI Port
- 1 USB 2.0 Port, Micro AB
- 1 USB 3.0 Port, A
- 1 RS232 Serial Port
- 1 ALC5639 Realtek Audio Codec with Mic
- In and Line Out
- 1 RTL8111GS Realtek GigE LAN
- 1 SATA Data Port
- SPI 4 MByte Boot Flash



The following signals are available through an expansion port:

- DP/LVDS
- Touch SPI 1x4 + 1x1 CSI-2
- GPIOs
- UART
- HSIC

- i2c

Installing JetPack L4T

JetPack (the Jetson SDK) is an on-demand all-in-one package that bundles and installs all software tools required to develop for the NVIDIA® Jetson Embedded Platform (including flashing the Jetson Developer Kit). JetPack includes host and target developer tools, APIs and packages (OS images, tools, APIs, middleware, samples, documentation including compiling samples) to enable developers to jump-start their development environment for developing with the Jetson Embedded Platform. The latest release of JetPack runs on an Ubuntu 14.04 Linux 64-bit host system and supports the Jetson TK1 Developer Kit[1].

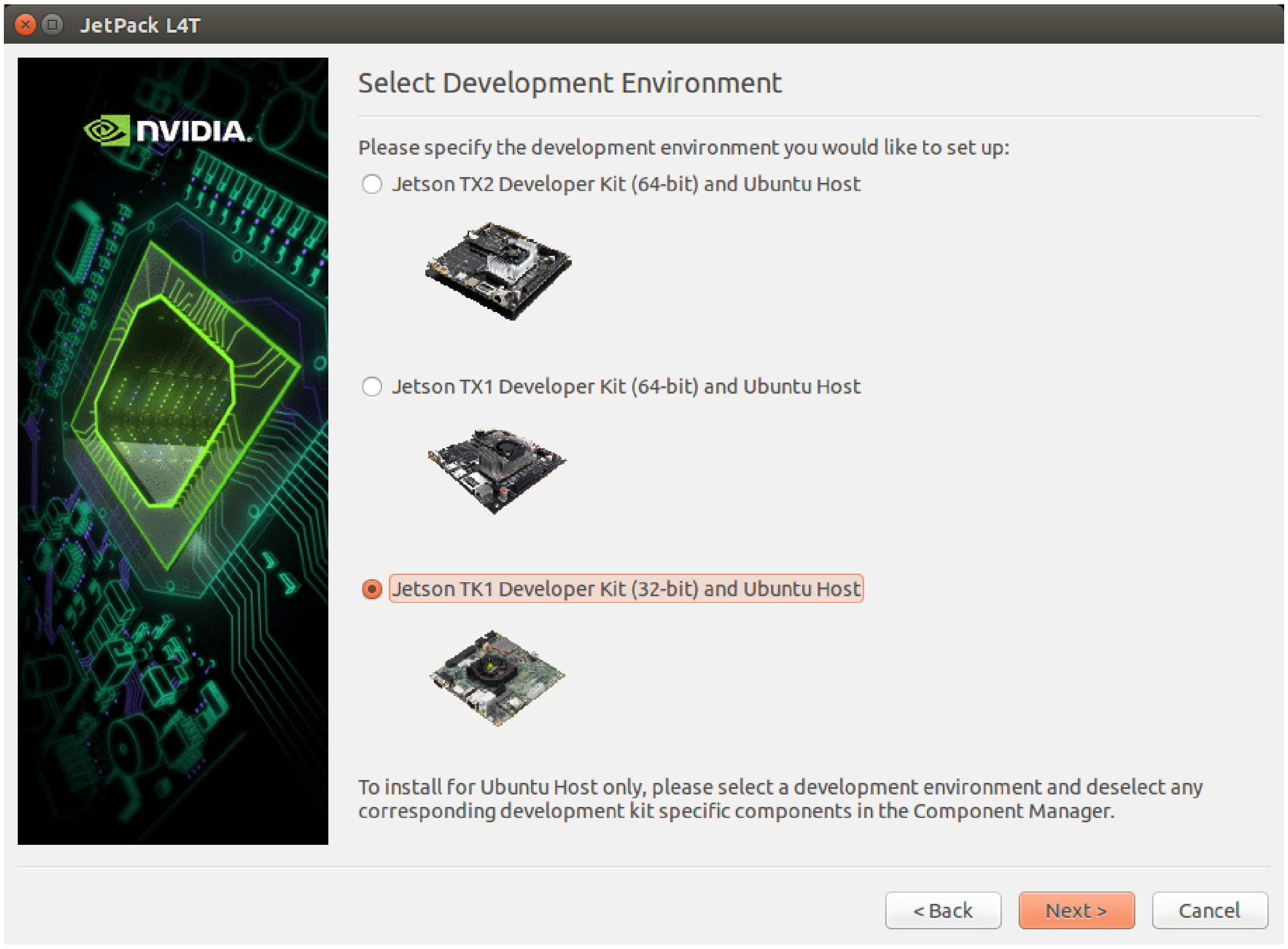
Setting up the local host

In order to flash the Jetson board with the SDK, the first step consists of installing Jetpack in a local machine. The local machine can set up all the environment before flashing the board with the SDK. From an Ubuntu 14.04 PC 64 bit host computer, you simply download the JetPack LT4 3.0 installer with the latest OS image from NVIDIA Web site (you'll have to sign in with your developer account) and follow the instructions in the setup guide. After downloading JetPack, install it in your local machine (not in the Jetson board).

```
$ chmod +x JetPack-L4T-3.0-linux-x64.run
$ ./JetPack-L4T-3.0-linux-x64.run
```

The process starts asking the folder to install JetPack and the board you are using to install the SDK, as follows the image:

[1] JetPack release notes



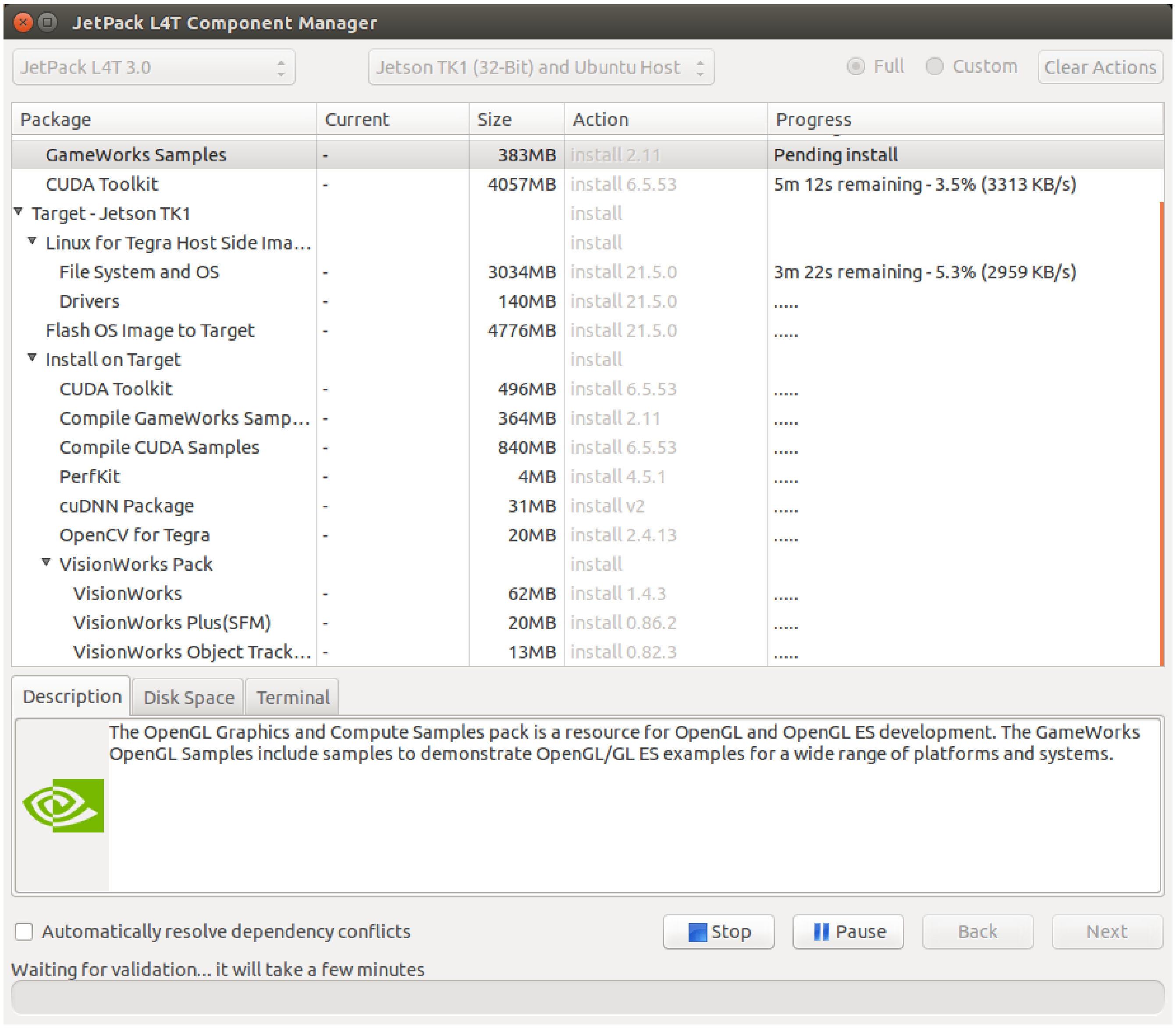
Select Development Environment

Please specify the development environment you would like to set up:

- ○ Jetson TX2 Developer Kit (64-bit) and Ubuntu Host
- ○ Jetson TX1 Developer Kit (64-bit) and Ubuntu Host
- ● Jetson TK1 Developer Kit (32-bit) and Ubuntu Host [✓]

To install for Ubuntu Host only, please select a development environment and deselect any corresponding development kit specific components in the Component Manager.

Select **JETSON TK1 Developer Kit (32-bit) and Ubuntu host** option and click `Next`. Keep configuring according to your needs. After finishing the selections, click `Next` to start downloading all packages, as the image:



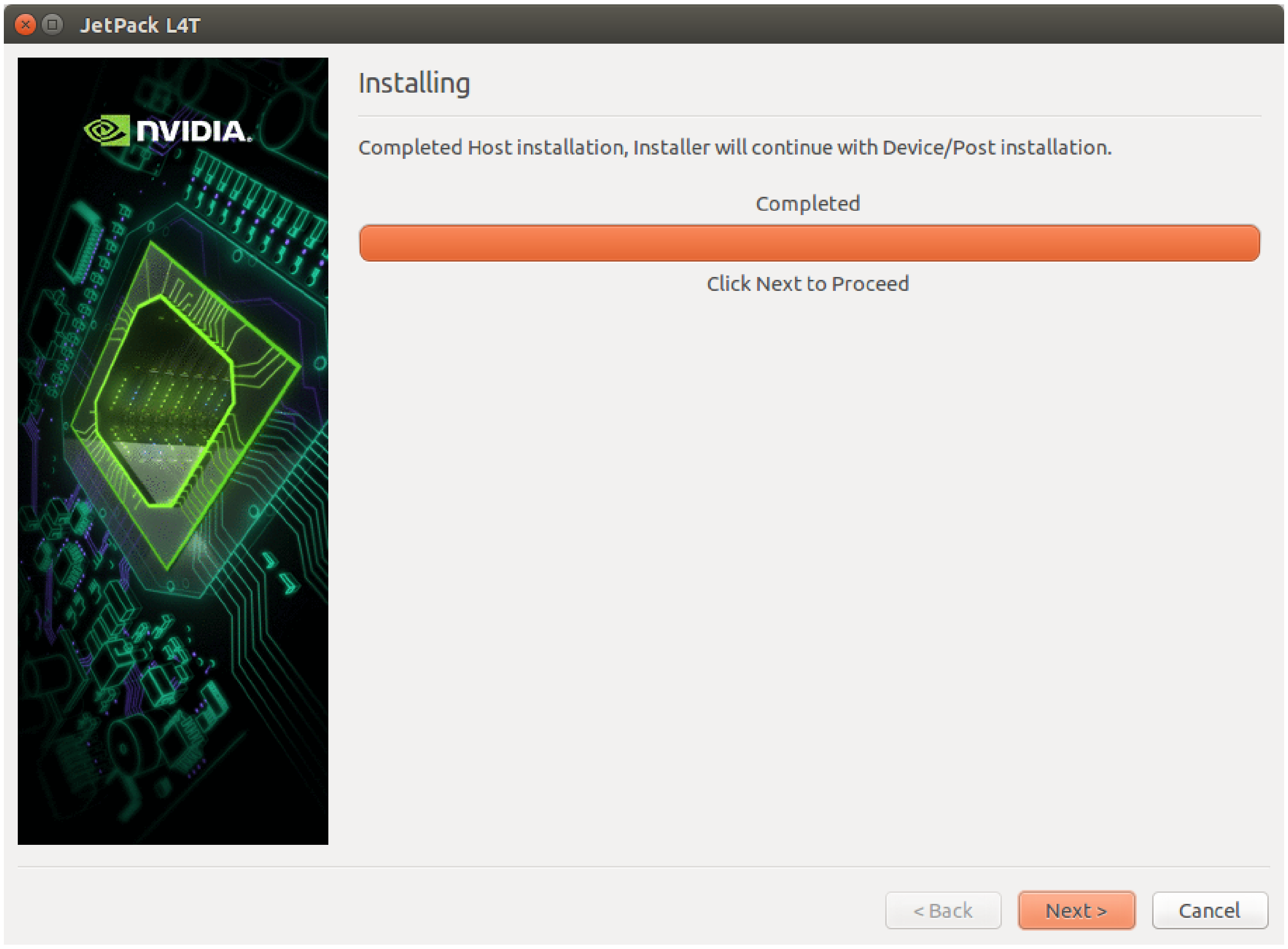
Package	Current	Size	Action	Progress
GameWorks Samples	-	383MB	install 2.11	Pending install
CUDA Toolkit	-	4057MB	install 6.5.53	5m 12s remaining - 3.5% (3313 KB/s)
▼ Target - Jetson TK1			install	
▼ Linux for Tegra Host Side Ima…			install	
File System and OS	-	3034MB	install 21.5.0	3m 22s remaining - 5.3% (2959 KB/s)
Drivers	-	140MB	install 21.5.0	.....
Flash OS Image to Target	-	4776MB	install 21.5.0	.....
▼ Install on Target			install	
CUDA Toolkit	-	496MB	install 6.5.53	.....
Compile GameWorks Samp…	-	364MB	install 2.11	.....
Compile CUDA Samples	-	840MB	install 6.5.53	.....
PerfKit	-	4MB	install 4.5.1	.....
cuDNN Package	-	31MB	install v2	.....
OpenCV for Tegra	-	20MB	install 2.4.13	.....
▼ VisionWorks Pack			install	
VisionWorks	-	62MB	install 1.4.3	.....
VisionWorks Plus(SFM)	-	20MB	install 0.86.2	.....
VisionWorks Object Track…	-	13MB	install 0.82.3	.....

The OpenGL Graphics and Compute Samples pack is a resource for OpenGL and OpenGL ES development. The GameWorks OpenGL Samples include samples to demonstrate OpenGL/GL ES examples for a wide range of platforms and systems.

☐ Automatically resolve dependency conflicts

Waiting for validation... it will take a few minutes

After downloading all packages, it starts to installing them in your local machine. It took about 20 minutes in my machine… When the installation in the local host is finished, the following image is presented:



Installing

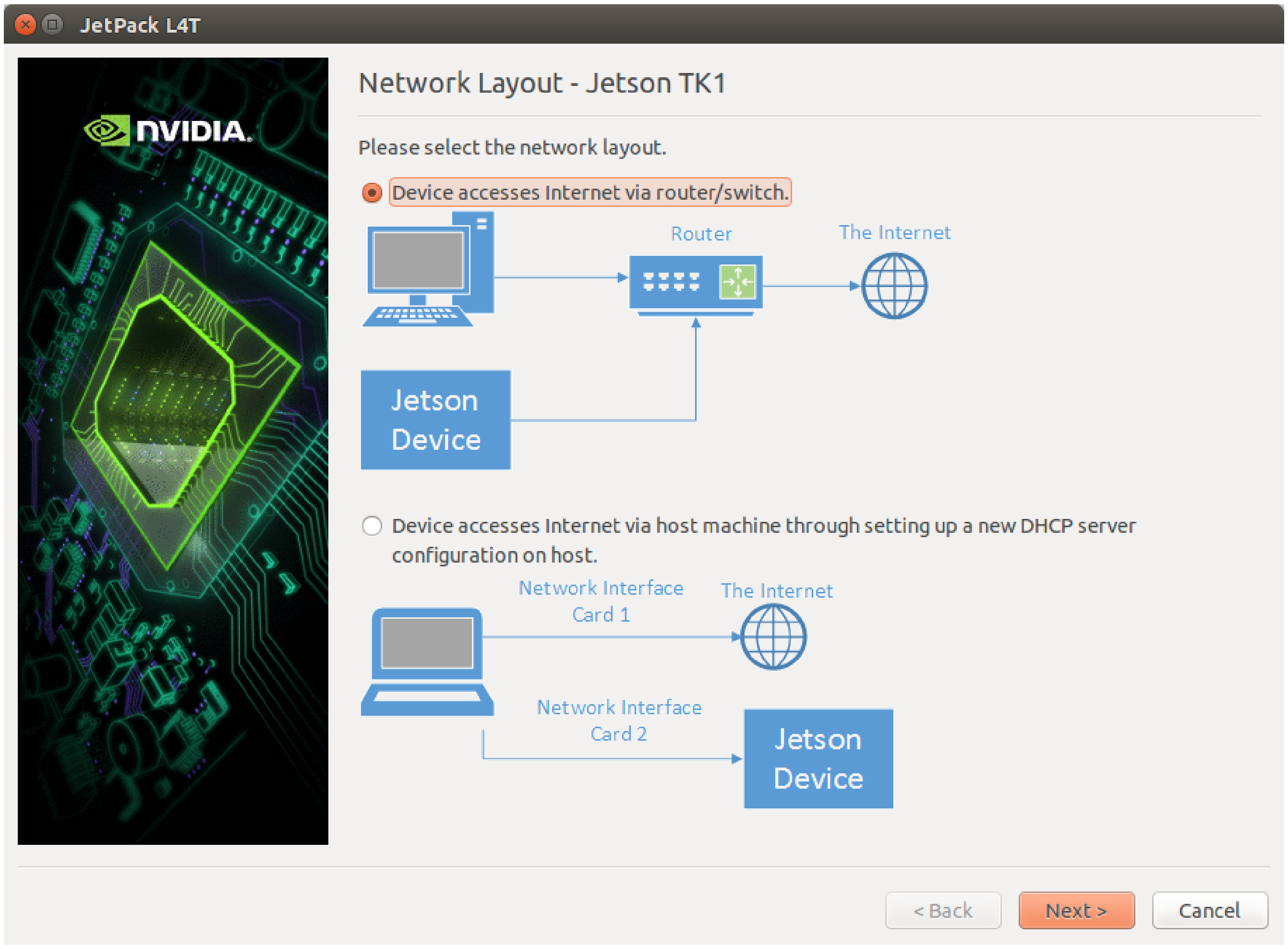
Completed Host installation, Installer will continue with Device/Post installation.

Completed

Click Next to Proceed

Flashing the Jetson board

Next step is to configure how the binaries are transmited to the Jetson board. Thus, JetPack asks what is the layout of the network to transmit the data. Select **Device accesses internet via houter/switch.** and click `Next`, as the image:

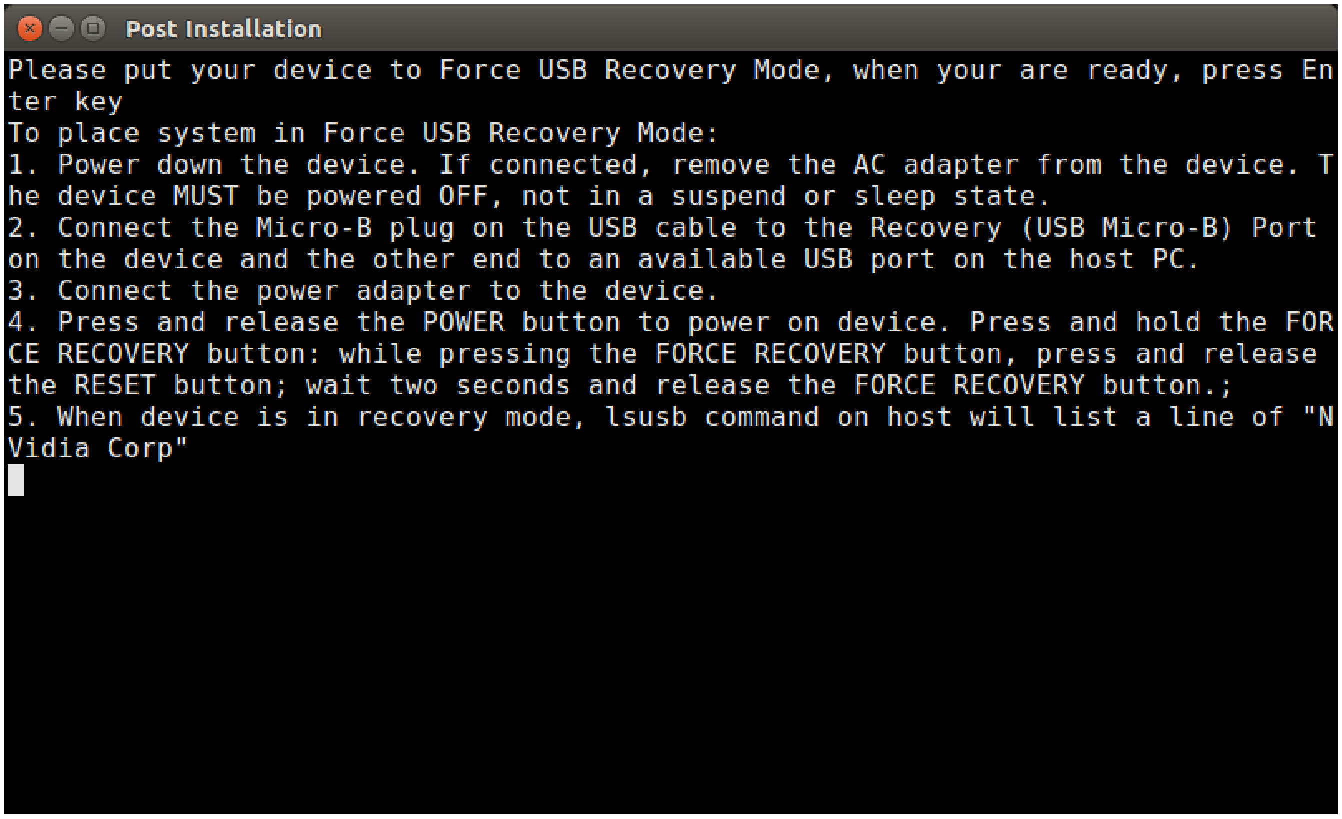


Please select the network layout.

[✓ Device accesses Internet via router/switch.]

Device accesses Internet via host machine through setting up a new DHCP server configuration on host.

JetPack then asks you to put the Jetson board in the *Recovery Mode*, by powering down the device (in case Jetson is on), connecting the micro-USB cable in the recovery port and in the USB of the local host, pressing and holding the FORCE RECOVERY button while turning the board on, as the image:

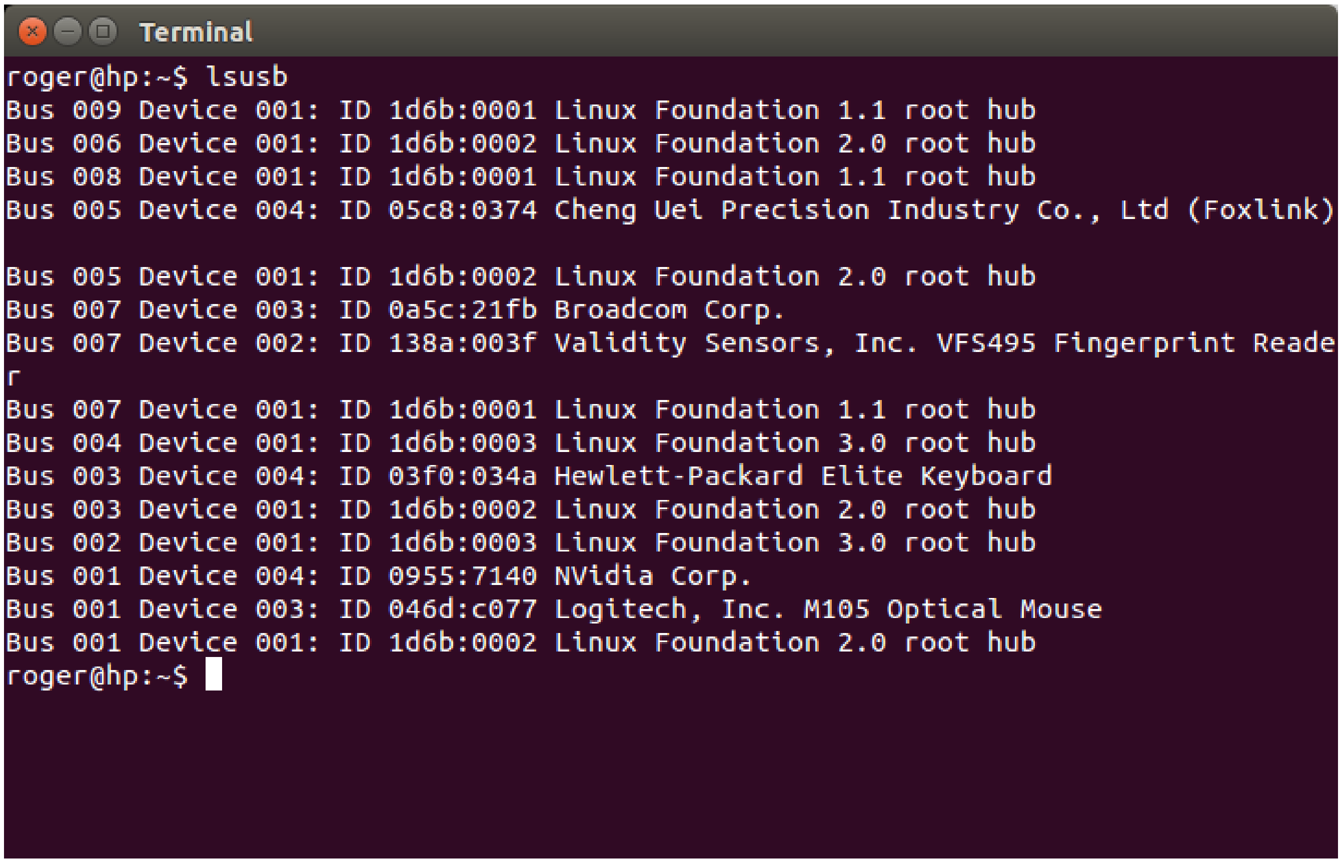


```
Please put your device to Force USB Recovery Mode, when your are ready, press En
ter key
To place system in Force USB Recovery Mode:
1. Power down the device. If connected, remove the AC adapter from the device. T
he device MUST be powered OFF, not in a suspend or sleep state.
2. Connect the Micro-B plug on the USB cable to the Recovery (USB Micro-B) Port
on the device and the other end to an available USB port on the host PC.
3. Connect the power adapter to the device.
4. Press and release the POWER button to power on device. Press and hold the FOR
CE RECOVERY button: while pressing the FORCE RECOVERY button, press and release
the RESET button; wait two seconds and release the FORCE RECOVERY button.;
5. When device is in recovery mode, lsusb command on host will list a line of "N
Vidia Corp"
```

In order to check if the Jetson board is set in recovery mode, open the terminal in the local host and type:

```
$ lsusb
```

If the board is in revery mode, you should see the Jetson listed as NVidia (*ID 0955:7140 NVidia Corp.*) in the output, as the image:



```
roger@hp:~$ lsusb
Bus 009 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 004: ID 05c8:0374 Cheng Uei Precision Industry Co., Ltd (Foxlink)

Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 003: ID 0a5c:21fb Broadcom Corp.
Bus 007 Device 002: ID 138a:003f Validity Sensors, Inc. VFS495 Fingerprint Reade
r
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 03f0:034a Hewlett-Packard Elite Keyboard
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 0955:7140 NVidia Corp.
Bus 001 Device 003: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
roger@hp:~$
```

If you don't see the Jetson using `lsusb`, then the device will not be flashed. In case the Jetson appears in `lsusb`, return to the screen with the instructions and press `Enter` to start flashing the OS image in the board.

References

Setting Up the OS

NVIDIA Jetson TK1 exhibits a lot of promise with lots of raw performance for its form factor and intended use, with low power consumption to boot. But as is typical with most of these types of products, the "out of the box" experience needs some help. A missing part in TK1 is a support out of the box for WiFi or Bluetooth. For people coming from commodity PCs, tablets, phones and such this is a little confusing. Usually one just installs a driver and the device starts to work. In the case of the Jetson, the actual signals on the board need to be played with a little, as well as having the driver issue[1]. In order to overcome such issues, a new Linux kernel named **Grinch** includes a lot of the features to which most desktop users are accustomed.

Installing Grinch kernel

The Grinch Kernel for L4T provides over 60 changes and additions to the kernel, including fixes, configuration, module and firmware support to the stock kernel. The kernel is written and supported by Jetson Forum user Santyago. For further information about the Grinch Kernel, please see the NVidia Jetson Forum.

In order to install the Grinch kernel, download the `installGrinch.sh` file and run as:

[1] Installing Grinch L4T

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/jetson/scripts/installGrinch.sh
$ chmod +x installGrinch.sh
$ ./installGrinch.sh
```

Running these command will download the kernel image and install it in Jetson board. After the install script completes, reboot the Jetson TK1.

Enabling USB 3.0

By default, the USB port in Jetson TK1 is not compatible to 3.0 devices. In order to enable the USB 3.0 support, you have to configure the `extlinux.conf` file, changing the parameter `usb_port_owner_info=0` to `usb_port_owner_info=2`. In order to do so, run:

```
$ sudo gedit /boot/extlinux/extlinux.conf
```

The `extlinux.conf` file looks like:

```
TIMEOUT 30
DEFAULT primary

MENU TITLE Jetson-TK1 eMMC boot options

LABEL primary
    MENU LABEL primary kernel
    LINUX /boot/zImage
    FDT /boot/tegra124-jetson_tk1-pm375-000-c00-00.dtb
    APPEND console=ttyS0,115200n8 console=tty1 no_console_suspend=1 lp0_
↪vec=2064@0xf46ff000 mem=2015M@2048M memtype=255 ddr_die=2048M@2048M section=256M␣
↪pmuboard=0x0177:0x0000:0x02:0x43:0x00 tsec=32M@3913M otf_
↪key=c75e5bb91eb3bd947560357b64422f85 usbcore.old_scheme_first=1 core_edp_mv=1150␣
↪core_edp_ma=4000 tegraid=40.1.1.0.0 debug_uartport=lsport,3 power_supply=Adapter␣
↪audio_codec=rt5640 modem_id=0 android.kerneltype=normal fbcon=map:1 commchip_id=0␣
↪usb_port_owner_info=2 lane_owner_info=6 emc_max_dvfs=0 touch_id=0@0 board_
↪info=0x0177:0x0000:0x02:0x43:0x00 net.ifnames=0 root=/dev/mmcblk0p1 rw rootwait␣
↪tegraboot=sdmmc gpt
```

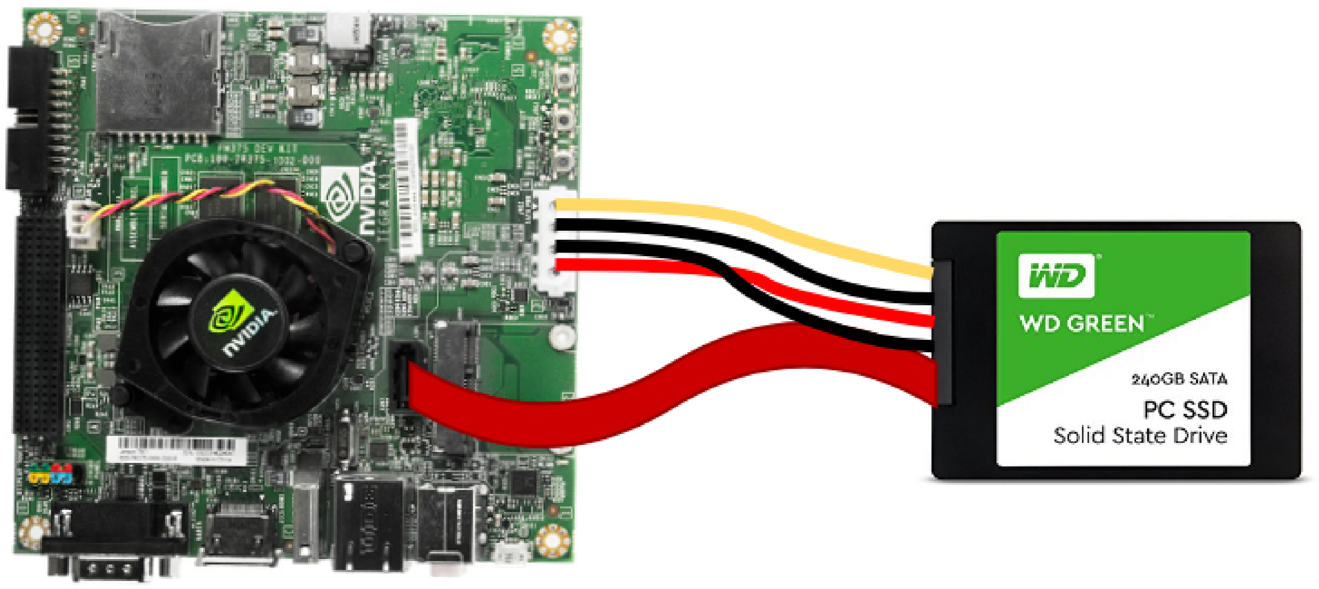
Look for `usb_port_owner_info=0` in the file and replace it by `usb_port_owner_info=2`. Finally, save the file and exit.

References

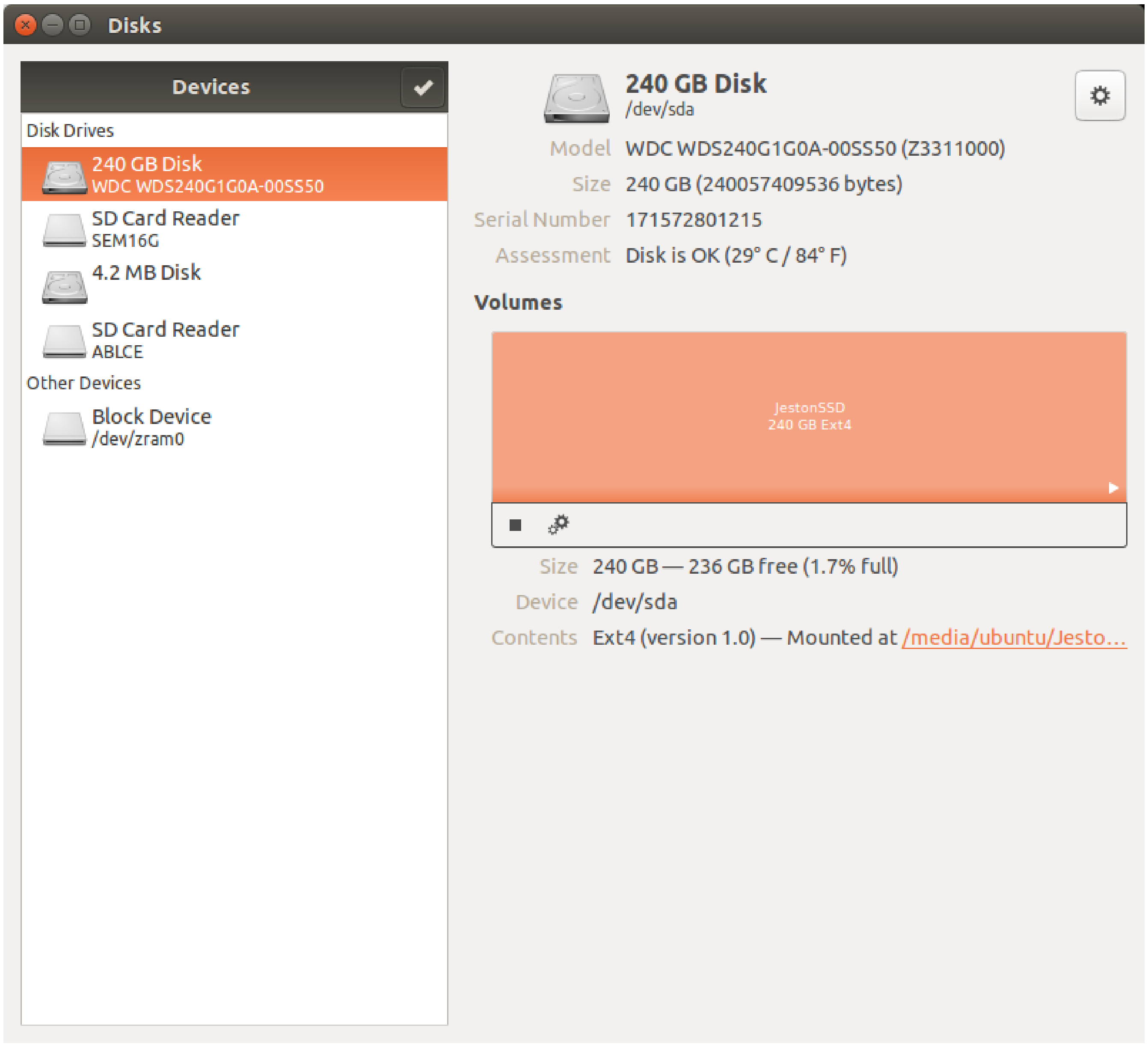
Post Installation

Adding a SSD Disk

In order to add a SSD disk in Jetson board, we have to connect a SATA cable and a power cable in the board as illustrated in the image below:



After connecting the SSD disk, we format it using `ext4` file system by logging into Ubuntu and accessing `Disks` application. Inside the application, select the SSD disk and click on the gear and select `Format` (or press `Shift+Ctrl+F`). Then, add a name to the disk (e.g. `JetsonSSD`) and click in `Format`, as the images below:

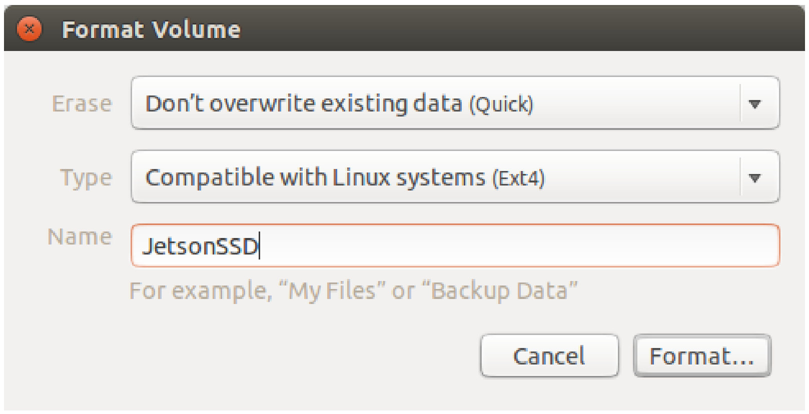


240 GB Disk
/dev/sda

Model	WDC WDS240G1G0A-00SS50 (Z3311000)
Size	240 GB (240057409536 bytes)
Serial Number	171572801215
Assessment	Disk is OK (29° C / 84° F)

JestonSSD
240 GB Ext4

Size	240 GB — 236 GB free (1.7% full)
Device	/dev/sda
Contents	Ext4 (version 1.0) — Mounted at /media/ubuntu/Jesto…



Erase	Don't overwrite existing data (Quick)
Type	Compatible with Linux systems (Ext4)
Name	JetsonSSD

For example, "My Files" or "Backup Data"

After formating the SSD disk, we have to mount it at startup. In order to do it, we have to add a call in `/etc/fstab` with the mounting point. In order to add this line, we have to discover the uuid of the device. With the disk manually mounted, we run the `mount` command to discover where the SSD is mounted, obtaining:

```
$ mount
/dev/sda on /media/ubuntu/JetsonSSD type ext4 (rw,nosuid,nodev,uhelper=udisk2)
```

Knowing the mounting local of the SSD disk (`/dev/sda`), we have to discover its uuid. To discover the uuid we run:

```
$ ls -al /dev/disk/by-uuid
lrwxrwxrwx 1 root root   9 Dec 31 21:00 ac183b24-3e75-4190-bcb7-32160e9a7c55 -> ../../
↪sda
```

Having the uuid of the disk we can add a line to the `/etc/fstab` with a call to the mounting point. Running the command:

```
$ sudo gedit /etc/fstab
```

We add the following line to the file:

```
/dev/disk/by-uuid/ac183b24-3e75-4190-bcb7-32160e9a7c55 /media/JetsonSSD ext4 defaults␣
↪0 2
```

Save the file and close it. Next time Ubuntu is started, the SSD disk will be mounted at startup.

Important Packages

Here we add several packages that should be installed to work in Jetson. All packages are installed via `apt-get`. In order to easily install all packages a script was created and can be downloaded by running:

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/jetson/scripts/additionalPackages.sh
$ chmod +x additionalPackages.sh
$ ./additionalPackages.sh
```

The script installs the following packages:

- Informational list of build-essential packages (build-essential)
- Cross-platform, open-source make system (CMake)
- Curses based user interface for CMake (cmake-curses-gui)
- GNU C++ compiler (G++)
- Set of I2C tools for Linux
- Userspace I2C programming library development files
- Distributed revision control system (Git)
- Multiple GNOME terminals in one window
- Terminal multiplexer with VT100/ANSI terminal emulation (Screen)

Cloning and Restoring Image

In many cases there is a large amount of work that goes into creating a working Jetson development system. This can include such things as loading drivers, compiling source files and libraries, system configuration, and so on. During development, sometimes things can go south and leave the system in a non-working state. Wouldn’t it be great if you could take a snapshot of a baseline system for just such occasions? Well, you can! This is known as “cloning” the system[1]. In this section we describe how to save and restore the system image from a Jetson to a file in the local host. The host must be the same that contains the JetPack installer that flashed the Jetson.

Cloning the system image to localhost

In order to clone the image to the localhost, first you navigate to the proper directory on the localhost to begin the process. Open a Terminal and go to the JetPack install directory (e.g., `/home/roger/JetPack/`). The correct folder has the following content:

```
GameWorksOpenGLSamples
JetPack-L4T-3.0-linux-x64.run
JetPack_Uninstaller
TK1
_installer
host-x64-linux-public-3.7.224-e982b7b
jetpack_docs
manifest.json
repository.json
tmp
update.lock
```

From this folder, navigate to the `bootloader` folder with:

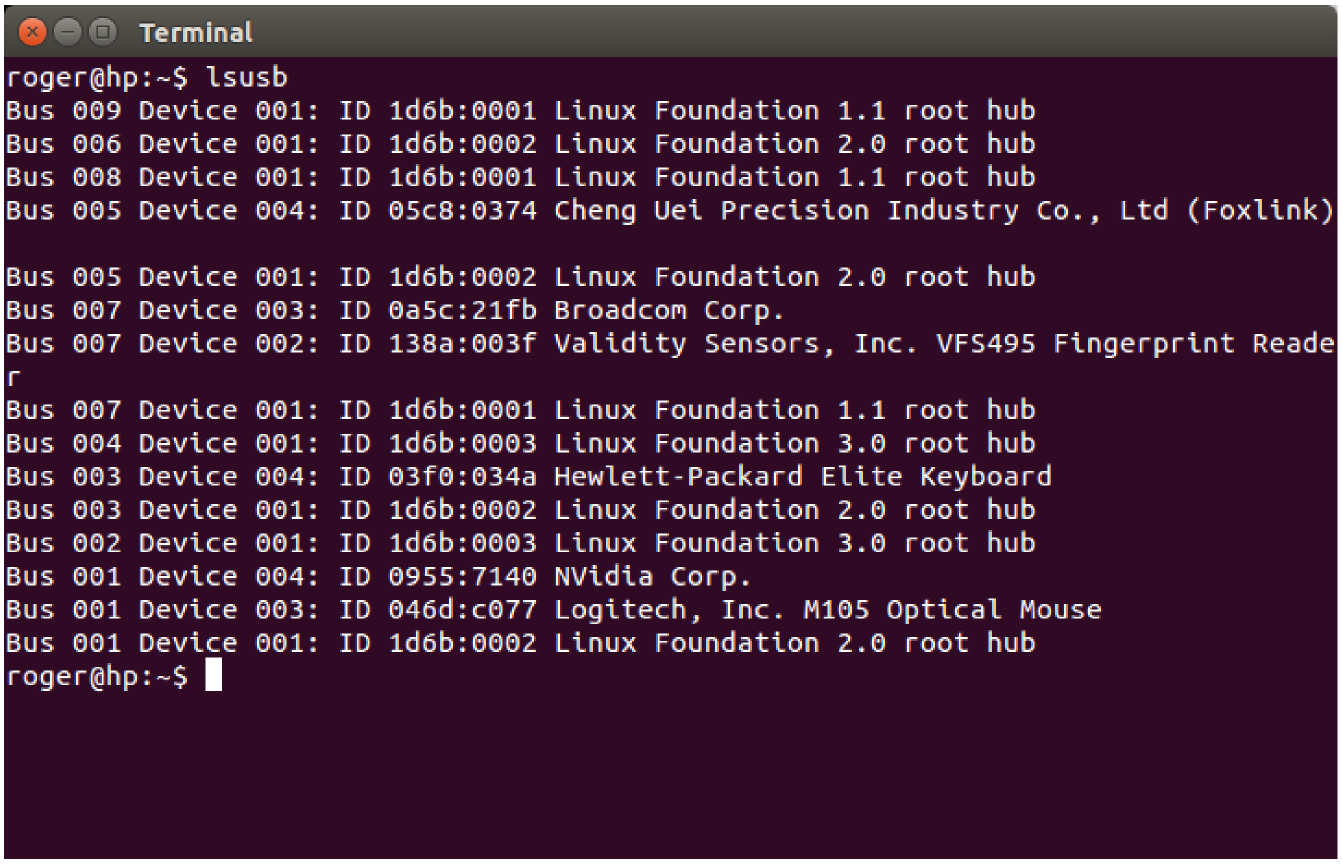
```
$ cd TK1/Linux_for_Tegra_tk1/bootloader
```

When in the correct folder, turn on the Jetson board in the *Recovery Mode* by powering down the device (in case Jetson is on), connecting the micro-USB cable in the recovery port and in the USB of the local host, pressing and holding the FORCE RECOVERY button while turning the board on. In order to check if Jetson is in *Recovery Mode*, in your localhost, run:

```
$ lsusb
```

If the board is in *Recovery Mode*, you should see the Jetson listed as NVidia (ID 0955:7140 NVidia Corp.) in the output, as the image:

[1] JetsonHacks: Clone Image Tk1



```
roger@hp:~$ lsusb
Bus 009 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 004: ID 05c8:0374 Cheng Uei Precision Industry Co., Ltd (Foxlink)

Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 003: ID 0a5c:21fb Broadcom Corp.
Bus 007 Device 002: ID 138a:003f Validity Sensors, Inc. VFS495 Fingerprint Reade
r
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 03f0:034a Hewlett-Packard Elite Keyboard
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 0955:7140 NVidia Corp.
Bus 001 Device 003: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
roger@hp:~$
```

Having the Jetson board in *Recovery Mode*, run:

```
$ sudo ./nvflash --read APP clone.img --bl ardbeg/fastboot.bin --go
```

When running this command, an image of the Jetson board starts to be recorded into `clone.img` file.

Restoring the system image to Jetson

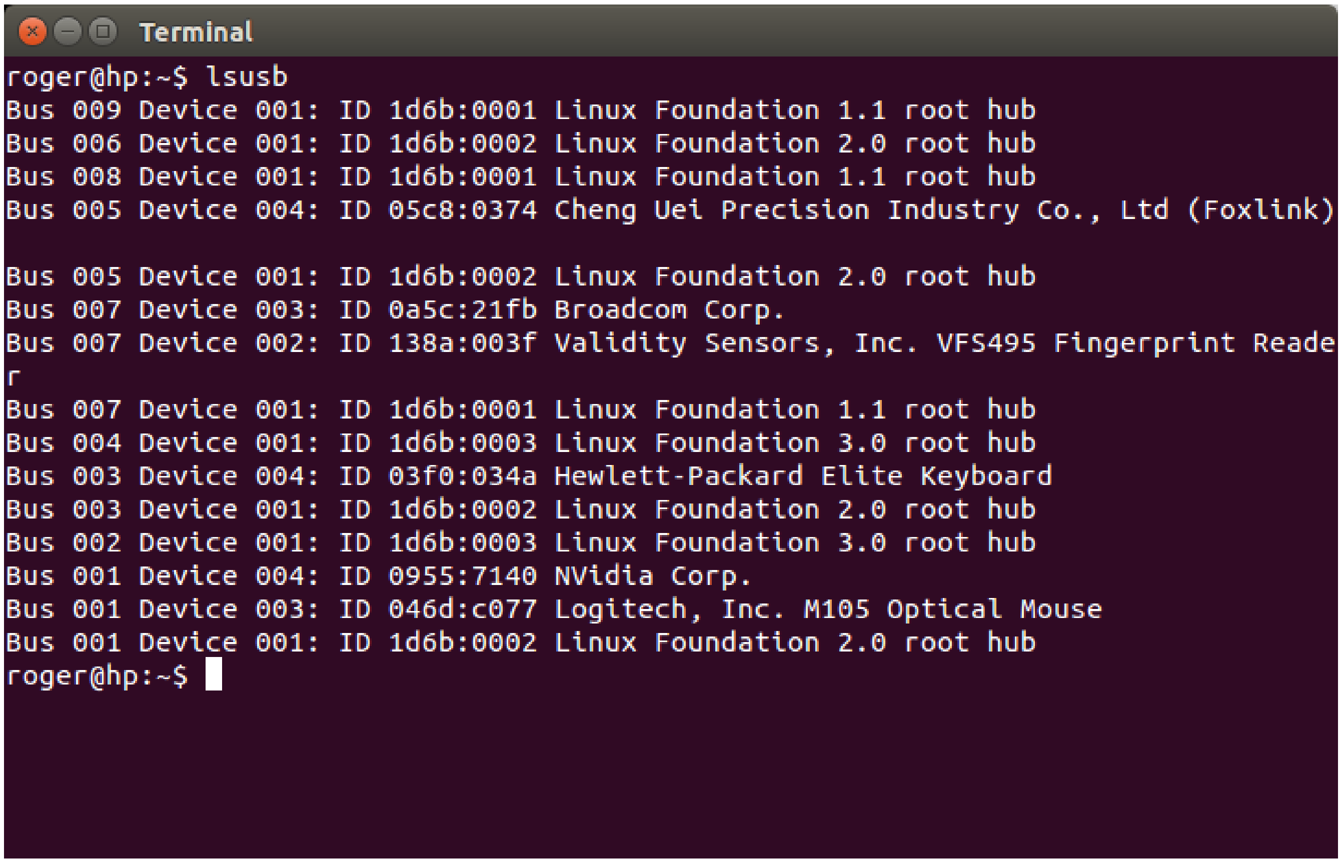
In order to restore the image from the localhost to the Jetson, first you navigate to the proper directory on the localhost to begin the process. Open a Terminal and go to the JetPack install directory (e.g., `/home/roger/JetPack/`). From this folder, navigate to the `Linux_for_Tegra_tk1` folder with:

```
$ cd TK1/Linux_for_Tegra_tk1/
```

When in the correct folder, turn on the Jetson board in the *Recovery Mode* by powering down the device (in case Jetson is on), connecting the micro-USB cable in the recovery port and in the USB of the local host, pressing and holding the FORCE RECOVERY button while turning the board on. In order to check if Jetson is in *Recovery Mode*, in your localhost, run:

```
$ lsusb
```

If the board is in *Recovery Mode*, you should see the Jetson listed as NVidia (ID 0955:7140 NVidia Corp.) in the output, as the image:



```
roger@hp:~$ lsusb
Bus 009 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 004: ID 05c8:0374 Cheng Uei Precision Industry Co., Ltd (Foxlink)

Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 003: ID 0a5c:21fb Broadcom Corp.
Bus 007 Device 002: ID 138a:003f Validity Sensors, Inc. VFS495 Fingerprint Reade
r
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 03f0:034a Hewlett-Packard Elite Keyboard
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 0955:7140 NVidia Corp.
Bus 001 Device 003: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
roger@hp:~$
```

Having the Jetson board in *Recovery Mode*, run:

```
$ sudo ./flash.sh -r -S 14580MiB jetson-tk1 mmcblk0p1
```

When running this command, the image stored in `system.img` in the `bootloader` folder will be flashed in the Jetson board. The `-r` flag skips building and reuse the existing `system.img` file. The partition size `-S 14580MiB` is the default that JetPack uses. When the flashing process termines, reboot the Jetson. The Jetson will be restored to the state of the original at the time of cloning.

Note: If you already have performed a cloning before restoring the image to the Jetson, you should rename the clone image to `system.img`. Otherwise, the image restored to the Jetson board contains the original SDK.

References

Robot Operating System (ROS) on Jetson TK1

Robot Operating System (ROS) is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. The ROS runtime “graph” is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server[1].

The primary goal of ROS is to support code reuse in robotics research and development so you can find a built-in package system. It is interesting to note that although ROS contains Operation System in the name, keep in mind that ROS is not an OS, a library, or an RTOS, but a framework using the concept of an OS. A good introduction is given in the freely available book named A Gentle Introduction to ROS by Jason O'Kane. The ROS Wiki also contains lots of tutorials to introduce you to its main concepts.

Installing ROS in Jetson TK1

JetsonPack for Jetson TK1 is based on Ubuntu 14.04 and thus, the ROS version to install is named *Indigo*. In order to install the ROS package, you can run the `installROS.sh` file as:

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/jetson/TK1/scripts/installROS.sh
$ chmod +x installROS.sh
$ ./installROS.sh
```

This script installs ROS Indigo Desktop and the Point Cloud Library.

Creating Catkin workspace to run ROS

Catkin packages can be built as a standalone project, in the same way that normal cmake projects can be built, but catkin also provides the concept of workspaces, where you can build multiple, interdependent packages together all at once. A catkin workspace is a folder where you modify, build, and install catkin packages[2]. In order to create a Catkin workspace, you should download and run the file `createCatkin.sh` as:

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/jetson/scripts/createCatkin.sh
$ chmod +x createCatkin.sh
$ ./createCatkin.sh
```

This script creates a structure in the home folder with the root workspace set in `/home/ubuntu/catkin_ws`.

Installing vision_opencv in ROS

By default, ROS has support to native OpenCV. However, when trying to compile a C++ code with a call to `cv_bridge`, the following error

Error: make[2]: *** No rule to make target '/usr/lib/arm-linux-gnueabihf/libopencv_videostab.so.2.4.8'. Stop.

In order to get rid of the error, we have to compile vision_opencv in our Catkin workspace. Hence, the first step is to clone the Indigo version of `vision_opencv` into the `catkin_ws/src` folder. Then, we have to compile the workspace, by running:

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ros-perception/vision_opencv.git
$ cd vision_opencv/
```

(continues on next page)

[1] ROS Introduction
[2] Catkin Workspaces

(continued from previous page)

```
$ git checkout indigo
$ cd ../..
$ catkin_make
```

Next time you compile a C++ code that contains a call to `cv_bridge`, no errors should appear.

Testing ROS installation

To check if the ROS is installed correctly, run:

```
$ roscore
```

And see it starts running correctly. In case of problem, you can check the log files by running:

```
$ roscd log
```

References

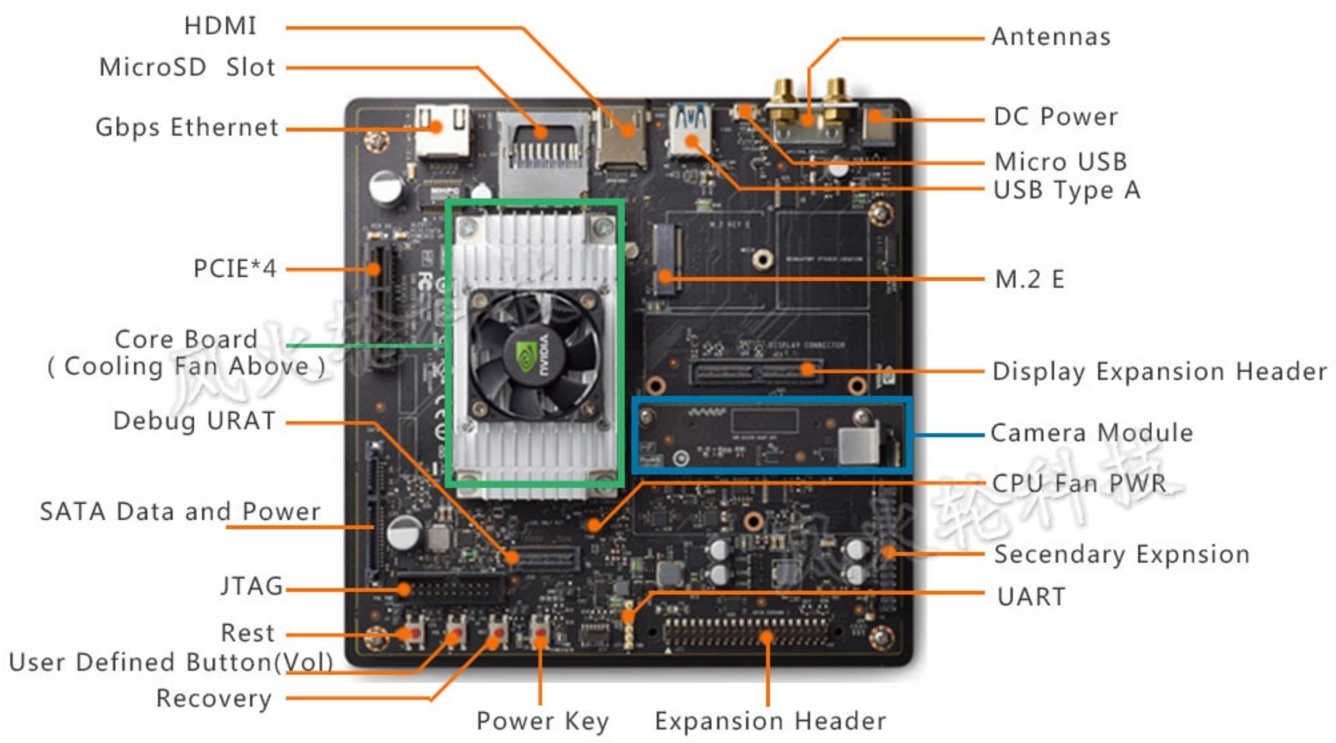
3.6.2 Jetson TX2 Module

NVIDIA Tegra X2 (TX2) is the fastest, most power-efficient embedded AI computing device. 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 8GB of memory and 59.7GB/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.[2].

[2] Jetson TX2

Specifications

- **Tegra X2 SOC**
  - NVIDIA Pascal™, 256 CUDA cores
  - HMP Dual Denver 2/2 MB L2 + Quad ARM A57/2 MB L2
- 8 GB 128 bit LPDDR4 59.7 GB/s
- 32 GB eMMC, SDIO, SATA
- 4K x 2K 60 Hz Encode (HEVC)
- 4K x 2K 60 Hz Decode (12-Bit Support)
- 2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4
- 1 USB 2.0 Port, Micro AB
- 1 USB 3.0 Port, A
- 1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth
- 50 mm x 87 mm (400-Pin)



The following signals are available through an expansion port:

- SPI
- GPIOs
- UART
- i2c
- i2s

Installing JetPack L4T

JetPack (the Jetson SDK) is an on-demand all-in-one package that bundles and installs all software tools required to develop for the NVIDIA® Jetson Embedded Platform (including flashing the Jetson Developer Kit). JetPack includes host and target developer tools, APIs and packages (OS images, tools, APIs, middleware, samples, documentation including compiling samples) to enable developers to jump-start their development environment for developing with the Jetson Embedded Platform. The latest release of JetPack runs on an Ubuntu 16.04 Linux 64-bit host system and supports the Jetson TX2 Developer Kit[1].

Setting up the local host

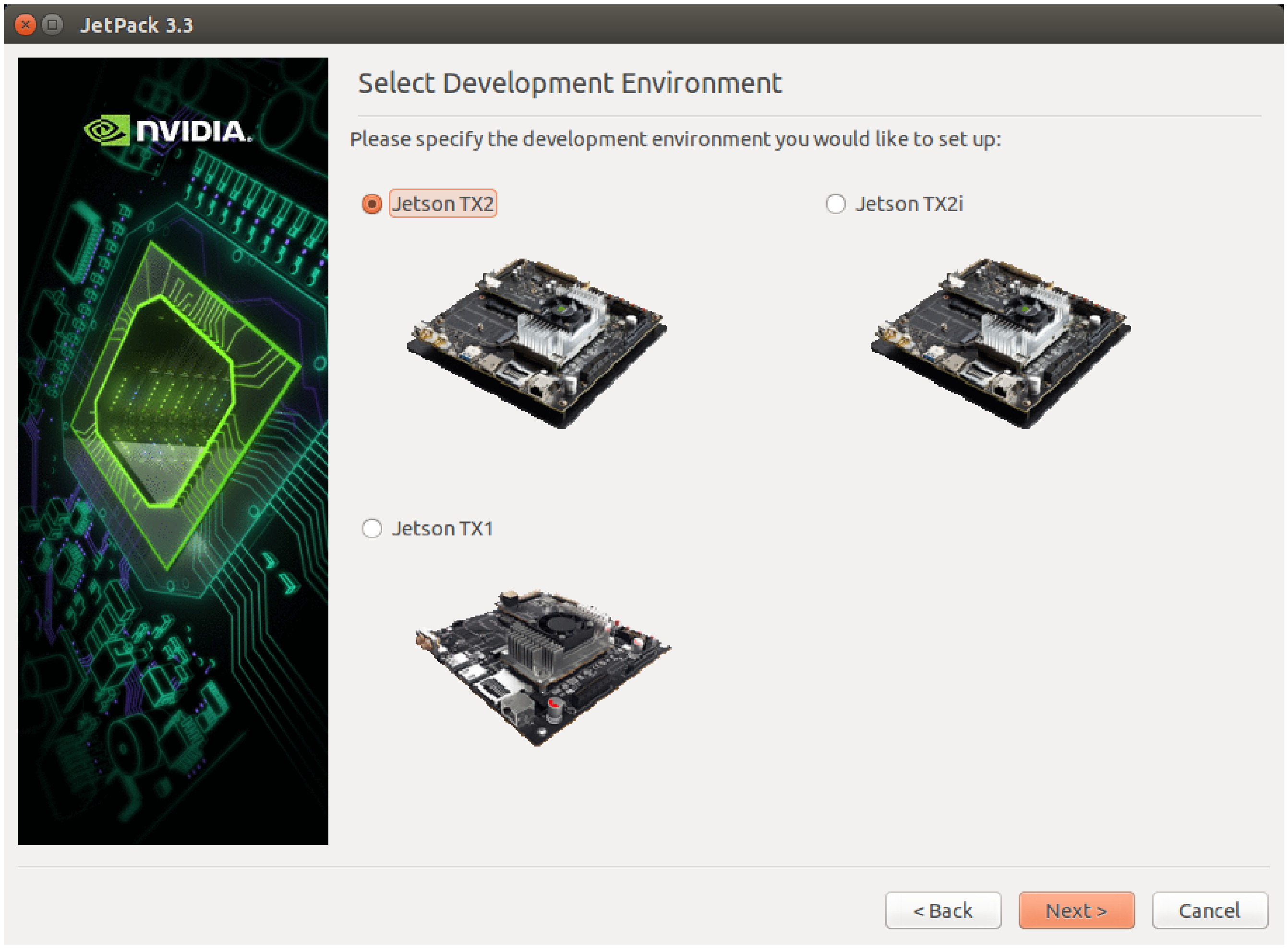
In order to flash the Jetson board with the SDK, the first step consists of installing Jetpack in a local machine. The local machine can set up all the environment before flashing the board with the SDK. From an Ubuntu 16.04 PC 64 bit host computer, you simply download the JetPack LT4 3.3 installer with the latest OS image from NVIDIA Web site

[1] JetPack release notes

(you'll have to sign in with your developer account) and follow the instructions in the setup guide. After downloading JetPack, install it in your local machine (not in the Jetson board).

```
$ chmod +x JetPack-L4T-3.3-linux-x64_b39.run
$ ./JetPack-L4T-3.3-linux-x64_b39.run
```

The process starts asking the folder to install JetPack and the board you are using to install the SDK, as follows the image:

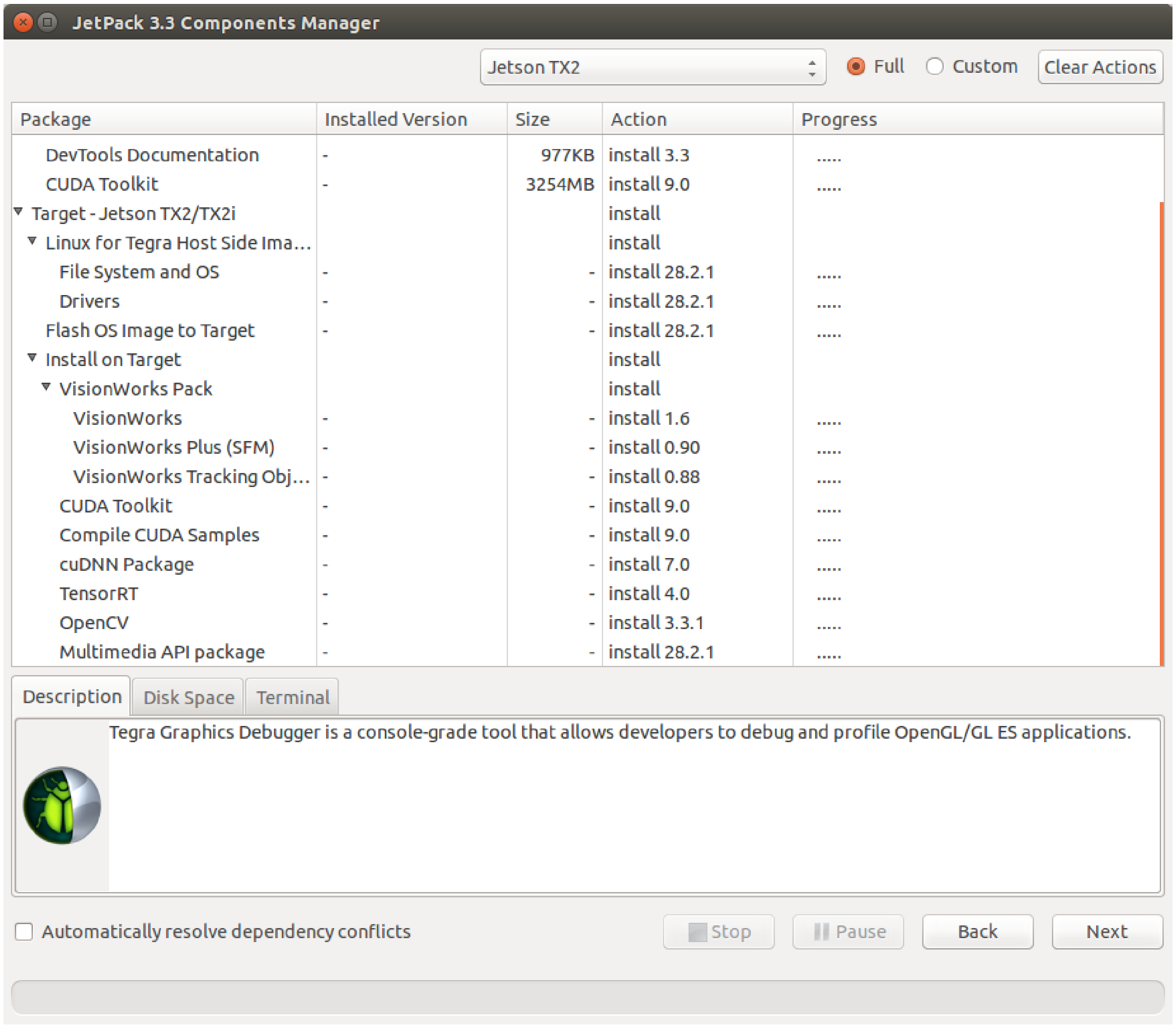


Select Development Environment

Please specify the development environment you would like to set up:

[✓ Jetson TX2] Jetson TX2i Jetson TX1

Select **Jetson TX2** option and click `Next`. Keep configuring according to your needs. After finishing the selections, click `Next` to start downloading all packages, as the image:

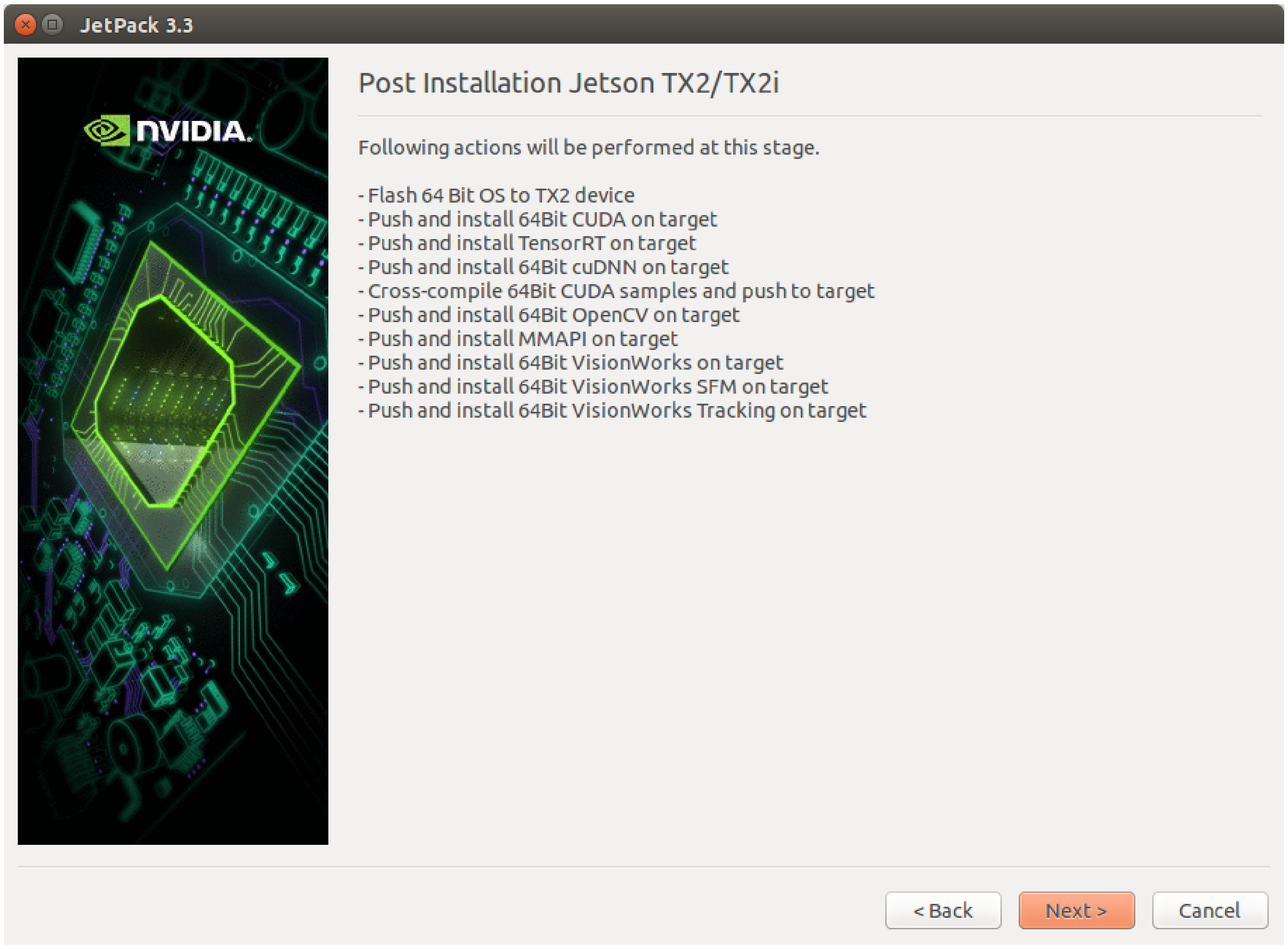


Package	Installed Version	Size	Action	Progress
DevTools Documentation	-	977KB	install 3.3	.....
CUDA Toolkit	-	3254MB	install 9.0	.....
▼ Target - Jetson TX2/TX2i			install	
▼ Linux for Tegra Host Side Ima…			install	
File System and OS	-	-	install 28.2.1	.....
Drivers	-	-	install 28.2.1	.....
Flash OS Image to Target	-	-	install 28.2.1	.....
▼ Install on Target			install	
▼ VisionWorks Pack			install	
VisionWorks	-	-	install 1.6	.....
VisionWorks Plus (SFM)	-	-	install 0.90	.....
VisionWorks Tracking Obj…	-	-	install 0.88	.....
CUDA Toolkit	-	-	install 9.0	.....
Compile CUDA Samples	-	-	install 9.0	.....
cuDNN Package	-	-	install 7.0	.....
TensorRT	-	-	install 4.0	.....
OpenCV	-	-	install 3.3.1	.....
Multimedia API package	-	-	install 28.2.1	.....

Tegra Graphics Debugger is a console-grade tool that allows developers to debug and profile OpenGL/GL ES applications.

☐ Automatically resolve dependency conflicts

After downloading all packages, it starts to installing them in your local machine. It took about 20 minutes in my machine… When the installation in the local host is finished, the following image is presented:



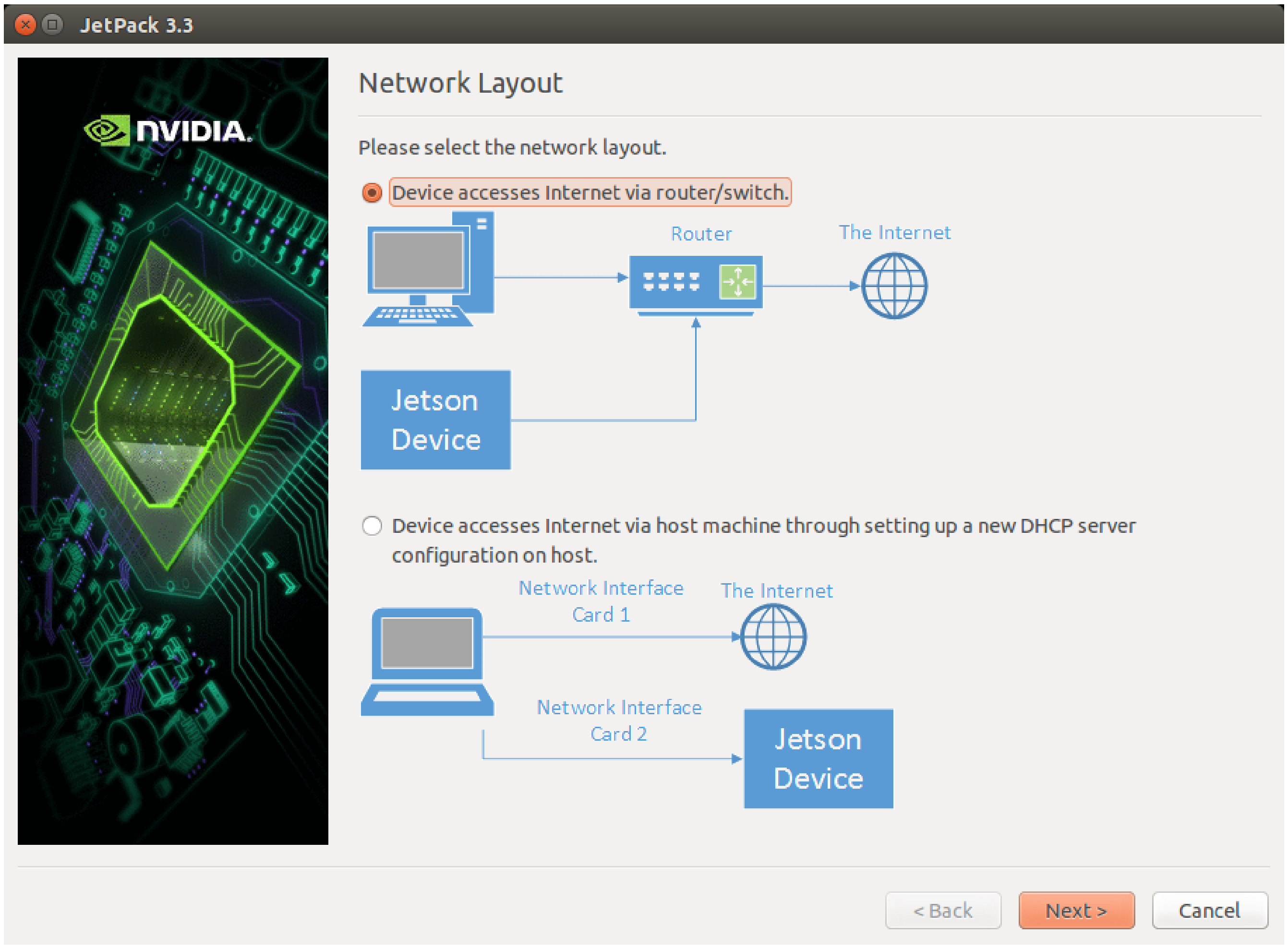
Post Installation Jetson TX2/TX2i

Following actions will be performed at this stage.

- Flash 64 Bit OS to TX2 device
- Push and install 64Bit CUDA on target
- Push and install TensorRT on target
- Push and install 64Bit cuDNN on target
- Cross-compile 64Bit CUDA samples and push to target
- Push and install 64Bit OpenCV on target
- Push and install MMAPI on target
- Push and install 64Bit VisionWorks on target
- Push and install 64Bit VisionWorks SFM on target
- Push and install 64Bit VisionWorks Tracking on target

Flashing the Jetson board

Next step is to configure how the binaries are transmited to the Jetson board. Thus, JetPack asks what is the layout of the network to transmit the data. Select **Device accesses internet via houter/switch.** and click `Next`, as the image:


JetPack 3.3

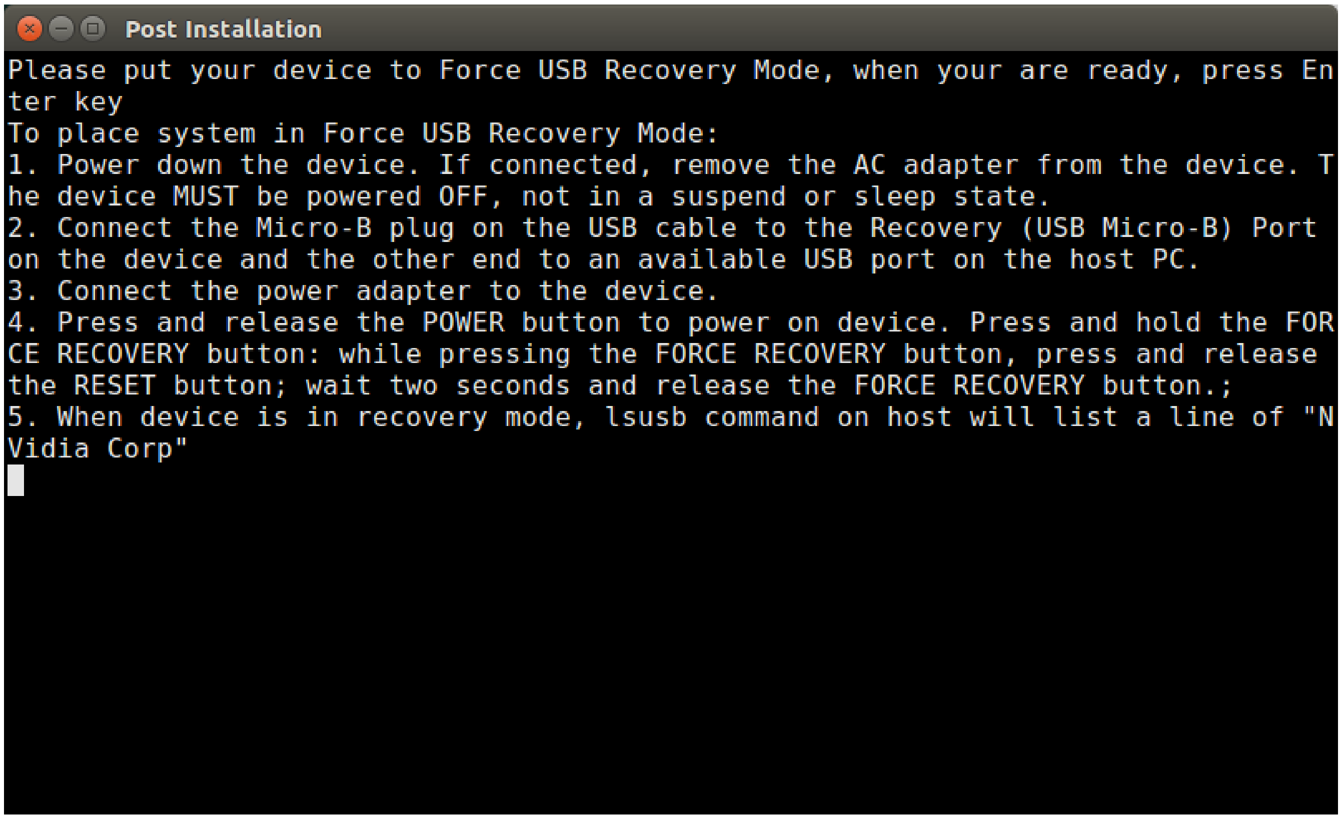
Network Layout

Please select the network layout.

[✓ Device accesses Internet via router/switch.]

Device accesses Internet via host machine through setting up a new DHCP server configuration on host.

JetPack then asks you to put the Jetson board in the *Recovery Mode*, by powering down the device (in case Jetson is on), connecting the micro-USB cable in the recovery port and in the USB of the local host, pressing and holding the *REC* button while turning the board on, as the image:

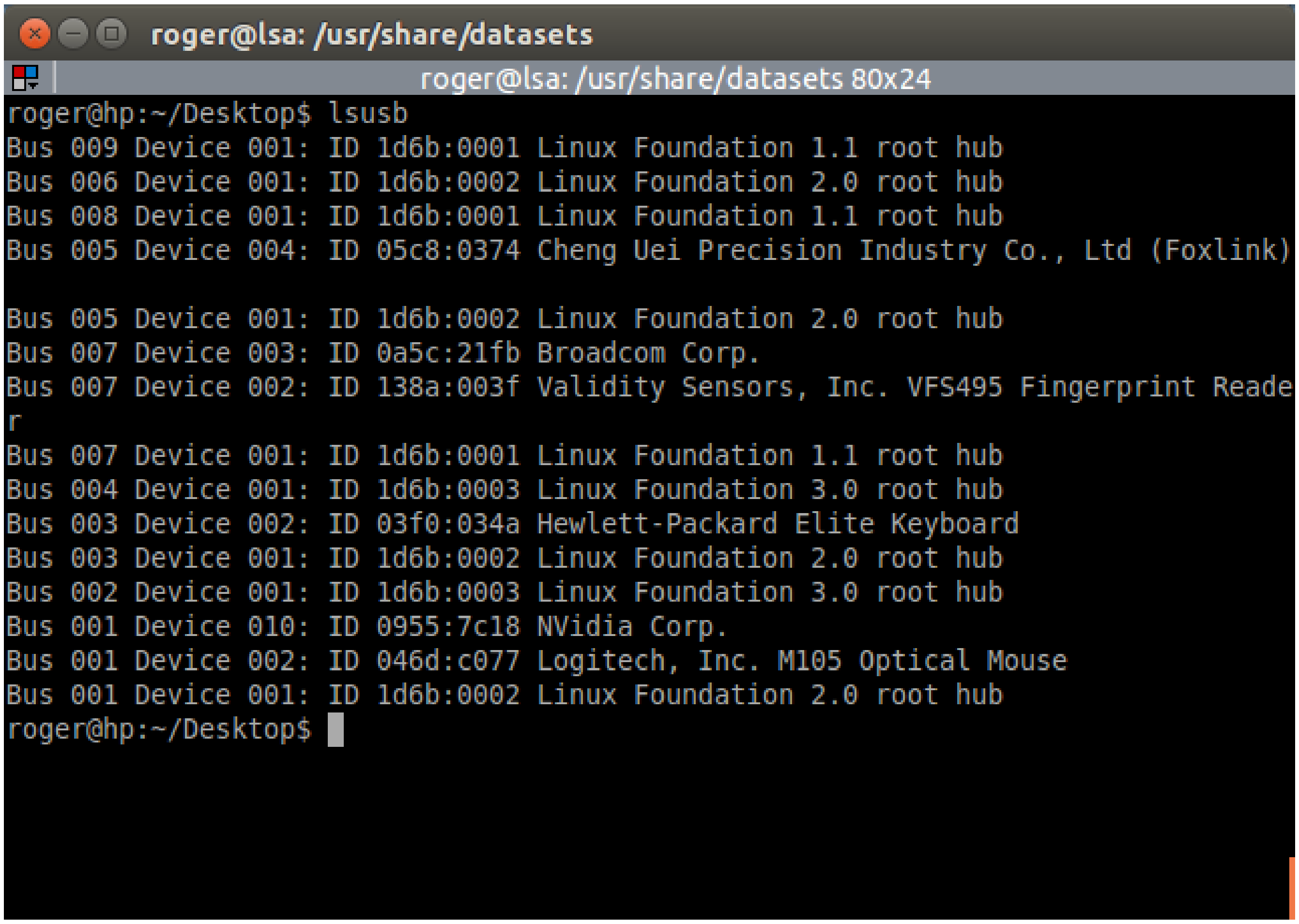


```
Please put your device to Force USB Recovery Mode, when your are ready, press En
ter key
To place system in Force USB Recovery Mode:
1. Power down the device. If connected, remove the AC adapter from the device. T
he device MUST be powered OFF, not in a suspend or sleep state.
2. Connect the Micro-B plug on the USB cable to the Recovery (USB Micro-B) Port
on the device and the other end to an available USB port on the host PC.
3. Connect the power adapter to the device.
4. Press and release the POWER button to power on device. Press and hold the FOR
CE RECOVERY button: while pressing the FORCE RECOVERY button, press and release
the RESET button; wait two seconds and release the FORCE RECOVERY button.;
5. When device is in recovery mode, lsusb command on host will list a line of "N
Vidia Corp"
```

In order to check if the Jetson board is set in recovery mode, open the terminal in the local host and type:

```
$ lsusb
```

If the board is in revery mode, you should see the Jetson listed as NVidia (*ID 0955:7c18 NVidia Corp.*) in the output, as the image:


roger@lsa: /usr/share/datasets

roger@lsa: /usr/share/datasets 80x24

```
roger@hp:~/Desktop$ lsusb
Bus 009 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 004: ID 05c8:0374 Cheng Uei Precision Industry Co., Ltd (Foxlink)

Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 003: ID 0a5c:21fb Broadcom Corp.
Bus 007 Device 002: ID 138a:003f Validity Sensors, Inc. VFS495 Fingerprint Reade
r
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 002: ID 03f0:034a Hewlett-Packard Elite Keyboard
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 010: ID 0955:7c18 NVidia Corp.
Bus 001 Device 002: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
roger@hp:~/Desktop$
```

If you don't see the Jetson using `lsusb`, then the device will not be flashed. In case the Jetson appears in `lsusb`, return to the screen with the instructions and press `Enter` to start flashing the OS image in the board.

References

Robot Operating System (ROS) on Jetson TX2

Robot Operating System (ROS) is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server[1].

The primary goal of ROS is to support code reuse in robotics research and development so you can find a built-in package system. It is interesting to note that although ROS contains Operation System in the name, keep in mind that ROS is not an OS, a library, or an RTOS, but a framework using the concept of an OS. A good introduction is given in the freely available book named A Gentle Introduction to ROS by Jason O'Kane. The ROS Wiki also contains lots of tutorials to introduce you to its main concepts.

[1] ROS Introduction

Installing ROS in Jetson TX2

JetsonPack for Jetson TX2 is based on Ubuntu 16.04 and thus, the ROS version to install is named *Kinetic*. In order to install the ROS package, you can run the `installROS.sh` file as:

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/jetson/TX2/scripts/installROS.sh
$ chmod +x installROS.sh
$ ./installROS.sh
```

This script installs ROS Kinetic Desktop and the Point Cloud Library.

Creating Catkin workspace to run ROS

Catkin packages can be built as a standalone project, in the same way that normal cmake projects can be built, but catkin also provides the concept of workspaces, where you can build multiple, interdependent packages together all at once. A catkin workspace is a folder where you modify, build, and install catkin packages[2]. In order to create a Catkin workspace, you should download and run the file `createCatkin.sh` as:

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/jetson/scripts/createCatkin.sh
$ chmod +x createCatkin.sh
$ ./createCatkin.sh
```

This script creates a structure in the home folder with the root workspace set in `/home/ubuntu/catkin_ws`.

Installing vision_opencv in ROS

By default, ROS has support to native OpenCV. However, when trying to compile a C++ code with a call to `cv_bridge`, the following error

Error: make[2]: *** No rule to make target '/usr/lib/arm-linux-gnueabihf/libopencv_videostab.so.2.4.8'. Stop.

In order to get rid of the error, we have to compile vision_opencv in our Catkin workspace. Hence, the first step is to clone the Indigo version of `vision_opencv` into the `catkin_ws/src` folder. Then, we have to compile the workspace, by running:

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ros-perception/vision_opencv.git
$ cd vision_opencv/
$ git checkout kinetic
$ cd ../..
$ catkin_make
```

Next time you compile a C++ code that contains a call to `cv_bridge`, no errors should appear.

Testing ROS installation

To check if the ROS is installed correctly, run:

[2] Catkin Workspaces

```
$ roscore
```

And see it starts running correctly. In case of problem, you can check the log files by running:

```
$ roscd log
```

References

3.6.3 References

3.7 Boat Configuration

Here we describe the settings to perform on the boat in order to test it in the water.

3.7.1 Tablet Emulator

Some Platypus LLC boats are configured to work with two cellphones: one to control the path of the boat and the other to receive the information of the path and transmit to the eboard. Here we change the cellphone that send controls to the boat by an emulator, and thus, we need only the cellphone to receive the commands and control the boat. Here, our steps are based on the Android SDK emulator using the version 7.0 (Android Nougat) with Google API Intel Atom CPU/ABI.

Downloading and Installing the Emulator

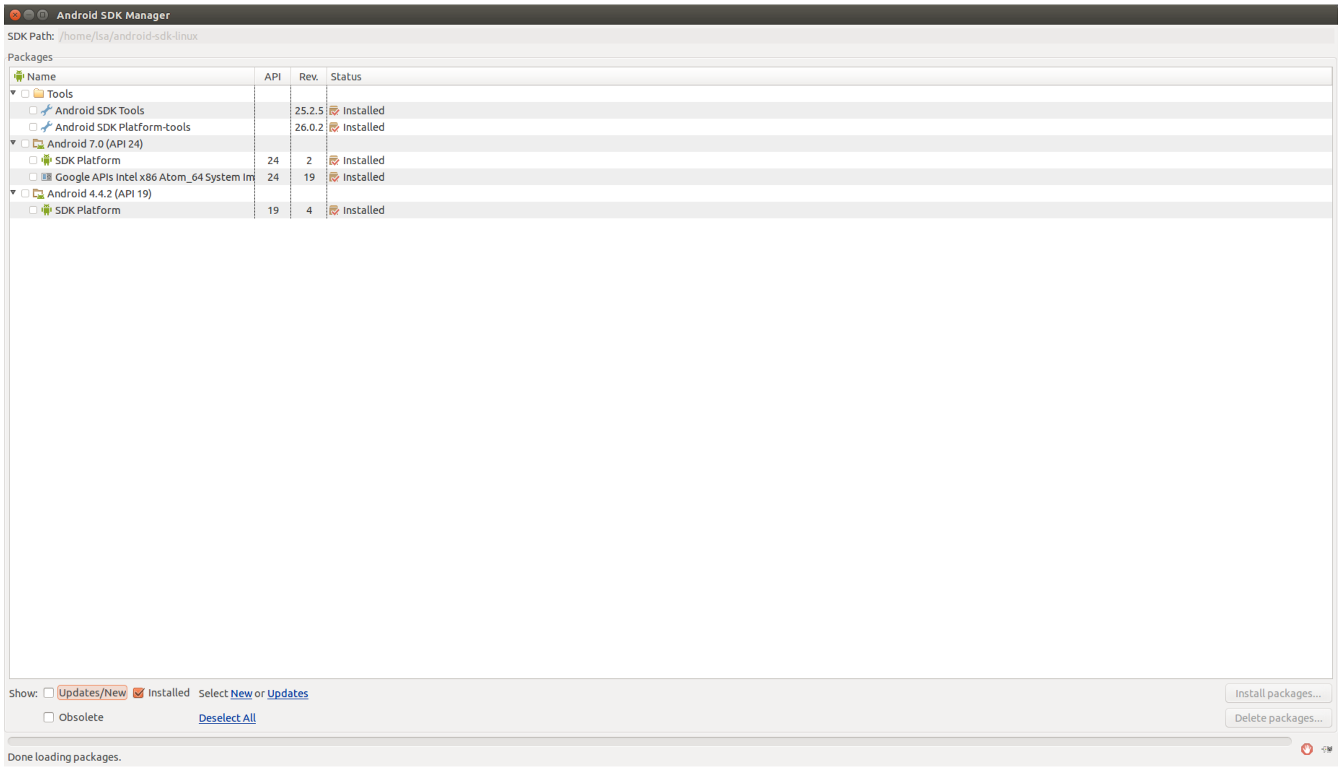
Android SDK emulator can be download directly from Android site using `wget`. After downloading, extract the content in the folder of your preference and add the paths to the `tool` folder in `.bashrc` file as:

```
$ wget http://dl.google.com/android/android-sdk_r20-linux.tgz
$ tar -zxvf android-sdk_r20-linux.tgz
$ mv android-sdk-linux /home/
$ echo "export PATH=$PATH:$HOME/android-sdk-linux/tools" >> .bashrc
$ source ~/.bashrc
```

Having the SDK installed, you should update it and download the API 24 corresponding to the Android version 7.0. Although the many options you have to install the API support, you can download only the SDK and the Google API Intel Atom CPU/ABI. In order to download the corresponding SDK you may type:

```
$ android
```

This command opens a window with options to download. Inicially, the API 24 to Android will not appear in the options, but you can update the current `Android SDK tools` and `Android SDK Platform-Tools` and restart the application up to appear in the screen the newest versions of the APIs. The resulting screen seems like the picture below:


Android SDK Manager

SDK Path: /home/lsa/android-sdk-linux

Packages

Name	API	Rev.	Status
Tools			
Android SDK Tools		25.2.5	Installed
Android SDK Platform-tools		26.0.2	Installed
Android 7.0 (API 24)			
SDK Platform	24	2	Installed
Google APIs Intel x86 Atom_64 System Im	24	19	Installed
Android 4.4.2 (API 19)			
SDK Platform	19	4	Installed

Show: ☐ Updates/New ☑ Installed Select New or Updates

☐ Obsolete Deselect All

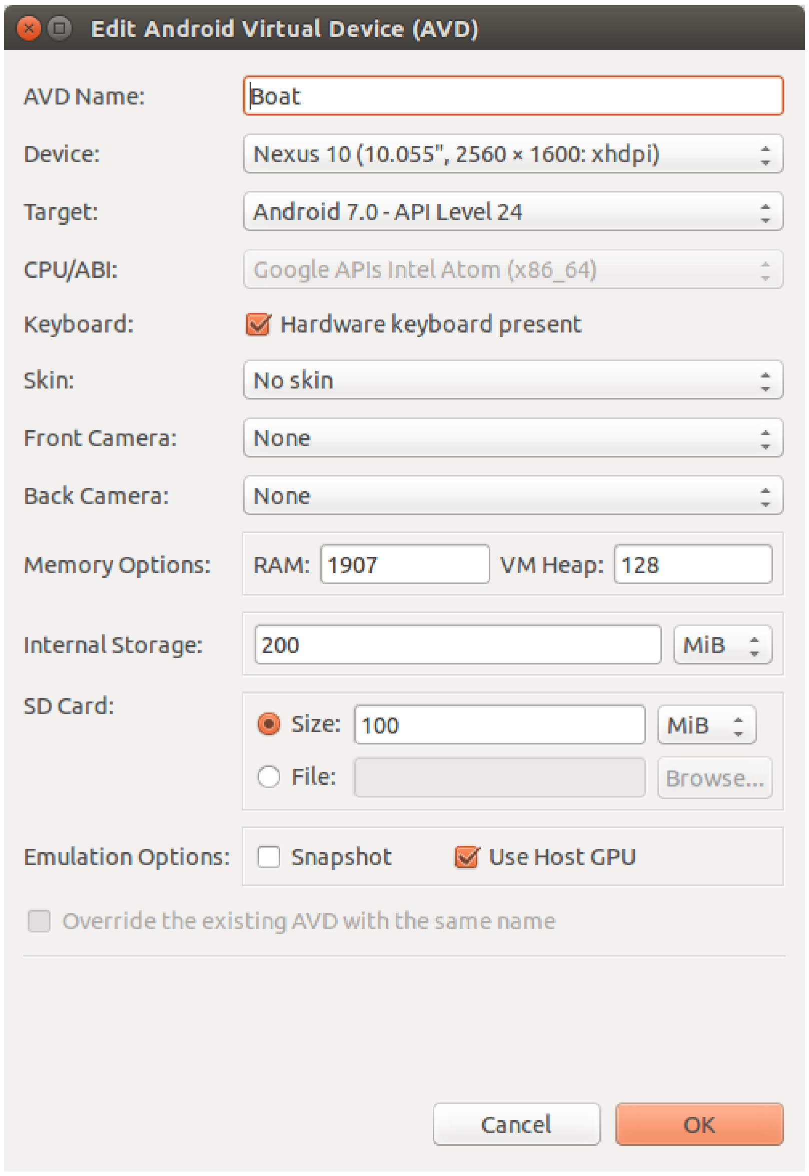
Install packages... Delete packages...

Done loading packages.

After installing the SDK, you have to create the Android Virtual Device (AVD). You can do that by typing:

```
$ android avd
```

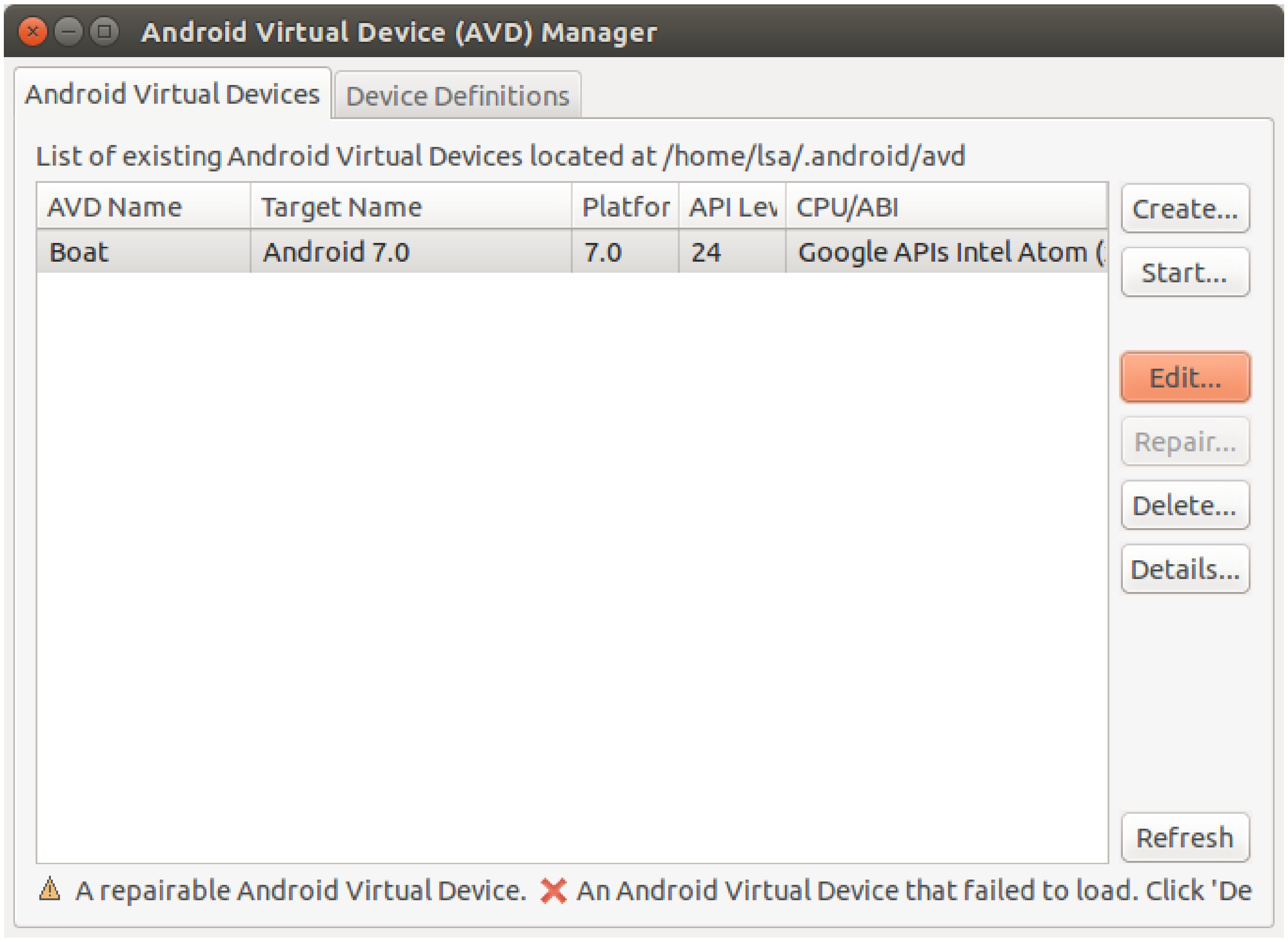
And a window to create AVDs appears. On this window, select `Create...` and a screen to create an AVD will appear. On this screen, create an AVD name and select `Nexus 10` as device, `Android 7.0 - API Level 24` as Target, `Google API Intel Atom` as CPU/ABI, `No skin` in Skin, set a value to SD Card and finally check the `Use Host GPU` in emulation options in order to increase the processing speed. Following these steps you have a screen filled as follows:



AVD Name:	Boat
Device:	Nexus 10 (10.055", 2560 × 1600: xhdpi)
Target:	Android 7.0 - API Level 24
CPU/ABI:	Google APIs Intel Atom (x86_64)
Keyboard:	✓ Hardware keyboard present
Skin:	No skin
Front Camera:	None
Back Camera:	None
Memory Options:	RAM: 1907 VM Heap: 128
Internal Storage:	200 MiB
SD Card:	● Size: 100 MiB ○ File: Browse...
Emulation Options:	☐ Snapshot ✓ Use Host GPU

☐ Override the existing AVD with the same name

Clicking in `OK` will create a AVD in your hard drive that you can access next time as the picture shows:

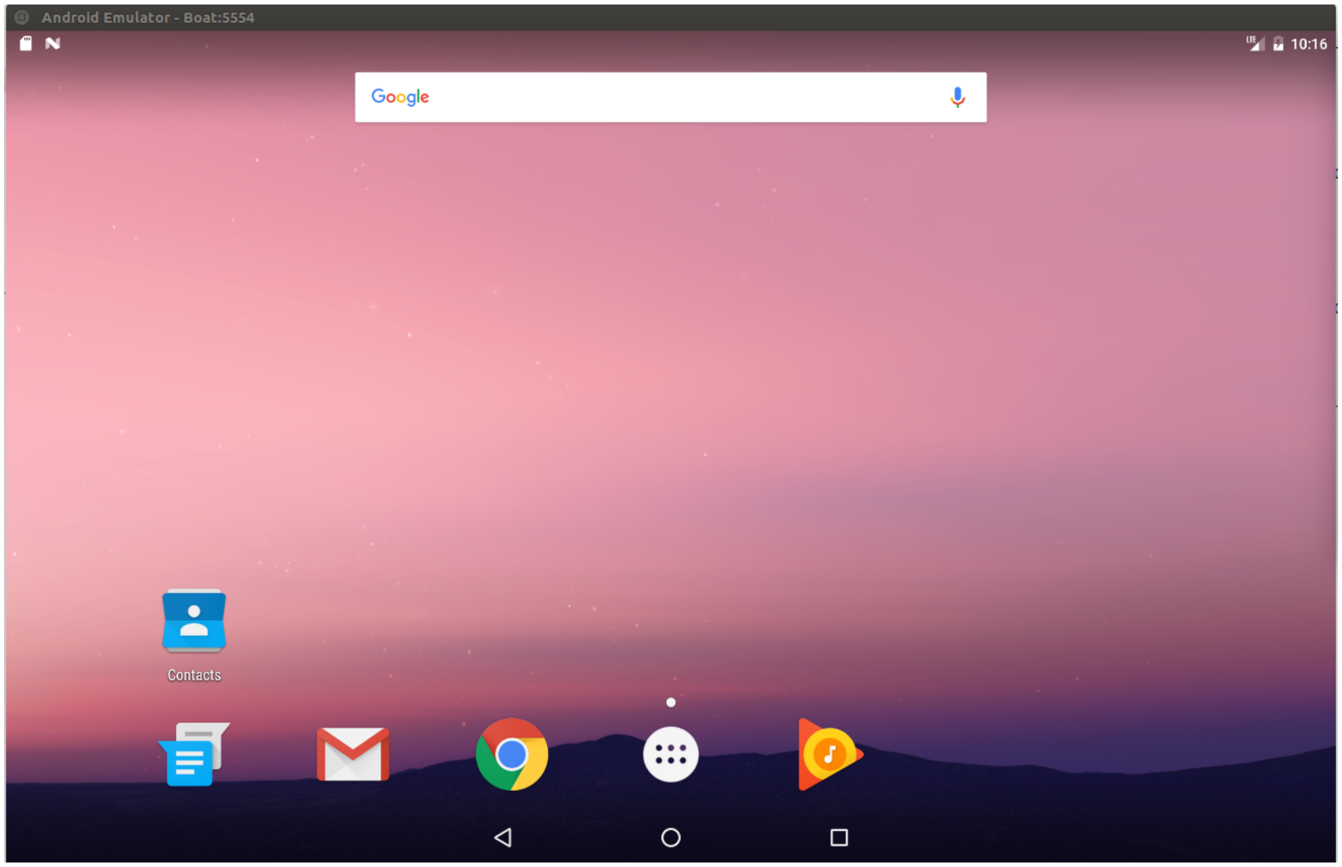


List of existing Android Virtual Devices located at /home/lsa/.android/avd

AVD Name	Target Name	Platfor	API Lev	CPU/ABI
Boat	Android 7.0	7.0	24	Google APIs Intel Atom (

⚠ A repairable Android Virtual Device. ✗ An Android Virtual Device that failed to load. Click 'De

If everything goes right, you can select your AVD and `Start...`. Android emulator will start and finally a screen will appear like:



Error: When loading the AVD (Boat) may appear an error `sh: 1: glxinfo: not found`.

In case the `sh: 1: glxinfo: not found` error appear, you have to install `mesa-utils` via `apt-get` by using:

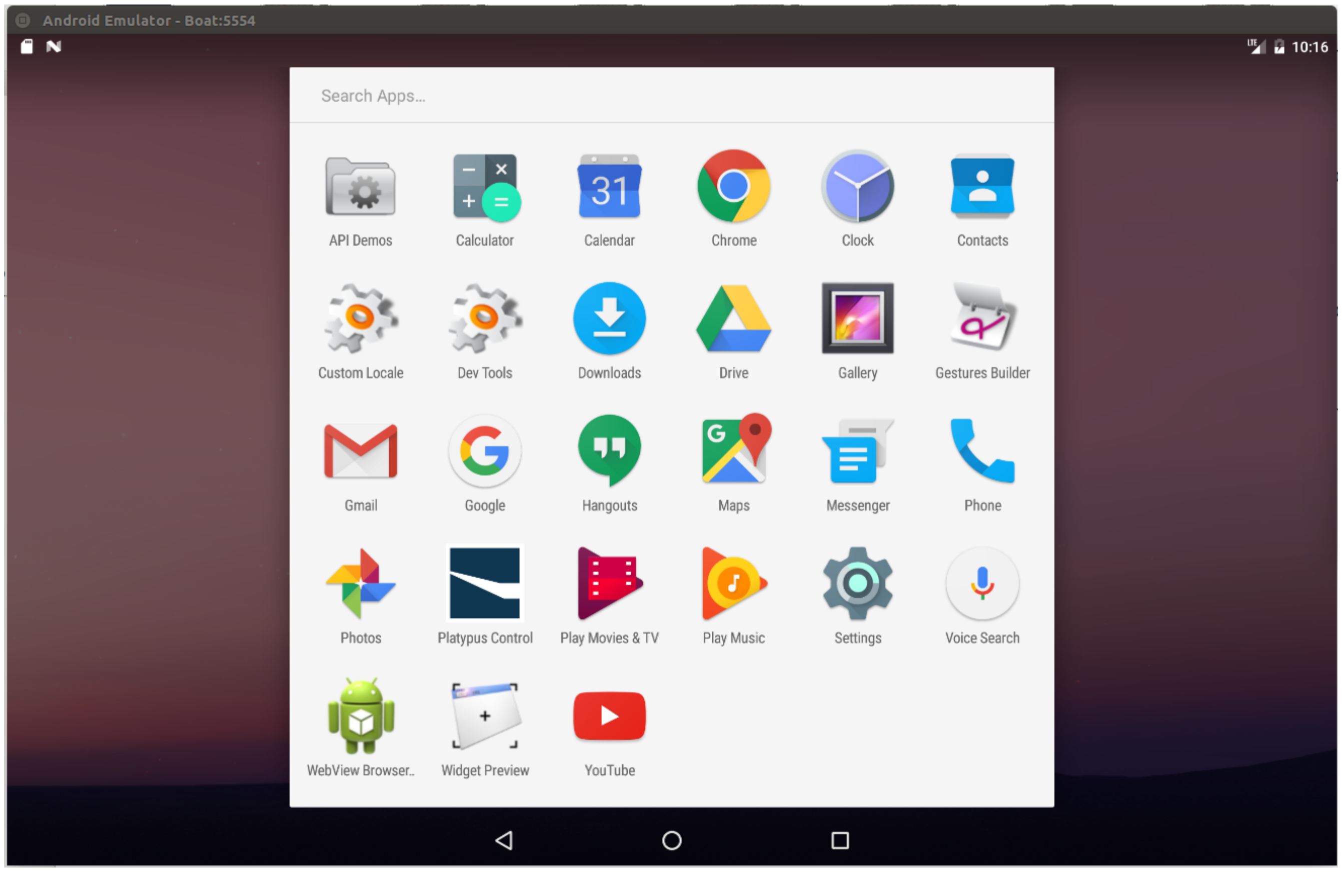
```
$ sudo apt-get install mesa-utils
```

Installing and Configuring Platypus Control

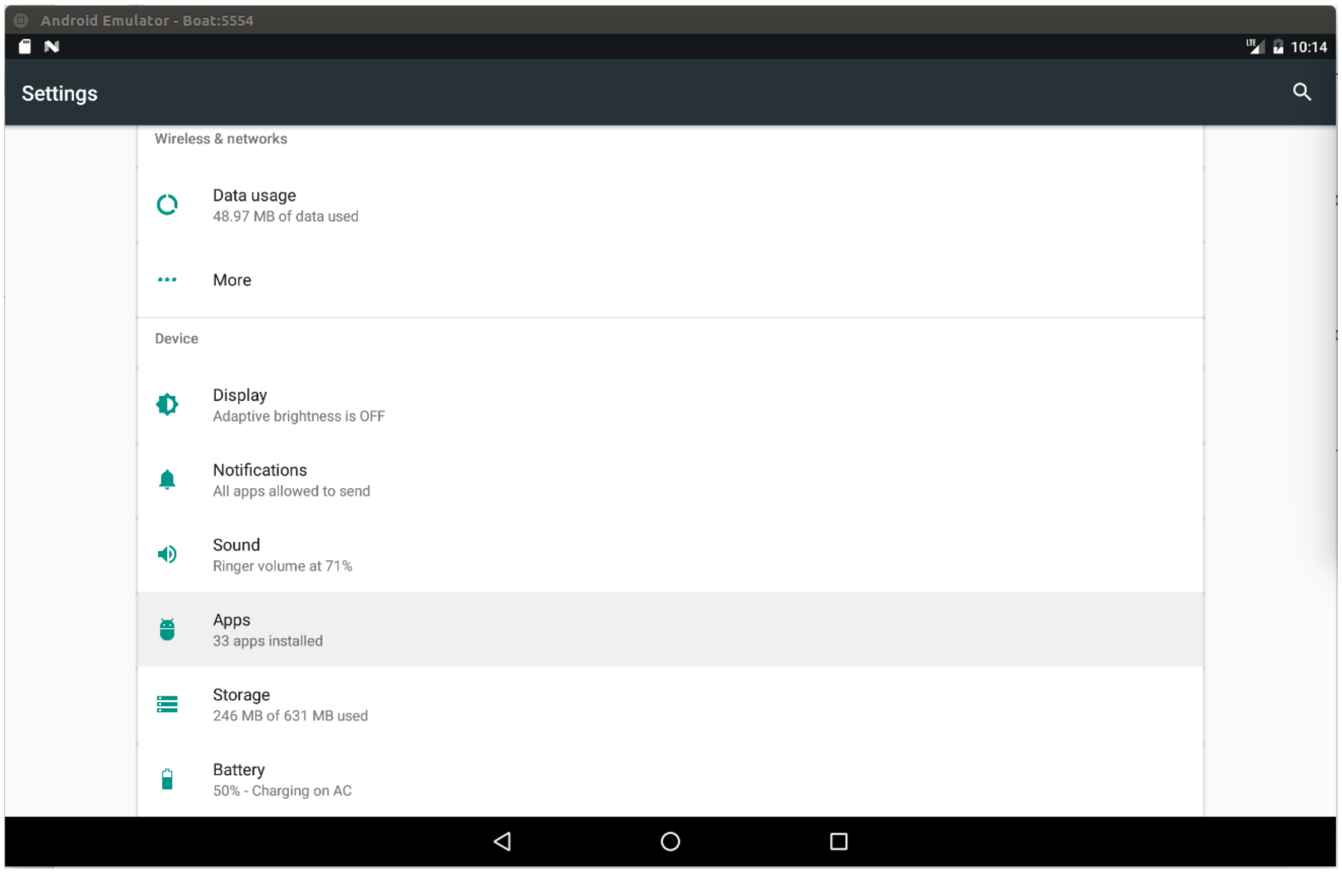
Platypus Control is the app responsible to send the commands to the boat indicating the paths it should follow, as well as controlling the boat with a joystick. The code of this app can be download and compiled from Github official site using Android SDK, but in order to simplify this process, you can download the apk direclty from our repository using `wget` as:

```
$ wget --no-check-certificate --content-disposition https://raw.githubusercontent.com/
↪lsa-pucrs/platypus_doc/master/docs/source/boat/scripts/tablet.apk
```

Send the apk to the emulator and install it. In order to send the apk to the emulator, just drag and drop the file inside the emulator. If it was successful, the app will appear in the list of installed apps as presented in the picture:



In the next step we have to set the permissions to the app work, otherwise you cannot open the app. In order to set the permissions, go to `Settings` -> `Apps` -> `Platypus Control`. In `App Info` select `Permissions`. In the permissions window, slide to the right all options `Location`, `Phone` and `Storage`. The sequence of steps is presented in the pictures:



Android Emulator - Boat:5554

10:14

Settings

Wireless & networks

Data usage
48.97 MB of data used

More

Device

Display
Adaptive brightness is OFF

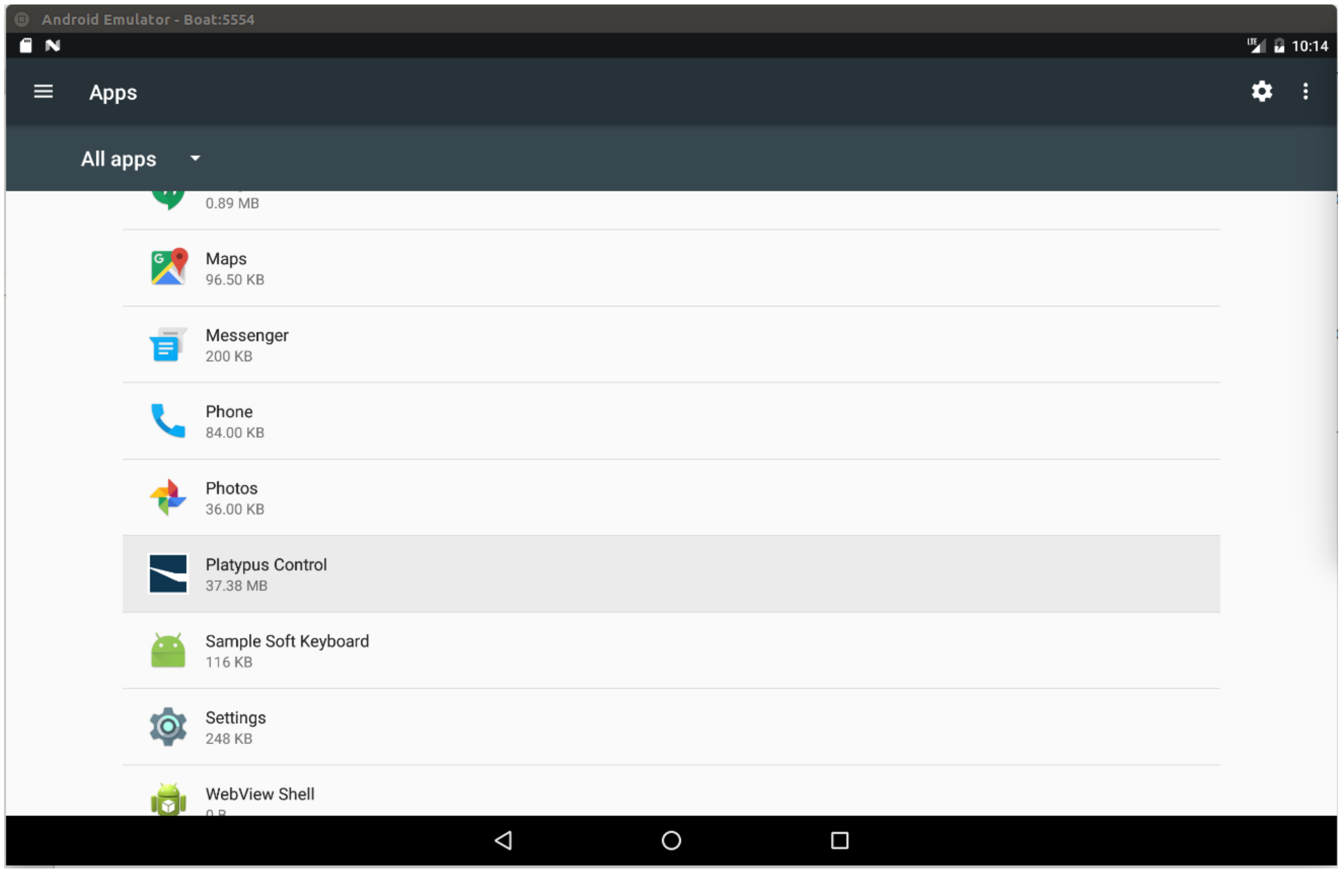
Notifications
All apps allowed to send

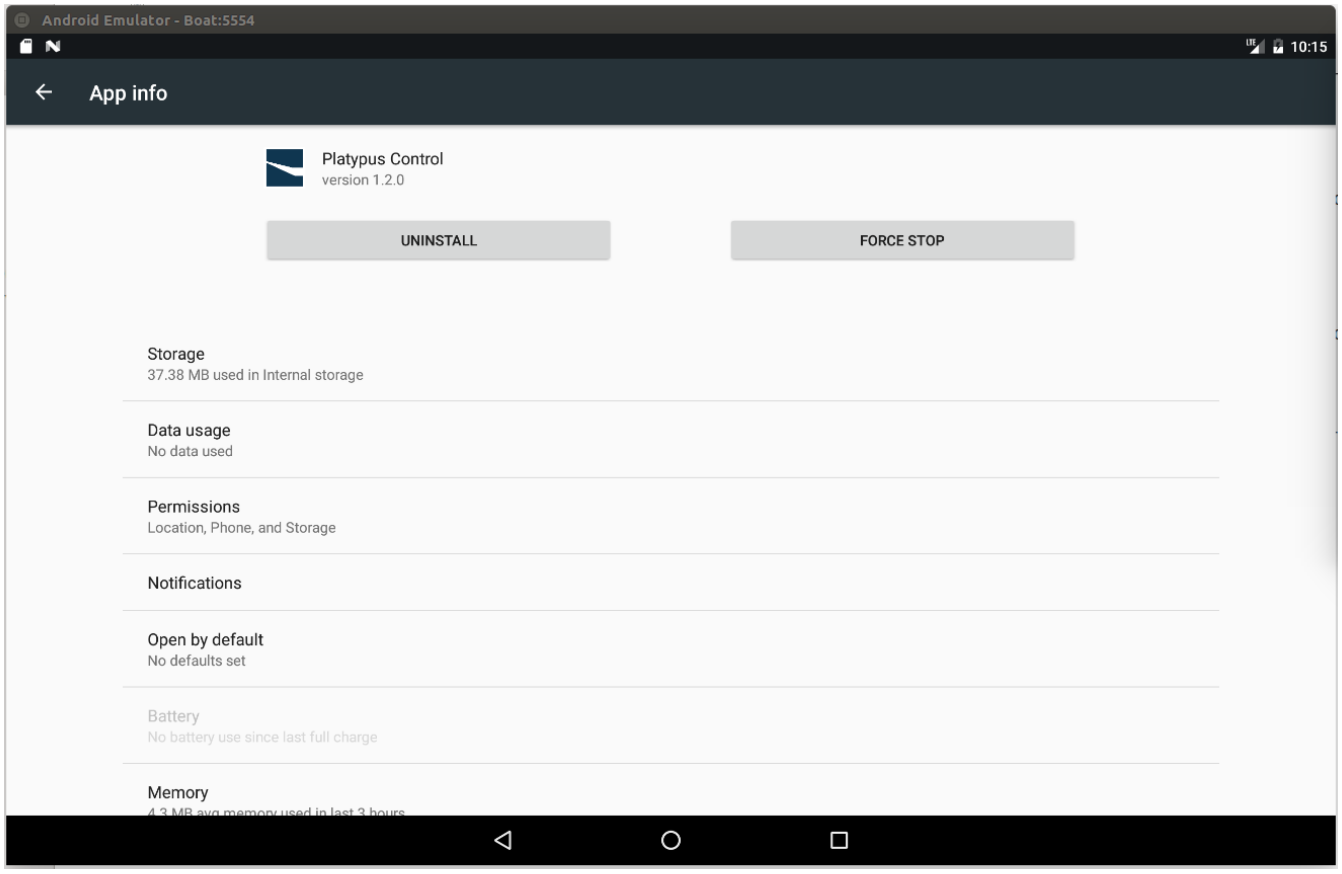
Sound
Ringer volume at 71%

Apps
33 apps installed

Storage
246 MB of 631 MB used

Battery
50% - Charging on AC


Android Emulator - Boat:5554
10:14
Apps
All apps
0.89 MB
Maps
96.50 KB
Messenger
200 KB
Phone
84.00 KB
Photos
36.00 KB
Platypus Control
37.38 MB
Sample Soft Keyboard
116 KB
Settings
248 KB
WebView Shell



Platypus Control
version 1.2.0

Storage
37.38 MB used in Internal storage

Data usage
No data used

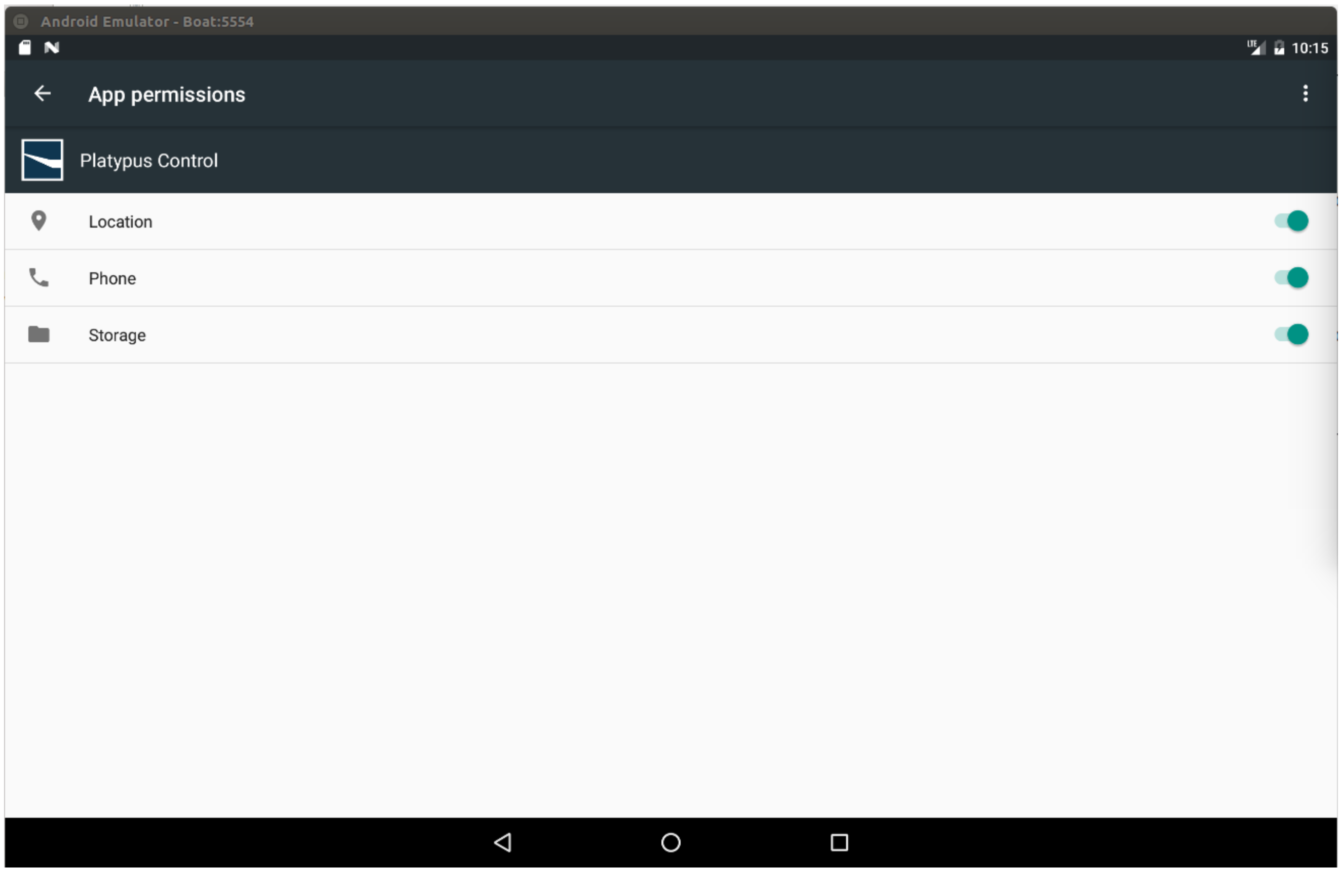
Permissions
Location, Phone, and Storage

Notifications

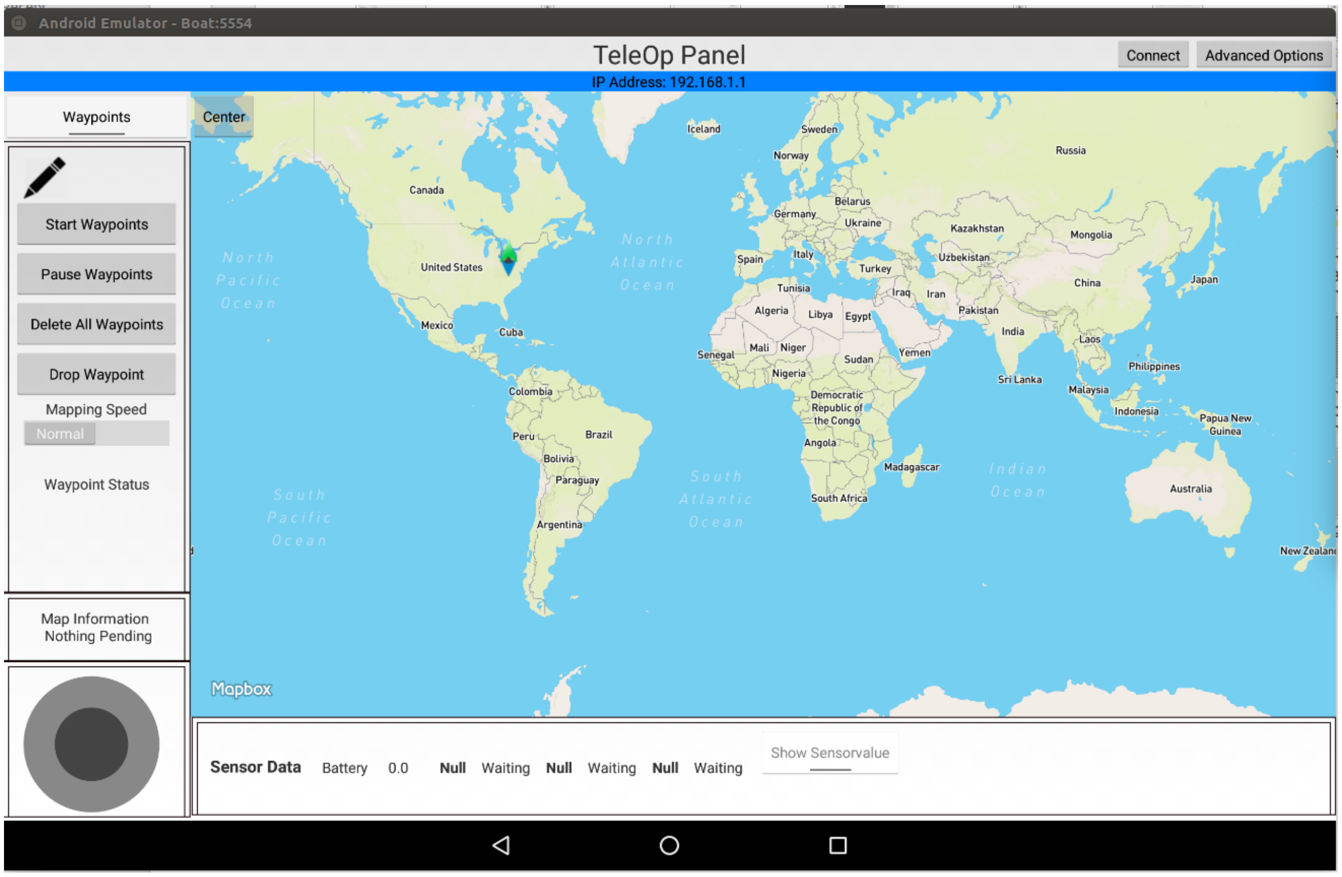
Open by default
No defaults set

Battery
No battery use since last full charge

Memory
4.3 MB avg memory used in last 3 hours



With all permissions set, you can open the Platypus Control app. If everything is fine, you should see a screen like:



3.7.2 Setting Up the eBoard

Warning: @ To be done by Roger

Hardware Description

Installing the Firmware

Testing

3.8 Preparations for the Field Test

Field testing is all about preparation, prior testing, and forseen possible failures. Failing to follow these steps will certainly cause alot of wasted time, effort, money (transportation fuel, admission fees, cost of lunch, etc).

Warning: @ To be done by Roger and Amory

3.8.1 Before Packing up for the Field Test

- make sure the goal of the test is clear for every one involved. The goal will determine which materials must be carried to the testing field, the time required for settiing up the boat, etc. Example of goals: take images of obstacles, test visual odometry, take gps and imu readings, take water samples, use the water probes, perform a bathymetry survey, etc.
- prepare a check list of materials to carry on the field test. You will waste the entire effort if, for instance, you forget to carry the laptop power adaptor, power extension cord, etc.
- setup the boat according to the mission of the field test
- test the boat on the lab and outside the lab. for instance, in the campus
- check the weather forecast one day before the planned date. Dont waste your time if there is a slightly chance of rain and wind. In addition, you will put the Lab‘s resources at risk.
- one day before, let all the baterries been carried
- you will need a NF to carry the resources out of the campus

3.8.2 Packing up for the test

- use appropriate boxes to carry the materials. cardboard boxes will probably not survive the handling and you might loose materials due to wholes on the boxes.
- separate the materials according to its costs, and/or fragility. Those materials you dont want to place them under other heavy materials.
- use your checklist
- for heavy or big materials, you could you LSA‘s cart localted in the 3rd floor of P32.
- dont forget to carry protective materials againts mosquitos, sun burn, etc. Use appropriate shoes (they might get wet), and light clothes (you are going to sweet carrying all the materials)
- check the battery of everything: laptops, eletronics, etc

3.8.3 On the test field

- first choose an appropriate place with the following caracteristics: protected from the sun, not so many people passing around, etc
- make sure you have a backup plan to rescue the boats, like a kaiak, a swimmer (use life jacket or any buoyancy device!)
- setup the power cord and the battery chargers
- check the GPS reception
- double check the batteries
- setup the wifi and any other RF communication system (antennas, signal boosters)
- setup the external access to the boat using the laptops
- test the boats on the margins
- place at the margin an drive around for about 5 min. If you are confident of it, start the planned test

Good luck !

Setting Up Wireless Communication in the Field

Warning: @ To be done by Davi

Resources Available

list here the types of antennas and signal boosters availabe at the lab.

Setting up Antenna XYZ

Setting up Antenna ABC

3.9 LSA Contributors

The list of contributors to this document.

- @Alexandre Amory
- @Roger
- @Renan
- @Marcelo
- @Davi
- @Beltrano com webpage

3.10 Boat Applications

describe here the applications developed for the boats

3.10.1 Path Coverage

describe here the applications developed for the boats

Warning: @ To be done by Marcelo

3.10.2 Boat Control

describe some simple control method for the boat

Warning: @ To be done by Davi

3.10.3 Beacon Based Localization

describe here the setup pf the localization method proposed by Renan

Warning: @ To be done by Renan

CHAPTER 4

Disclaimer

The purpose of this document is for the use of LSA group only, but we open it in case it can be usefull for someone else. Thus, we dont fill obliged to give any technical support, although we might help in some special situations. In addition, we are not associated with Platypus LLC.

Everything you find here is without absolutly no waranty and I'm not responsible for any inconveniences or issues that might occurs.

CHAPTER 5

Feedback

Don’t hesitate to ask about some additional info or next guides and also if you find some mistakes, please let me know. This can be done by submitting an issue or a push request on github.