**Rover ProtoSpace: Telepresence Robot**

**Muhammad Fazlur Rahman**

Robotic Intern – Protospace Toulouse

Email: [fazlur.rahman.ae@gmail.com](mailto:fazlur.rahman.ae@gmail.com)

This rover goal is to be operated as a telepresence robot, part of Glob’ibule Telepresence Robot project by Humanity Lab Airbus. The project goal is to enable children inside an isolation room in hospital to be able to see outside the room by utilizing this robot. A team is formed to work on this project, realizing the rover as a telepresence robot. The team work separated into several topics, which are physical, system, and design. The rover itself can be separated into three main parts, namely rover (lower part), interaction (upper part), user interface (UI).

This internship covers the system part specifically for the rover (lower part). In this document, several sections are provided to have the idea for the current state of the lower part, with brief discussion on the interfacing possibilities with the upper part and UI. The first section is rover design overview, providing the general idea of the rover. Secondly, hardware utilized in this rover is discussed, both for basic configuration and enhanced configuration. Third, an introduction to Robotic Operating System (ROS) is provided as it is used as the main framework for the interfacing. Afterwards, a general idea of companion computer and Arduino used in this project is presented. Lastly, brief explanation of the Graphical User Interface (GUI) is provided.

# Rover Design Overview

A rover is a vehicle designed to move across surface. The rover built in ProtoSpace, shown in Figure 1, utilizes 5 wheels, 2 of them are driving wheel. The other 3 wheels are casters, type of wheel which can turn along z-axis if it is given force on xy-plane. This rover is design by ***Gautier Aidli***, maker in ProtoSpace Toulouse.

Aluminum with 10 mm thickness is chosen as the frame material as it is strong and heavy. As this is a first prototype, a strong frame is required to prevent the rover from being broken. For the heaviness, it is required be heavy as the wheel commonly have a load of a person. Less weight could result to less friction force which can cause slip upon the wheel.

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure 1. Rover Overview

To control the rover, on a basic mode, remote control and Pixhawk is used. This hardware is discussed later in this document. The main consideration to use Pixhawk is the availability of rover firmware and its firmware safety. Thus, this should be a robust proposition. The other benefit is Pixhawk ability to be interfaced with a lot of modules, to control or to be controlled.

# Hardware

This section explains several basic hardware used in this rover, in which if combined can have basic functions of the rover. The basic functions include controlling the rover by using remote control and connecting it to ground control station (GCS) via mission planner. There are 4 basics hardware, namely Pixhawk, motor and ESC, power system, and remote control. Figure 2 shows the wiring diagram of basic hardware. The enhanced configuration is also discussed in this section, providing information on advanced hardware implementation for the rover.

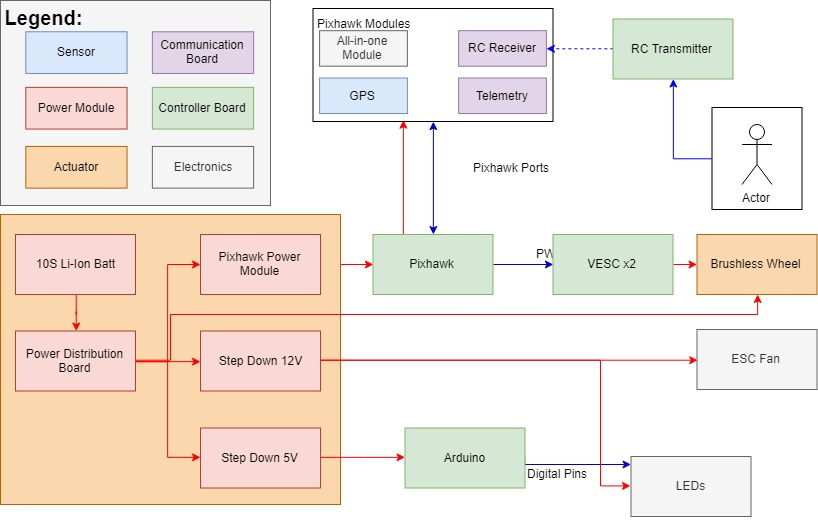


Figure . Wiring Diagram, Basic Hardware

## Pixhawk

Pixhawk, as shown in Figure 2, is a dedicated board to control unmanned vehicle such as fixed wing, multirotor, rover, et cetera. It consists of several sensors including 3 axis accelerometers, 3 axis gyroscope, 3 axis magnetometers, and a barometer. An additional sensor can be added to enhance the navigation, such as GPS and rangefinder. With these sensors, one can obtain rover’s orientations, altitude, as well as coordinate (if GPS is included).



Figure . Pixhawk Pro 3

On a basic rover, Pixhawk is utilized to handle the motor and ESC movement by taking input from remote control. Pixhawk can also control system’s safety such as activating failsafe when an unwanted event occurs. There are also other parameters that can be configured from Pixhawk firmware, using the GCS.

The firmware used in this project is ArduRover V3.4.2. The saved parameters can be found in <https://github.com/RoverProtospace>. By utilizing this firmware, the rover can maneuver in several movements, shown in figure XXX. The first maneuver is pure translation where the rover moves forward and backward. It happens when rover only receives command in from the throttle and the steering command is in trim position. The second one is pure rotation, where the rover spins around the center point between the two wheels.

|  |  |  |
| --- | --- | --- |
| 1. Translation | 1. Rotation | 1. Turning |

Figure . Rover Maneuver

## Wheel and ESC

The motor used in this project is a wheel integrated motor, with rated power of 350W. The operation voltage is between 24V, 36V, or 48V. The wheel has a Hall effect sensor, its diameter is 200mm, and rim size of 8 inches. The speed, based on *banggood* website, is around 12-28 km/h and the wheel should operate under load between 80-150 kg. The wheel is shown in Figure 4.

|  |  |
| --- | --- |
| Figure . Wheel with Hall Sensor | Figure . VESC |

The ESC used in this project is a version of 50A Brushless 4.12 VESC Controller, shown in Figure 4. One of the benefits of this ESC is that the firmware is programmable through a software called VESC Tool. The ESC can have several modes, including field-oriented control (FOC), which enable the brushless motor to be controlled in a high performance such as full torque at zero speed and high dynamic including fast acceleration and deceleration. Also, it can be used to reverse the wheel spinning direction or to have a brake. Unfortunately, there’s no firmware which allows direction reverse and brake at the same firmware.

## Power System

This rover uses 10S lithium ion battery (*li-ion*), manufactured by LALALA, shown in figure XXX. The recommended voltage for the battery to be used is between 35V to 42V. Lower than that, the battery should be recharged. The other feature of the battery is the output current which can go up to LALA amperes.

|  |  |
| --- | --- |
| Figure . 5V Step Down Module | Figure . 12V Step Down Module |

A power distribution board is used to supply power to several components such as Pixhawk power module, 5V step down module, and 12V step down module, as shown in Figure 1. The 5V step-down module, shown in Figure 1, has an input voltage between 6-50V and supply current up to 20A. It can supply a variety of output, ranging from 5V to 9V. The other step-down module, ***LM2596***, has input voltage up to 40V and can supply a variable output voltage up to 37 volts and output load current of 3A. In this case, it is used to supply 12V.

## Remote Control

Remote control used in this thesis project is Futaba T8J which has 8 output channels. The main channel which is 1 to 4 is allocated to control roll, pitch, throttle, and yaw respectively. The remaining channels are allocated to control flight mode, avoidance availability, and to check pitch servo.

This remote-control operates at 2.4 GHz which is a general frequency band for remote control. There are two available output of this remote-control receiver or airside, which are PWM and S-Bus. For this research, PWM is used since Arduino can easily read this signal. The connected signal to Arduino will be sent out again after it is processed and with the other channel are bypassed via a PPM encoder before it is connected to Pixhawk.

## Enhanced Configuration

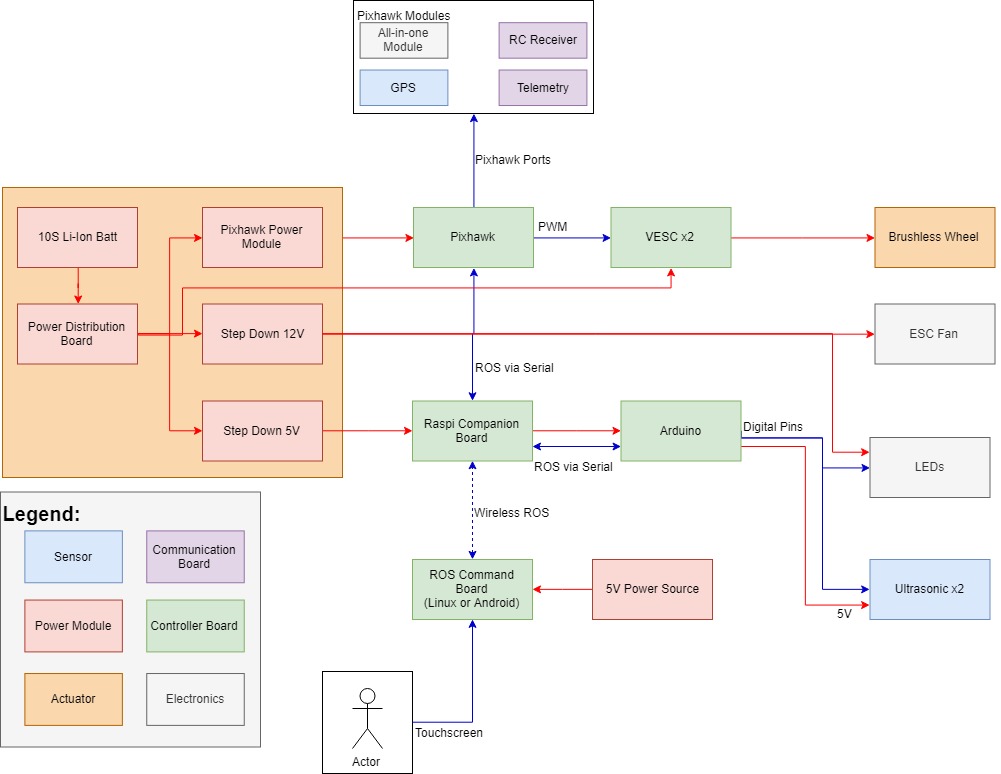


Figure . Wiring Diagram, ROS Integration

# Robotic Operating System (ROS) Framework

ROS is an open-source, meta-operating system which provides services including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [18]. ROS is not a hard-real-time framework, since it has no guarantee of missing a deadline. It is a language independent framework as one can implement either Python or C++. Understanding ROS itself is challenging at the beginning since the architecture of this software is big and it provides a lot of feature. One should understand the architecture and feature before working on it.

There are three levels of concept in ROS which are file system level, Computation Graph level, and community level. The file system level covers ROS resources that one can encounter on disk, such as packages, meta-packages, package manifest, repositories, message (*msg*) types, and service (*srv*) types. One should specify type of message of each transferred data. The other level, computation graph, is the peer-to-peer network of ROS processes that are processing data together [18]. The basic concepts include nodes, Master, parameter server, messages, services, topics, and bags. All these concepts provide data to the Graph in different ways. The last level, community level, enables separate communities to exchange software and knowledge. This is common in an open source platform.

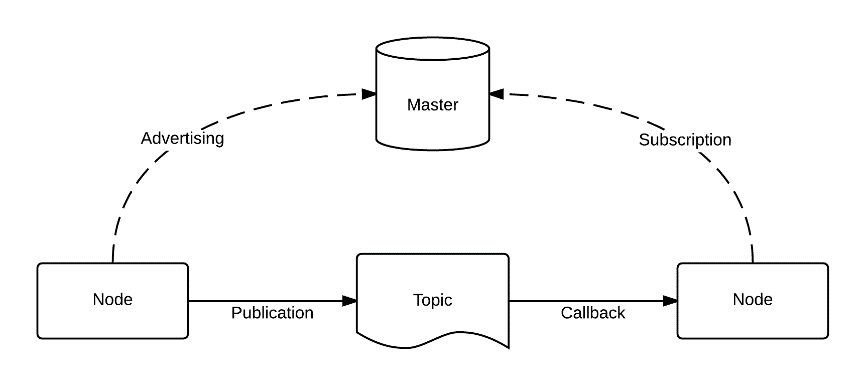


Figure 10 . ROS Basic Computation Graph Concepts [19]

For the basic operation, one needs to understand specifically the basic computation graph concepts, as shown in Figure 4. Master acts as a name service which stores topics and services registration information for the nodes. Nodes can communicate with the master to report their registration information. A node can publish one or several topics, by writing a message with a specified data type. One can access this published topic by using subscription, which can be done by a node. This subscribing node should specify the data type of the message that it is expecting.

As an example, an inertial measurement unit (IMU) provides a ROS library which starts a node called IMU. This node publishes a message *sensor\_msgs/accel* on the sensor topic. To process the data, such as making a command based on its value, one needs to write a node that subscribes to message *sensor\_msgs/accel* on the specified topic. After the subscription, one can receive the data in the other node.

The benefit of ROS is its abstraction method. One can easily change the hardware and rewrite the same message data type on the same topic. The other nodes do not need any change as it is not affected by the change. The other benefit is one can easily communicate between devices as long as they are connected in one network and have a master running.

# Companion Computer

Companion computer for this research project is used to host ROS in its operating system. Several requirements for the selection of the companion computer are Linux based operating system, supports serial communication on its pins (as an interface to Pixhawk), and its size and weight is relatively small as it will be on board.

For this research project, an Raspberry Pi is used, as shown in Figure XX. It has a powerful and energy efficient hardware and smaller in terms of size. The board supports various Linux operating system, for this project, an Ubuntu 16.04 is used as it has a stable ROS version with a lot of libraries.

## Interfacing: Companion Computer and Pixhawk

Interfacing between Pixhawk and companion computer can be done by utilizing serial communication. Connecting *Telem2* port from Pixhawk to corresponding companion computer pins is a common way in to make the two boards interfaced, as shown in Figure 7.

Port *Telem2* is chosen, instead of *Telem1* and USB, port because *Telem1* is used for data transmission between rover and ground control station. While for the USB, it is used to connect it to ground control station when the rover is on troubleshooting mode and a faster connection is needed.

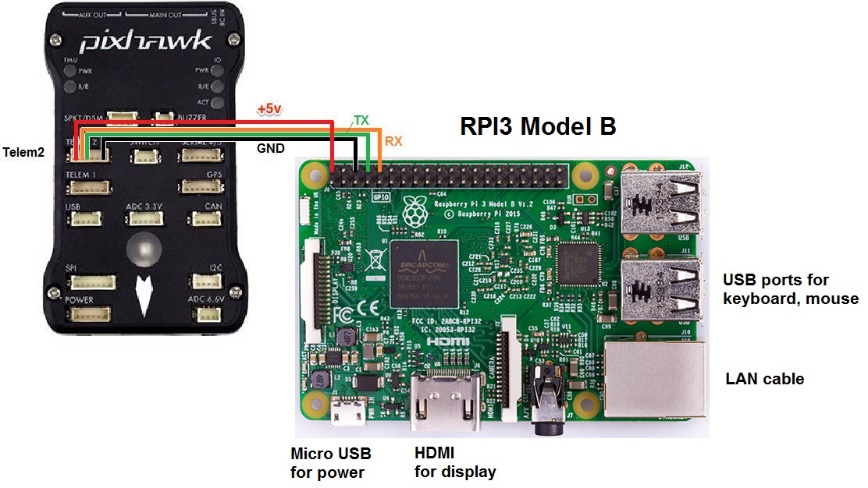


Figure 11. Pixhawk to Companion Computer Wiring

On the software side, MAVROS is used as the interface between both boards. When all setup is ready, on the terminal, a command ***roslaunch mavros px4.launch***can be executed, as shown in Figure 7. This command creates all the nodes, topics, and messages relate to data coming and from Pixhawk. The list of those data can be found using ROS list.

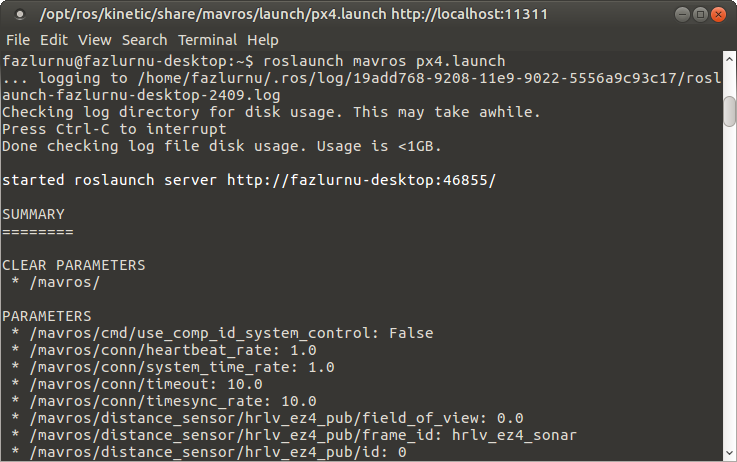


Figure 12. Launching MAVROS

Figure 12 shows an example of data sampling. Using *rostopic echo name\_of\_topic*, one can see that the Pixhawk rotates about 90 degrees in x axis. On the monitor, the printed roll value is around 90 degrees, as what we expect. Therefore, one can be sure that the data is correctly obtained. With the ability to connect to Pixhawk, one can obtain any data published by Pixhawk in the ROS topic. On the command side, Pixhawk listen, or subscribe, to several topics and it will react accordingly.

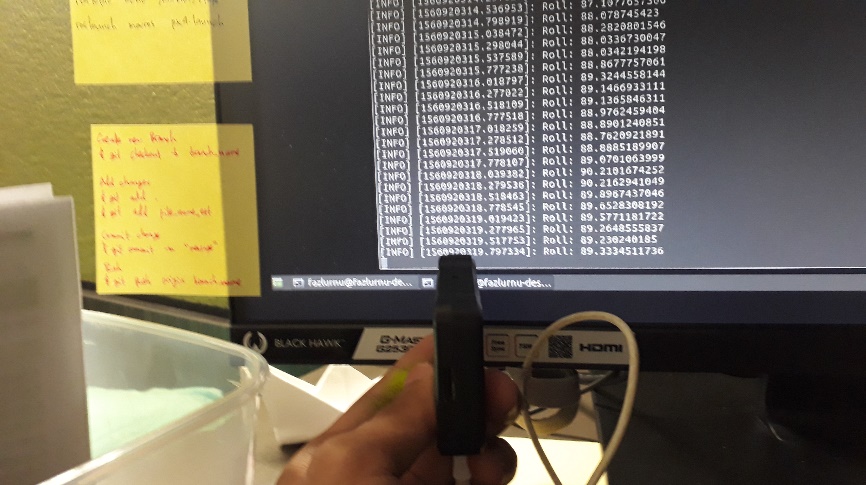


Figure 13. Data Sampling Example

**Interfacing: Companion Computer and Arduino**

## Software

# Arduino

# Graphical User Interface (GUI)

**Reference**

[18] <http://wiki.ros.org/ROS/Introduction>