**MINI PROJECT - I**
$V^{th}$ **Semester (CBGS)**
Report

# Robot development using ROS

*Submitted in partial fulfillment of*
*the requirements of the term work for subject MINI PROJECT - I*

Submitted by


| UCID | Names of Students |
|---|---|
| 2015110048 | Chirag Shah |
| 2015110037 | Srijal Poojari |

Under the guidance of
**Prof. Kumar Khandagle**

**Department of Electronics Engineering**
Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Munshi Nagar, Andheri(W), Mumbai-400058
University of Mumbai

Academic Year 2017 - 18

# CERTIFICATE

This is to certify that this is a bonafide record of the project presented by the students whose names are given below during Semister V in partial fulfilment of the requirements of the degree of Bachelor of Engineering in Electronics Engineering.

| UCID | Names of Students |
|---|---|
| 2015110048 | Chirag Shah |
| 2015110037 | Srijal Poojari |

**External Examiner**

**(signature)**

**Name:**

**Date:**

**Internal Examiner**

**(signature)**

**Name:**

**Date:**

**Seal of the Institute**

**Abstract**

This project deals with exploring the ROS framework for

- Development of a robotic system with various sensors and actuators
- To develop a quadcoptor capable of forming a 3D map of an indoor environment using a depth camera (Microsoft Kinect)

**Acknowledgement**

Acknowledgement here

Student name

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

The Kinect sensor is a horizontal bar connected to a motorized basewhich has an RGB camera, a multi-array microphone, and a depth sensor built into it. These three elements combined allow the Kinect to perform 3D motion capture.

# Chapter 2

# Literature Review

The main objective of the load emulation is to control the current drawn from the inverter to match the current, which would be drawn if it were connected to a real load. It achieves this by connecting the inverter to a power electronic $AC/AC$ converter via an appropriate interface impedance. The power electronics of the virtual load simply consists of two back to back, three-phase, six-switch, bridge converters in conventional fashion. This arrangement allows bidirectional power flow to and from the inverter. The power electronics is then controlled by the real time system (DSP) to draw/source the currents to emulate the electromechanical system on an instant by instant basis.

## 2.1 Characteristics of Real-Time Systems

As sequentially operating digital computers implement the control algorithm, it is crucial to provide appropriate computing power. The required computing speed depends on the time constants involved. power electronics systems operate in 'real time' which is a synonym of 'natural time'. Therefore, the control system must synchronize its operations to real time. The correctness of a real-time system depends not only on the logical result of the computation but also on the time at which the results are produced. Real-time systems have to respond to externally generated stimuli within a finite and specified delay. Whereas a deadline can be missed occasionally in 'soft real-time systems' such as on-line data banks, it is absolutely imperative for 'hard real-time systems' that responses occur within the specified deadline on each and every occasion [2]. This does definitely apply to power electronic control systems. Digital control systems constitute discrete-time sampled systems. With regard to power electronic systems, real-time operation typically involves control and sampling cycles in the range of 20 - 200 $\mu s$ for normal operation. However, in case of fault situations, a reaction time of less than 1 $\mu s$ might be required [3].

## 2.2 Classification of Real-Time Simulation

Testing and simulation of control algorithms is an important phase in the development of embedded control systems (ECS). Different types of simulation are possible during the design process of a controller [4] [5], ranging from simulation without time limitations, to partial real-time simulation in which only some parts of the complete control loop are simulated.

The initial functional evaluation of a control design is usually performed by off-line simulation of the control algorithm and the system. A successful evaluation leads to further tests and optimization under real-time conditions. These tests aim to improve the ability of a control design (1) to operate in real-time and (2) to interact with real equipment.
Interaction with real equipment requires a large variety of interfaces. Currently the interaction with equipment relies increasingly on complex and powerful digital interfaces replacing analog interfaces to sensors and actuators. Digital interfaces generally yield a more noise immune data transfer and facilitate additional auxiliary features such as diagnostics.
A functional control prototype is required for the validation under real-time conditions. Usually the final control hardware is not yet available at this stage. Instead, rapid prototyping methods are used to provide an early functional real-time prototype of the control system. For this prototype, the functional behavior of the control system is reproduced by an emulator. The emulator requires flexible and powerful hardware structures in order to achieve real-time operation and interaction with either a real or a simulated environment.
Real-time simulation allows comprehensive and safe tests in the laboratory if tests in the real environment are not feasible or desirable. It simulates the entire load system under normal and fault conditions. Digital simulation offers several appealing advantages over analog simulation with regard to the dynamic range of variables, flexibility and reproducibility of results for each performance etc. Digital real-time simulation is much more challenging than control system emulation because it has to operate five or ten times faster than control systems to avoid delays which may generate artificial low frequency effects. Thus, digital real-time simulation demands very high performance of its underlying hardware structure. The use of parallelism inherent in large systems is inevitable and has to be reflected in a similar parallelism in the simulator hardware.

# Chapter 3

# Project Objectives

- Learn the ROS framework
- Understand and implement SLAM algorithms for 3D mapping
- Understand the interfacing of different hardware components with ROS packages
- Combine the software and hardware components to create a stand-alone quadcoptor for 3D mapping

# Chapter 4

# Theory

# Chapter 5

# System Design

The system design can be best explained with the help of this diagram:

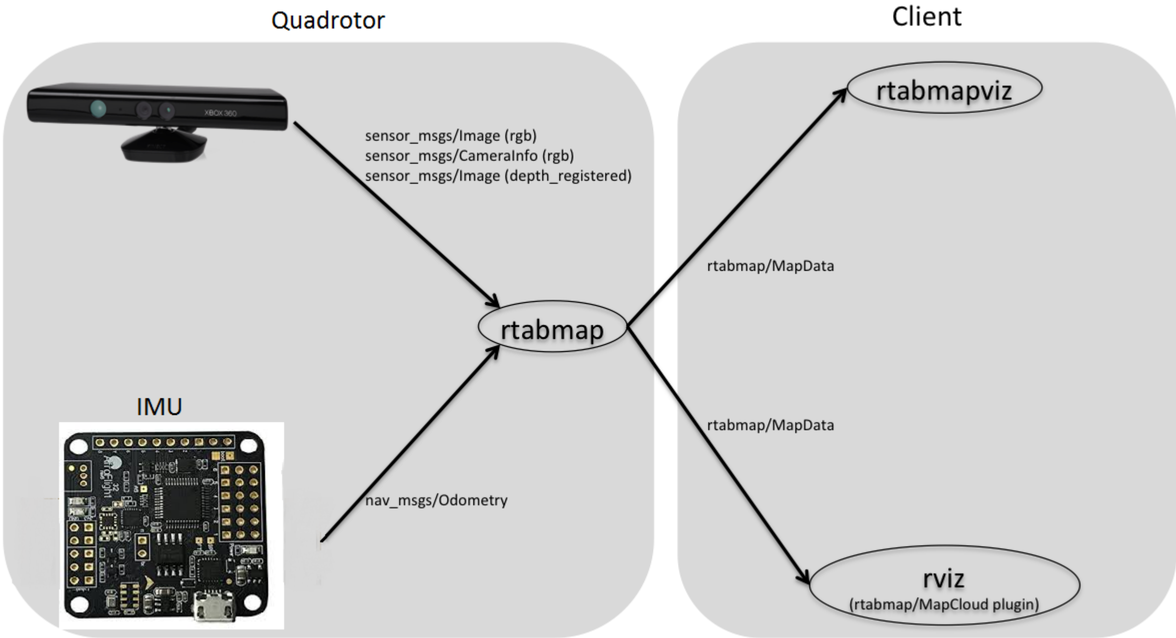
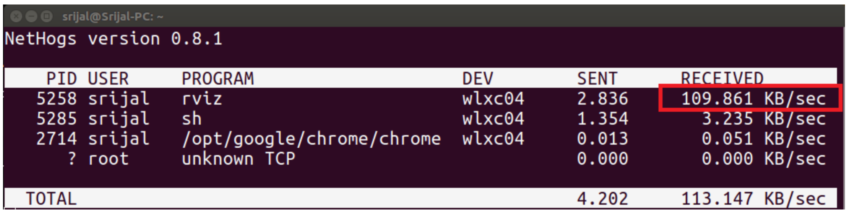



Figure 5.1: System Overview, Source: wiki.ros.org

There are two systems running ROS, the onboard computer on the quadcopter and the base station computer(client). The quadcopter takes the depth data from the Kinect and the IMU data from the Flight Controller(FC), which is required for odometry, and sends it to RTAB-Map. Using these two critical data, RTAB-Map determines position of each point in the world forms a Map Cloud. This Map Cloud is obtained by the client computer over the ROS network and visualized in Rviz.

Since this process involves many computations happening, transmitting the entire raw data would require a huge bandwidth and would not be feasible. Hence the data quality has to reduced and rate set to 5Hz to bring the bandwidth requirement to 80-120 kbps, which was measured using nethogs as seen in Figure 5.2


srijal@Srijal-PC: ~

NetHogs version 0.8.1

| PID | USER | PROGRAM | DEV | SENT | RECEIVED |
|---|---|---|---|---|---|
| 5258 | srijal | rviz | wlxc04 | 2.836 | 109.861 KB/sec |
| 5285 | srijal | sh | wlxc04 | 1.354 | 3.235 KB/sec |
| 2714 | srijal | /opt/google/chrome/chrome | wlxc04 | 0.013 | 0.051 KB/sec |
| ? | root | unknown TCP | | 0.000 | 0.000 KB/sec |
| TOTAL | | | | 4.202 | 113.147 KB/sec |

Figure 5.2: Bandwidth Requirement

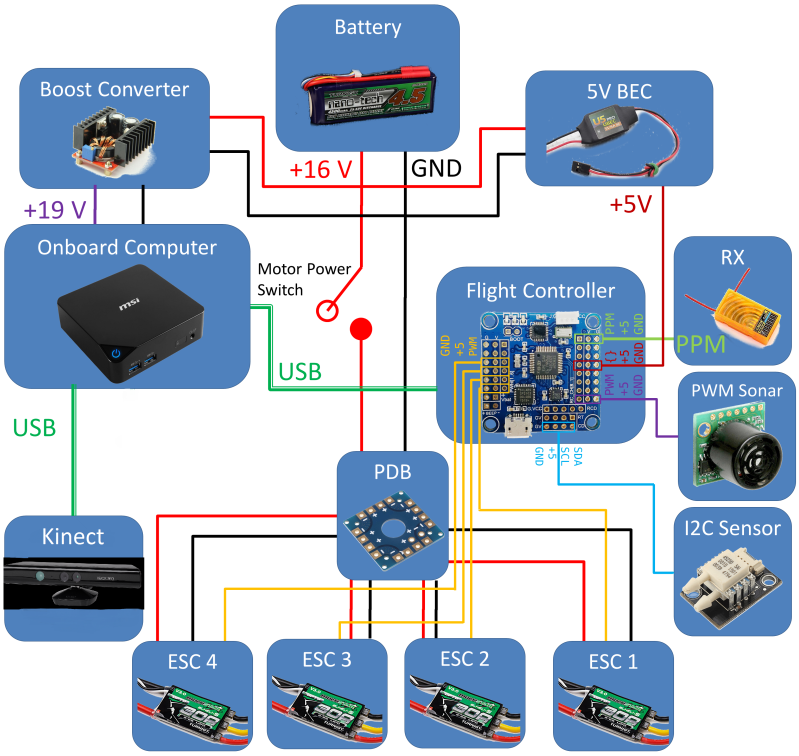



Figure 5.3: Quadcopter Hardware Layout, Source: docs.rosflight.org

Since the onboard computer's terminal is not easily accessible, ssh is used by the client computer to issue the initial commands.

# Chapter 6

# Softwares

- Ubuntu 16.04 (Xenial Xerus)
- ROS Kinetic Kame

# Chapter 7

# Results

The following images show the result of mapping using handheld Kinect:

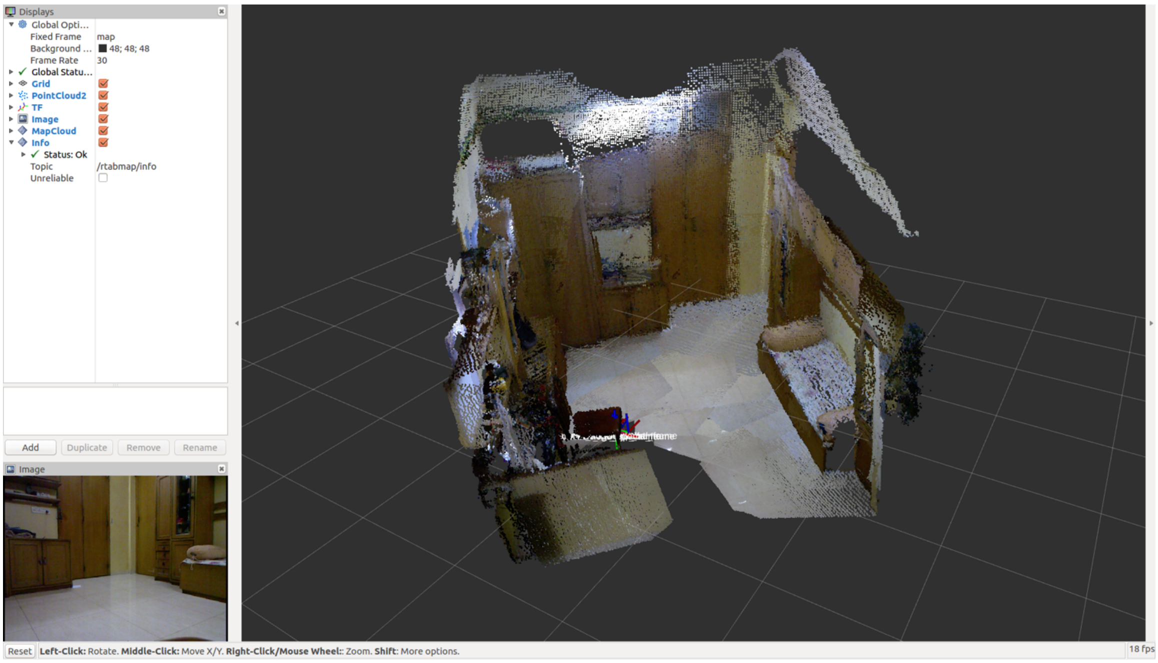



Figure 7.1: Wired mappping(PC)

The small section on the botoom left corener shows the RGB stream from the Kinect and the formed 3D map is displayed at the center.

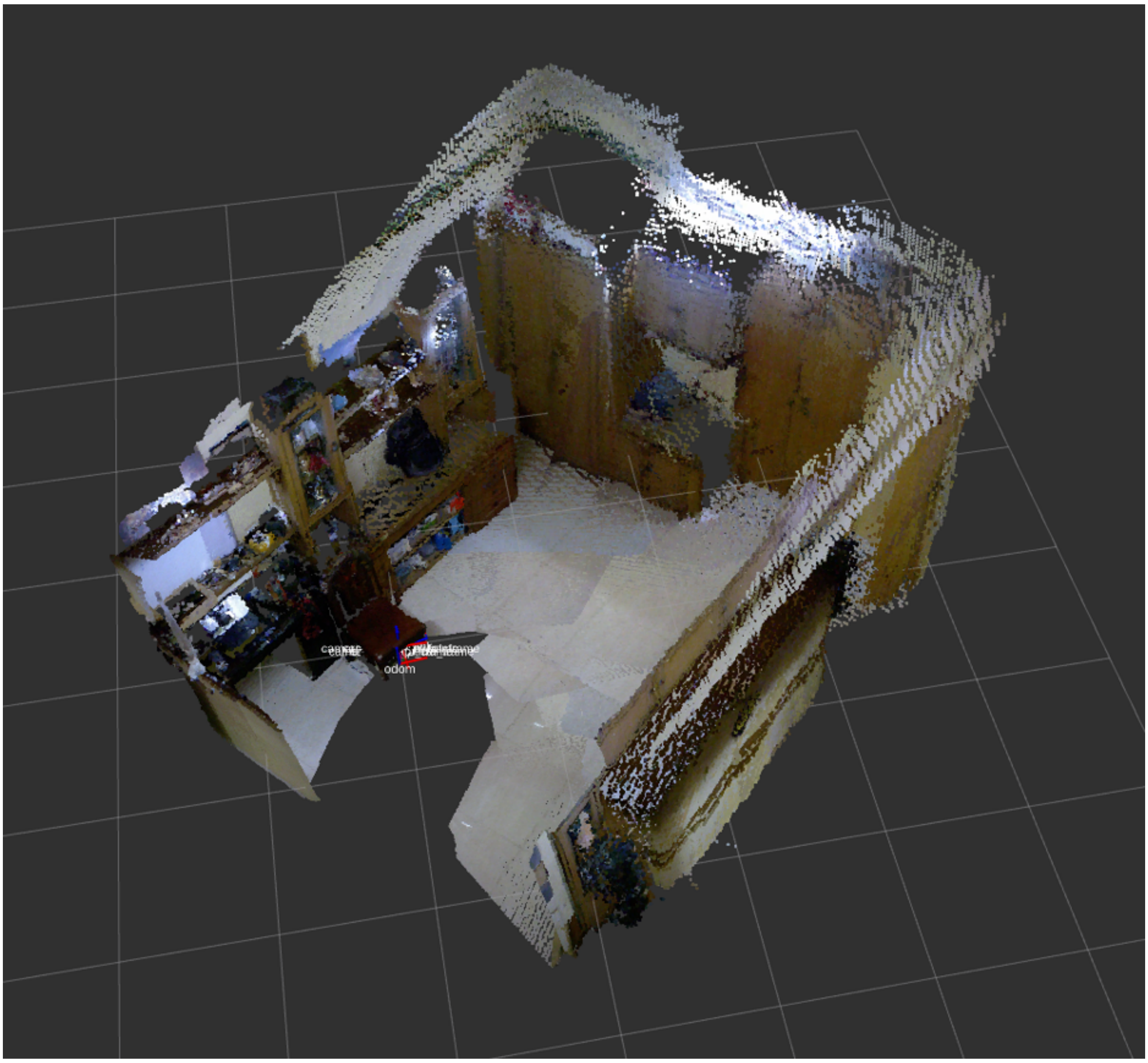



Figure 7.2: Wired mappping(PC)-complete

The complete formed map of the room is shown above. This can be interacted with and also be saved and loaded for later use. For example, continuing an earlier formed map to join new areas to it.

# Chapter 8

# Conclusions

An load emulation environment has been presented which is capable of simulating a electrical load in real-time. In this research the fundamental objective of the load emulation is to provide a simulated electrical load to allow an inverter to be tested at real power levels without the requirement of an actual load. The load emulation replaces the actual load during the testing and development stages of the inverter design, thus providing a safer and more flexible development environment. The ability to simulate an electrical load in real time is one of the key elements which facilitates in the load emulation.
The primary analysis and results show that acceptable accuracy can be achieved in real time using a digital signal processor dedicated to this task. The second requirement of the load emulation is the ability to draw current from the inverter equal to that predicted by the real time load model. To do this, the load emulation incorporates its own internal bidirectional converter which acts as a controllable voltage source. A current control loop ensures that this converter together with three-phase line inductors draw the appropriate current from the inverter. The transient response of the current loops determines the tracking accuracy between the demanded and actual current drawn from the inverter being tested. The passive components in the load emulation (i.e. the line inductors) act to slow the response of the system.
The future course work will be focused on an industrial induction motor drive, and to simulate virtual machine. It will be at real levels of voltage and current, the behavior of the actual machine. The results of testing a standard 'off-the-shelf inverter with the virtual machine will be compared with those obtained by testing the same inverter with the actual machine.

# Bibliography


[1] Peter M Visser, Marcel A Groothuis and Jan F Broenink, 'Multi-disciplinary design support using Hardware-in-the loop simulation ,' *Proceedings of the* $5^t h$ *Progress Symposium on Embedded Systems*, October 20, 2004, NBC Nieuwegein, NL pp 206-213.

[2] Qing Li, and Caroline Yao: 'Real-time concepts for embedded systems,' *CMP Books.*

[3] M.C. Chandorkar, Deepakraj M, 'Control of Parallel Connected Inverters in Standalone ac Supply Systems,' *IEEE Transactions Industry Applications.* vol. 29, no. 1, pp. 136-143, March 1993.

[4] Ram Kelkar, Ronnie A. Wunderlich and Leonard J. Hitchcock,'Software breadboards for power electronics circuits,' *IEEE Transactions on Power Electronics* vol. 6, no. 2 pp. 170-179, April 1991.

[5] Rainer Kraus and Hans Jurgen Mattausch, 'Status and Trends of power semiconductor device models for circuit simulation,' *IEEE Transactions on Power Electronics* vol. 13, no. 3, pp. 452-466, May 1998.