A MINOR PROJECT REPORT ON 'MARS ROVER',

SUBMITTED IN PARTIAL FULFILLMENT FOR THE

AWARD OF THE DEGREE OF BACHELOR OF TECHNOLOGY IN

ELECTRONICS AND COMMUNICATION ENGINEERING THROUGH

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

2012-2013

SUBMITTED TO:

THE DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING,

ARYA COLLEGE OF ENGINEERING AND I.T.
RIICO INDUSTRIAL AREA, KUKAS, JAIPUR

SUBMITTED BY:

ASHEESH SHARMA

09EAREC019

CONTENTS

ARYA College of Engineering & Information Technology
Approved by AICTE • Affiliated to University of Rajasthan, Jaipur
SP-42, Kukas Industrial Area (RIICO), Delhi Road, Kukas, Jaipur (Raj.) INDIA
Tel : +91-1426-247171,76-79 • Fax : 01426 - 247142

Date: _ _-_ _-_ _ _ _

CERTIFICATE

This is to certify that Mr. Asheesh Sharma, roll number 09EAREC019 delivered seminar on 'Mars Rover' and submitted project report to electronics & communication engineering department.

Signature	**Signature**	**Signature**
Mrs. Kirti Vyas	**Er. Ankit Mathur**	**Er. Rahul Srivastava**
Seminar Incharge	**Project guide**	**Head Of Dept., ECE**

C.S.

Dr. A.K.Sharma

Senior Professor

(Department of Electronics & Communication Engineering)

ACKNOWLEDGEMENT

I express sincere thanks to my project guide, Er. Ankit Mathur, for guiding me right form the inception till the successful completion of the project. I thankfully acknowledge Mr. Rahul Srivastava for extending his valuable guidance for literature, critical reviews of the project and this report. Above all, the moral support he provided during all stages of this project. I am also grateful to Dr. A.K Sharma, for providing me an opportunity to undergo this study.

I would also like to thank the staff of electronics and communication department, for their help and cooperation throughout the project.

Lastly, I also thank my family for their never-ending moral support and motivation.

LIST OF FIGURES

LIST OF TABLES

PREFACE

From many years, astrologers have changed our beliefs about the universe and its composition. To understand universe, it must be observed and studied. To study the observable universe, many sophisticated technologies are required. In Space few laws become universal truth. In such an alien environment, attempting to study requires new techniques and knowledge. New technologies originating from space-related endeavors are often useful in other economic activities. Above all, most essential and interesting studies are about life on a planet. In this project, we try to build a space vehicle, which can perform many sophisticated tasks. It is designed to move across the surface of an alien planet or other astronomical body. It works like a 'Swiss army knife' for astronomers (like lifting rocks, drilling and retrieving samples). It is also equipped with sensors to record the planet's environment and tectonic movements. It is designed to withstand high levels of acceleration, high and low temperatures, pressure, dust, corrosion, cosmic rays, remaining functional without repair for a needed period of time. It is capable of operating autonomously with little assistance from ground control as far as navigation and data acquisition are concerned, although it still requires human input for identifying promising targets in the distance to which it can drive. Rovers and other space vehicles are still largely dependent on commands from their human controllers back on Earth. But to decide what commands to send, operators must wait to receive images and other pertinent information from the spacecraft. This project tends to implement smart controlling devices, which enable fast control. In some cases these SCDs help in building artificially intelligent controlling algorithms.

Chapter-1

INTRODUCTION

Space communication is critical to all NASA missions. Astronauts, mission controllers and scientists depend upon the reliable transmission if information between the ground and spacecraft in low earth orbit (LEO) or deep space. Deep space network (one of may satellite networks used by NASA) is used for communication with SPIRIT.

1.1 Case Study: Communication with Mas rover SPIRIT

Step 1: Communication can only take place when the spacecraft is visible to a network node. Periods of visibility are determined by spacecraft's orbit and earth's rotation. For Communication with SPIRIT, MADRID is selected. This simple means MRO will be visible to MADRID DSN at the selected time. Figure 1.1 Shows the Time zone and various DSNs.

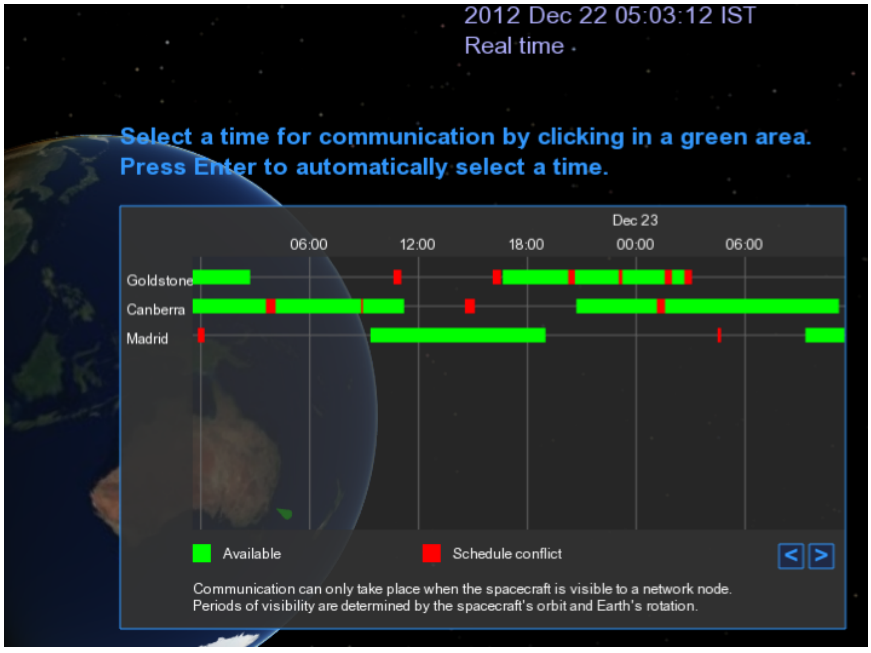

Figure 1.1 Time of communication

Step 2: At MADRID, Antenna is aligned with MRO as shown in figure 1.2. After Aligning, Commands to SPIRIT are sent via MRO as shown in figure 1.3.

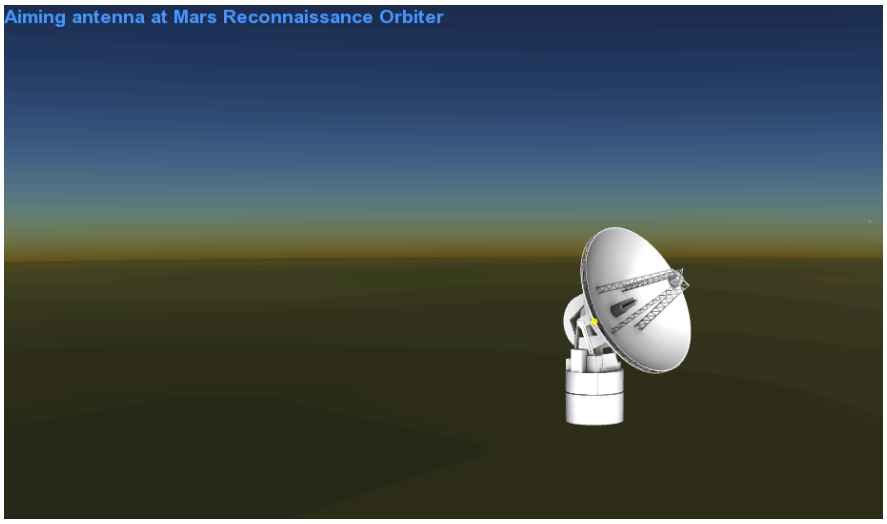

Figure 1.2 MADRID

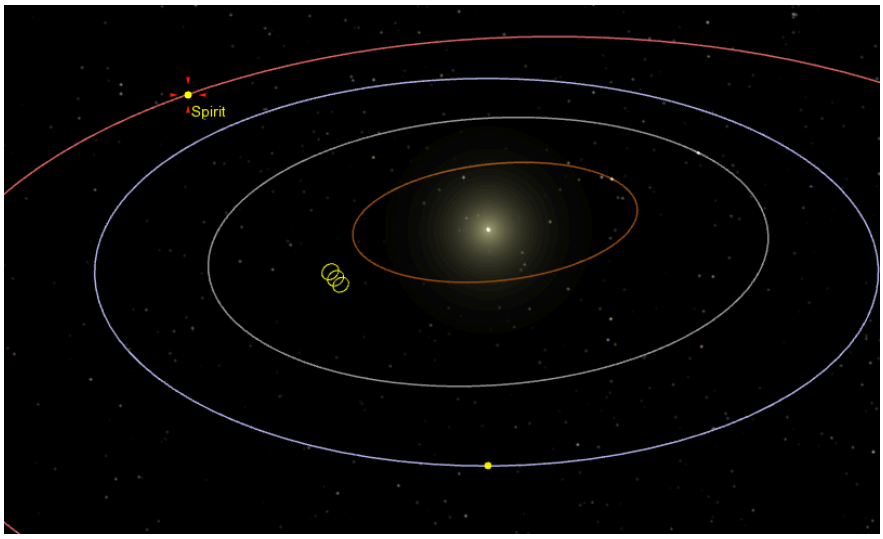

Figure 1.3 Signal transmissions

Step 3: After 21 minutes (figure 1.4), the signal is received by MRO. After receiving the signal, MRO needs to revolve over mars in order to align itself with the rover. (Figure 1.5)

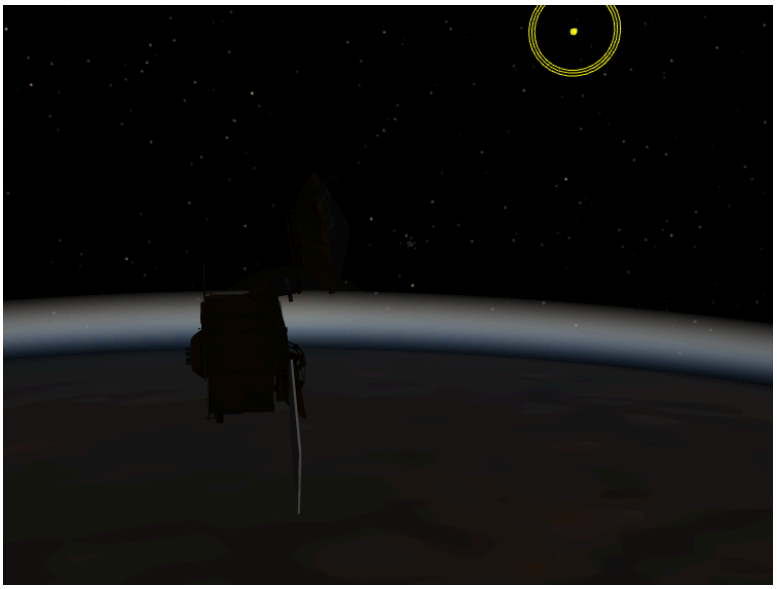

Figure 1.4 MRO

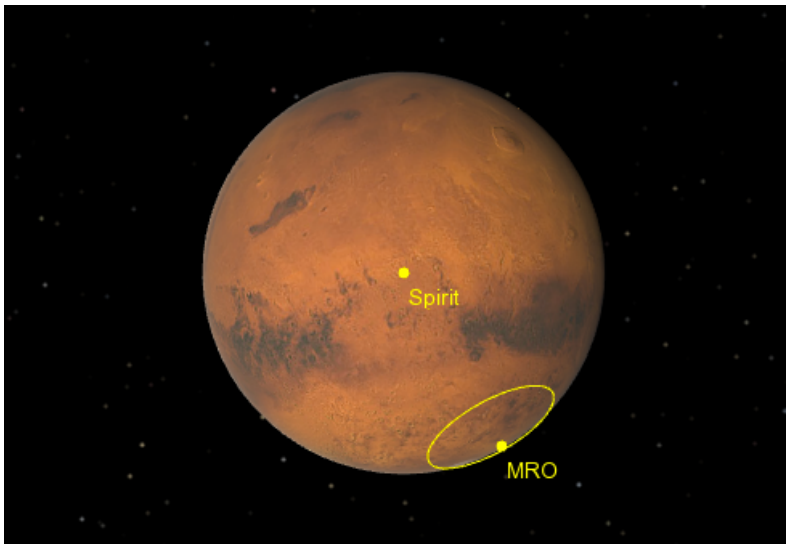

Figure 1.5 Signal transmissions

Step 4: After advancement of 5 days, MRO aligns to SPIRIT, and signal is transmitted to the rover (Figure 1.6 and 1.7).

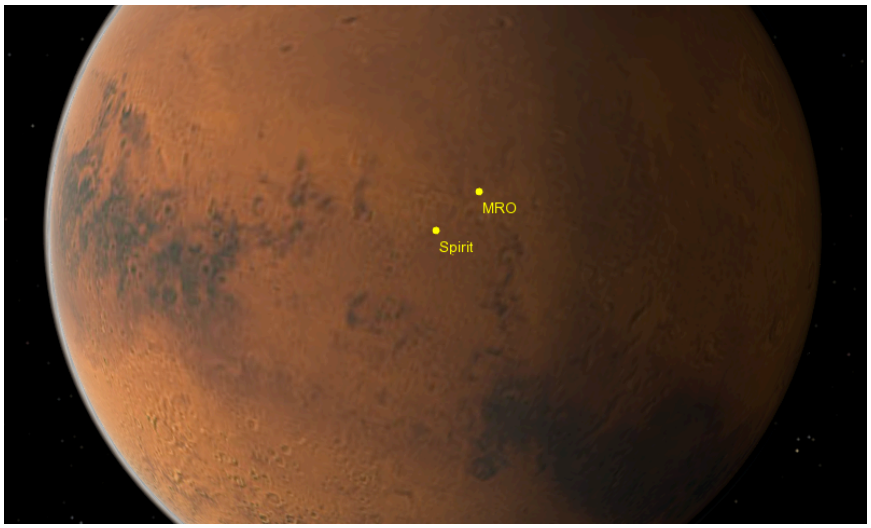

Figure 1.6 Commands being sent to SPIRIT

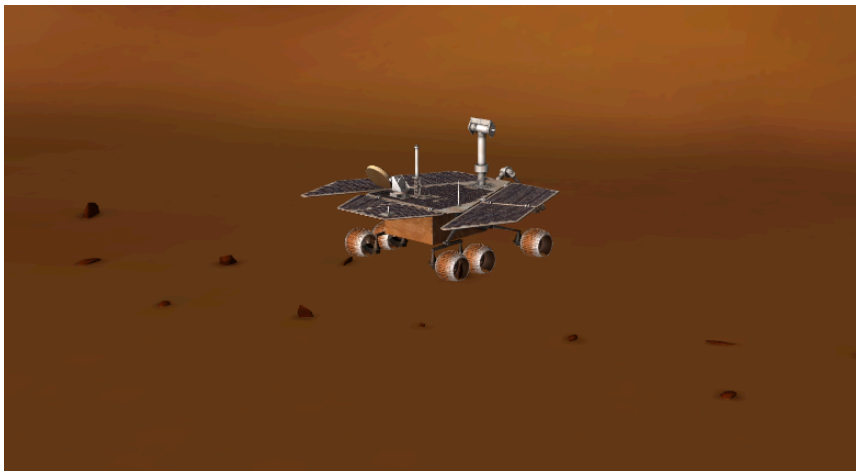

Figure 1.7 SPRIT following commands

1.2 **Conclusion**: It is evident that driving spirit on mars takes days in execution and months of planning. To minimize this time, it is needed to design a system, which automates execution and planning process.

Chapter-2

NAVIGATION

A satellite navigation system with global coverage may be termed a global navigation satellite system or GNSS. The Indian Regional Navigational Satellite System (IRNSS) is an autonomous regional satellite navigation system being developed by the Indian Space Research Organization (ISRO), which would be under total control of Indian government. The requirement of such a navigation system is driven by the fact that access to Global Navigation Satellite Systems, GPS, is not guaranteed in hostile situations

2.1 Present navigation systems:

A satellite navigation or SAT NAV system is a system of satellites that provide autonomous geo-spatial positioning with global coverage. It allows small electronic receivers to determine their location (longitude, latitude, and altitude) to within a few meters using time signals transmitted along a line-of-sight by radio from satellites. Receivers calculate the precise time as well as position, which can be used as a reference for scientific experiments.. The IRNSS would provide two services, with the Standard Positioning Service open for civilian use and the Restricted Service, encrypted one, for authorized users (military). Figure 2.1 Shows IRNSS coverage (Area).

Figure 2.1 IRNSS Coverage

2.2 IRNSS Architecture:

The proposed system would consist of a constellation of seven satellites and a support ground segment. Three of the satellites in the constellation will be placed in geostationary orbit. These GEOs will be located at 34 East 83 East and 132 East longitudes. The GSOs will be in orbits with a 24,000 km apogee and 250 km perigee inclined at 29 degrees. Two of the GSOs will cross the equator at 55 East and two at 111 East.

According to a presentation by A. Bhaskaranarayana [1], Director SCP/FMO & Scientific Secretary of the Indian Space Research Organization, to a meeting of COSPAR in Montreal on 15 July 2008, IRNSS signals will consist of a Special Positioning Service and a Precision Service. Both will be carried on L5 (1176.45 MHz) and S band (2492.08 MHz). The SPS signal will be modulated by a 1 MHz BPSK signal. The Precision Service will use BOC(5,2). The satellites would weigh approximately 1,330 kg and their solar panels generate 1,400 watts. The ground segment of IRNSS constellation would consist of a Master Control Center (MCC), ground stations to track and estimate the satellites' orbits and ensure the integrity of the network (IRIM), and additional ground stations to monitor the health of the satellites with the capability of issuing radio commands to the satellites (TT&C stations). Figure 2.2 shows the architecture.

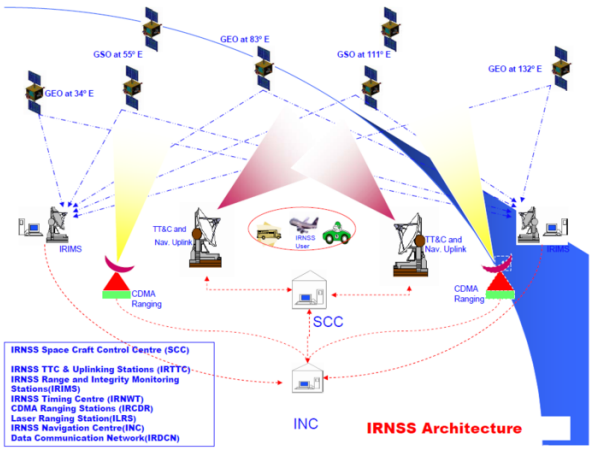

Figure 2.2 Architecture

2.3 GPS-aided geo-augmented navigation (GAGAN)

The GPS aided geo augmented navigation or GPS and geo-augmented navigation system (GAGAN) is a planned implementation of a regional satellite-based augmentation system (SBAS) by the Indian government. It is a system to improve the accuracy of a GNSS receiver by providing reference signals. The AAI's efforts towards implementation of operational SBAS can be viewed as the first step towards introduction of modern communication, navigation, surveillance/Air Traffic Management system over Indian airspace [2].

The project involves establishment of 15 Indian Reference Stations, three Indian Navigation Land Uplink Stations, three Indian Mission Control Centers and installation of all associated software and communication links. GAGAN is planned to get into operation by the year 2014. It will be able to help pilots to navigate in the Indian airspace by an accuracy of 3 m. This will be helpful for landing aircraft in tough weather and terrain like Mangalore airport and Leh.

The Rs. 774 crore (US$141 million) project is being implemented in three phases through 2008 by the Airport Authority of India with the help of the Indian Space Research Organization's (ISRO) technology and space support. The goal is to provide navigation system for all phases of flight over the Indian airspace and in the adjoining area. It is applicable to safety-to-life operations, and meets the performance requirements of international civil aviation regulatory bodies.

The space component will become available after the GAGAN payload on the GSAT-8 communication satellite, which was launched recently, is switched on. This payload was also on the GSAT-4 satellite that was lost when the Geosynchronous Satellite Launch Vehicle (GSLV) failed during launch in April 2010. Final System Acceptance Test will be conducted during June 2012 followed by system certification during July 2013.

To begin implementing an satellite-based augmentation system over the Indian airspace, Wide Area Augmentation System (WAAS) codes for L1 frequency and L5 frequency were obtained from the United States Air Force and U.S Department of Defense on November 2001 and March 2005. The system will use eight reference stations located in Delhi, Guwahati, Kolkata, Ahmedabad, Thiruvananthapuram, Bangalore, Jammu and Port Blair,

and a master control center at Bangalore. US defense contractor Raytheon has stated they will bid to build the system.

A national plan for satellite navigation including implementation of Technology Demonstration System (TDS) over the Indian air space as Airports Authority of India (AAI) and ISRO had prepared a proof of concept jointly. TDS was successfully completed during 2007 by installing eight Indian Reference Stations (INRESs) at eight Indian airports and linked to the Master Control Center (MCC) located near Bangalore. Preliminary System Acceptance Testing has been successfully completed in December 2010. The ground segment for GAGAN, which has been put up by the Raytheon, has 15 reference stations scattered across the country. Two mission control centers, along with associated uplink stations, have been set up at Kundalahalli in Bangalore. One more control center and uplink station is to come up at Delhi. As a part of the programmed, a network of 18 total electron content (TEC) monitoring stations were installed at various locations in India to study and analyze the behavior of the ionosphere over the Indian region.

GAGAN's TDS signal in space provides a three-meter accuracy as against the requirement of 7.6 meters. Flight inspection of GAGAN signal is being carried out at Kozhikode, Hyderabad, Nagpur and Bangalore airports and the results have been satisfactory so far.

One essential component of the GAGAN project is the study of the ionosphere behavior over the Indian region. This has been specially taken up in view of the rather uncertain nature of the behavior of the ionosphere in the region. The study will lead to the optimization of the algorithms for the ionosphere corrections in the region.

To study the ionosphere behavior more effectively over entire Indian airspace, Indian universities and R&D labs, which are involved in the development of regional based inotropic model for GAGAN, have suggested nine more TEC stations.

GAGAN after its final operational phase completion, will be compatible with other SBAS systems such as the Wide Area Augmentation System (WAAS), the European Geostationary Navigation Overlay Service (EGNOS) and the Multi-functional Satellite Augmentation System (MSAS) and will provide seamless air navigation service across regional boundaries. While the ground segment consists of eight reference stations and a master control center,

which will have sub systems such as data communication network, SBAS correction and verification system, operations and maintenance system, performance monitoring display and payload simulator, Indian land unlinking stations will have dish antenna assembly. The space segment will consist of one geo-navigation transponder.

The first GAGAN transmitter was integrated into the GSAT-4 geostationary satellite, and had a goal of being operational in 2008. Following a series of delays, GSAT-4 was launched on 15 April 2010, however it failed to reach orbit after the third stage of the Geosynchronous Satellite Launch Vehicle Mk.II that was carrying it malfunctioned.

In 2009, Raytheon had won a 82 million dollar contract. It was mainly dedicated to modernize Indian air navigation system. The vice president of Command & Control Systems, Raytheon Network Centric Systems, Andy Zogg commented:

"GAGAN will be the world's most advanced air navigation system and further reinforces India's leadership in the forefront of air navigation. GAGAN will greatly improve safety, reduce congestion and enhance communications to meet India's growing air traffic management needs"

2.4 Geosynchronous Satellite Launch Vehicle (GSLV)

The Geosynchronous Satellite Launch Vehicle (usually known by its abbreviation, GSLV) is an expendable launch system operated by the Indian Space Research Organisation (ISRO). It was developed to enable India to launch its INSAT-type satellites into geostationary orbit and to make India less dependent on foreign rockets.

GSLV has accomplished seven launches to date, since its first launch in 2001 through its most recent launch in 2010. Two launches have been successful, with one launch partially successful. Four launches have failed. The eighth flight for the GSLV is scheduled for early 2013.

2.4.1 History

The Geosynchronous Satellite Launch Vehicle (GSLV) project was initiated in 1990 with the objective of acquiring launch capability for geosynchronous satellites. Until then, India depended on the former Soviet Union for the launch of heavy satellites.

GSLV uses major components that are already proven in the Polar Satellite Launch Vehicle (PSLV) launchers in the form of the S125/S139 solid booster and the liquid fueled rocket engine Vikas engine. The first development flight of GSLV Mk.I (GSLV-D1) was launched on 18 April 2001. GSLV-F04 is the fifth flight of India's Geosynchronous Satellite launch Vehicle (GSLV), launched INSAT-4CR satellite, into a Geosynchronous Transfer Orbit (GTO) of 170 km perigee and 35,975 km apogee with an orbital inclination of 21.7 degree with respect to equator on September 2, 2007. Subsequently, the satellite was maneuvered into geostationary orbit using its own propulsion system.

The 49 m tall GSLV, with a lift-off mass of 415 tone, is a three-stage vehicle with solid, liquid and cryogenic stages. The first stage of GSLV, one of the largest in the world, uses Hydroxyl Terminated Polybutadiene (HTPB) based propellant. The second stage and the four strap-on motors surrounding the first stage use liquid propellant 'Vikas' engine burning UH25 and Nitrogen Tetraoxide. The third stage is a cryogenic stage using liquid Hydrogen as fuel and liquid Oxygen as oxidiser. GSLV employs S-band telemetry and C-band transponders for enabling vehicle performance monitoring, tracking, range safety/flight safety and Preliminary Orbit Determination.

The payload fairing, which is 7.8 m long and 3.4 m in diameter, protects the vehicle electronics and the spacecraft during its ascent through the atmosphere It is discarded when the vehicle reaches an altitude of about 115 km.

The Redundant Strap Down Inertial Navigation System/Inertial Guidance System of GSLV housed in its equipment bay guides the vehicle from lift-off to spacecraft injection. The digital auto-pilot and closed loop guidance scheme ensure the required attitude manoeuvre and guide injection of the spacecraft to the specified orbit.

2.5 Global Positioning System (GPS)

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world.

The GPS project was developed in 1973 to overcome the limitations of previous navigation

systems, integrating ideas from several predecessors, including a number of classified engineering design studies from the 1960s. GPS was created and realized by the U.S. Department of Defense (DoD) and was originally run with 24 satellites. It became fully operational in 1994.

Advances in technology and new demands on the existing system have now led to efforts to modernize the GPS system and implement the next generation of GPS III satellites and Next Generation Operational Control System (OCX). Announcements from the Vice President and the White House in 1998 initiated these changes. In 2000, U.S. Congress authorized the modernization effort, referred to as GPS III.

In addition to GPS, other systems are in use or under development. The Russian Global Navigation Satellite System (GLONASS) was in use by only the Russian military, until it was made fully available to civilians in 2007. There are also the planned European Union Galileo positioning system, Chinese Compass navigation system, and Indian Regional Navigational Satellite System.

The first satellite navigation system, Transit, used by the United States Navy, was first successfully tested in 1960. It used a constellation of five satellites and could provide a navigational fix approximately once per hour. In 1967, the U.S. Navy developed the Timation satellite that proved the ability to place accurate clocks in space, a technology required by GPS. In the 1970s, the ground-based Omega Navigation System, based on phase comparison of signal transmission from pairs of stations, became the first worldwide radio navigation system. Limitations of these systems drove the need for a more universal navigation solution with greater accuracy.

While there were wide needs for accurate navigation in military and civilian sectors, almost none of those were seen as justification for the billions of dollars it would cost in research, development, deployment, and operation for a constellation of navigation satellites. During the Cold War arms race, the nuclear threat to the existence of the United States was the one need that did justify this cost in the view of the United States Congress. This deterrent effect is why GPS was funded. It is also the reason for the ultra secrecy at that time. The nuclear triad consisted of the United States Navy's submarine-launched ballistic missiles (SLBMs)

along with United States Air Force (USAF) strategic bombers and intercontinental ballistic missiles (ICBMs). Considered vital to the nuclear deterrence posture, accurate determination of the SLBM launch position was a force multiplier

During Labor Day weekend in 1973, a meeting of about 12 military officers at the Pentagon discussed the creation of a Defense Navigation Satellite System (DNSS). It was at this meeting that "the real synthesis that became GPS was created." Later that year, the DNSS program was named Navstar. With the individual satellites being associated with the name Navstar (as with the predecessors Transit and Timation), a more fully encompassing name was used to identify the constellation of Navstar satellites, Navstar-GPS, which was later shortened simply to GPS. The user segment is composed of hundreds of thousands of U.S. and allied military users of the secure GPS Precise Positioning Service, and tens of millions of civil, commercial and scientific users of the Standard Positioning Service. In general, GPS receivers are composed of an antenna, tuned to the frequencies transmitted by the satellites, receiver-processors, and a highly stable clock (often a crystal oscillator). They may also include a display for providing location and speed information to the user. Its number of channels often describes a receiver: this signifies how many satellites it can monitor simultaneously. Originally limited to four or five, this has progressively increased over the years so that, as of 2007, receivers typically have between 12 and 20 channels.

While originally a military project, GPS is considered a dual-use technology, meaning it has significant military and civilian applications.

GPS has become a widely deployed and useful tool for commerce, scientific uses, tracking, and surveillance. GPS's accurate time facilitates everyday activities such as banking, mobile phone operations, and even the control of power grids by allowing well synchronized hand-off switching.

Civilian: This antenna is mounted on the roof of a hut containing a scientific experiment needing precise timing.

Chapter-3

MARS ROVER TWINS

A Mars rover is an automated motor vehicle which propels itself across the surface of the planet Mars after landing. Rovers have several advantages over stationary landers: they examine more territory, they can be directed to interesting features, they can place themselves in sunny positions to weather winter months, and they can advance the knowledge of how to perform very remote robotic vehicle control. [3]

There have been four successful robotically operated Mars rovers. The Jet Propulsion Laboratory managed the Mars Pathfinder mission and its now inactive Sojourner rover. It currently manages the Mars Exploration Rover mission's active Opportunity rover and inactive Spirit, and, as part of the Mars Science Laboratory mission, the Curiosity rover. Figure 3.1 Shows mars rover over the surface of mars.

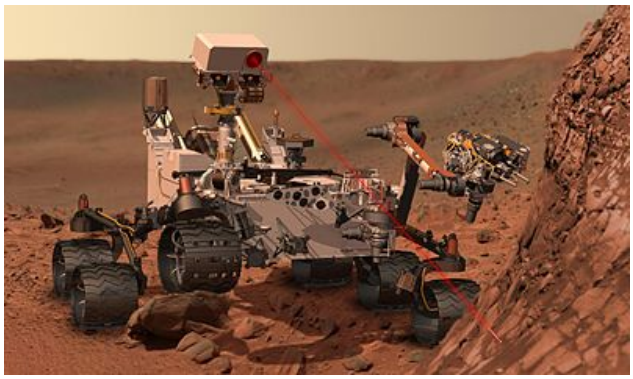

Figure 3.1 Mars rover

3.1 Spirit Rover

Spirit, MER-A (Mars Exploration Rover – A), is a robotic rover on Mars, active from 2004 to 2010. It was one of two rovers of NASA's ongoing Mars Exploration Rover Mission. It landed successfully on Mars at 04:35 Ground UTC on January 4, 2004, three weeks before its twin, Opportunity (MER-B), landed on the other side of the planet. Its name was chosen through a NASA-sponsored student essay competition. The rover became stuck in late 2009, and its last communication with Earth was sent on March 22, 2010.

The rover completed its planned 90-sol mission. Aided by cleaning events that resulted in higher power from its solar panels, Spirit went on to function effectively over twenty times longer than NASA planners expected following mission completion. Spirit also logged 7.73 km (4.8 mi) of driving instead of the planned 600 m (0.4 mi), allowing more extensive geological analysis of Martian rocks and planetary surface features. Initial scientific results from the first phase of the mission (the 90-sol prime mission) were published in a special issue of the journal Science.

On May 1, 2009 (5 years, 3 months, 27 Earth days after landing; 21.6 times the planned mission duration), Spirit became stuck in soft soil. This was not the first of the mission's "embedding events" and for the following eight months NASA carefully analyzed the situation, running Earth-based theoretical and practical simulations, and finally programming the rover to make extrication drives in an attempt to free itself. These efforts continued until January 26, 2010 when NASA officials announced that the rover was likely irrecoverably obstructed by its location in soft soil, though it continued to perform scientific research from its current location.

The rover continued in a stationary science platform role until communication with Spirit stopped on sol 2210 (March 22, 2010). JPL continued to attempt to regain contact until May 24, 2011, when NASA announced that efforts to communicate with the unresponsive rover had ended. A formal farewell was planned at NASA headquarters after the Memorial Day holiday and was televised on NASA TV.

The Jet Propulsion Laboratory (JPL), a division of the California Institute of Technology in Pasadena, manages the Mars Exploration Rover project for NASA's Office of Space Science, Washington.

3.1.1 Objectives:

The scientific objectives of the Mars Exploration Rover mission were to:

- Search for and characterize a variety of rocks and soils that hold clues to past water activity. In particular, samples sought will include those that have minerals deposited by water-related processes such as precipitation, evaporation, sedimentary cementation or hydrothermal activity.

- Determine the distribution and composition of minerals, rocks, and soils surrounding the landing sites.
- Determine what geologic processes have shaped the local terrain and influenced the chemistry. Such processes could include water or wind erosion, sedimentation, hydrothermal mechanisms, volcanism, and cratering.
- Perform calibration and validation of surface observations made by Mars Reconnaissance Orbiter instruments. This will help determine the accuracy and effectiveness of various instruments that survey Martian geology from orbit.
- Search for iron-containing minerals, identify and quantify relative amounts of specific mineral types that contain water or were formed in water, such as iron-bearing carbonates.
- Characterize the mineralogy and textures of rocks and soils and determine the processes that created them.
- Search for geological clues to the environmental conditions that existed when liquid water was present.
- Assess whether those environments were conducive to life.

During the next two decades, NASA would continue to conduct missions to address whether life ever arose on Mars. The search began with determining whether the Martian environment was ever suitable for life. Life, as we understand it, requires water, so the history of water on Mars is critical to finding out if the Martian environment was ever conducive to life. Although the Mars Exploration Rovers did not have the ability to detect life directly, they offered very important information on the habitability of the environment during the planet's history.

3.2 Curiosity rover

The Curiosity rover is a car-sized robotic rover exploring Gale Crater on Mars as part of NASA's Mars Science Laboratory mission (MSL).

Curiosity was launched from Cape Canaveral on November 26, 2011, at 10:02 EST aboard

the MSL spacecraft and successfully landed on Aeolis Palus in Gale Crater on Mars on August 6, 2012, 05:17 UTC. The Bradbury Landing site was less than 2.4 km (1.5 mi) from the center of the rover's touchdown target after a 563,000,000 km (350,000,000 mi) journey. The landing site coordinates are: 4.5895°S 137.4417°E.

The rover's goals include: investigation of the Martian climate and geology; assessment of whether the selected field site inside Gale Crater ever has offered environmental conditions favorable for microbial life, including investigation of the role of water; and planetary habitability studies in preparation for future human exploration.

Curiosity will serve as the basis for the design of an unnamed rover launching to Mars in 2020. In December 2012, its two-year mission was extended indefinitely.

3.2.1 Objectives:

As established by the Mars Exploration Program, the main scientific goals of the MSL mission are to help determine whether Mars could ever have supported life, as well as determining the role of water, and to study the climate and geology of Mars. The mission will also help prepare for human exploration.

Chapter-4

ARTIFICIAL INTELLIGENT ROVER: ANVESHAK

The purpose of Anveshak is to decrease the complexity of exploration in space. The rover is designed to mimic the features of SPIRIT rover. A major difference between the two rovers is that anveshak tends to perform task on it own. Unlike SPIRIT, Anveshak can be programmed to perform tasks on daily basis. It can differentiate between an interesting rock and others. Figure 4.1 shows a schematic of its software architecture:

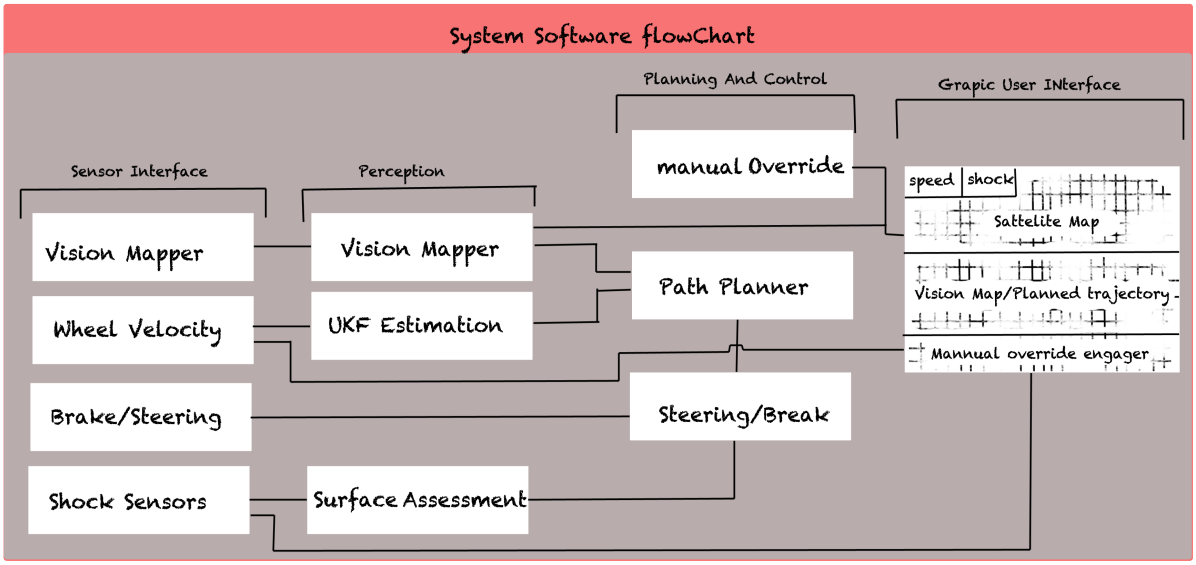

Figure 4.1 Four-layer system architecture

Layer 1: Sensor Interface

A laser-based sensor is used to calculate wheel velocity. Arduino with ATMega328, which has two photo-resistors, connected (with a 10k pull down resistor). Laser is aimed on a photo-resistor (which is enclosed within a dark box). The Arduino monitors the values returned from the light sensor, and watches for any changes that indicate that the laser beam has been broken. Arduino then calculates the amount of time between when each sensor was tripped. It then sends that value to the Adobe AIR based client, which is connected to the Arduino via USB. Figure 12 shows the schematic of the circuit. As a webcam can yield the required result for real-time performance, we use a 'minoru-3d camera', which is a stereo camera (30hz@ 20fps), yielding required time stamping.

Layer 2: Perception

Vision mapper: It is software, which actually calculates the 3d-Reconstruction data (Extrinsic matrix, Intrinsic matrix, camera parameter prediction matrix) and maps the environment in a point cloud. Based on Unscented Kalman filter (UKF), it uses camera pose estimation for measurement of car position (to be used for planning using the map obtained from vision mapper.

Layer 3: Planning and control

Steering and Break: It comprises direct interface with the steering and breaking actuators on the car. Once the path is obtained using path planner, it uses 'PID' algorithm to control break/steering.

Manual Override: it is a counter switch for manual (user) control, which can be engaged using the graphic user interface from computer.

Path planning: It uses A* search algorithm to plan routes using the map obtained.

Layer 4: Graphical user interface

It is used to present the data graphically to evaluate 'performance of ROVER (planned map, position on a satellite map etc.). Figure 1.3. Shows the software screenshot.

Figure 4.2 GUI

Chapter-5

CHASSIS

A chassis consists of an internal framework that supports a man-made object in its construction and use. It is analogous to an animal's skeleton. An example of a chassis is the under part of a motor vehicle, consisting of the frame (on which the body is mounted) with the wheels and machinery. For Anveshak, it is made up of Aluminum 6051. Gas welding was used to join beams. Any good chassis must do several things:

- Be structurally sound in every way over the expected life of the vehicle and beyond. This means nothing will ever break under normal conditions.
- Maintain the suspension mounting locations so that handling is safe and consistent under high cornering and bump loads.
- Support the body panels and other passenger components so that everything feels solid and has a long, reliable life.
- Protect the occupants from external intrusion.

Even though an individual rectangular tube is about 2% less stiff in torsion than the equivalent round tube, we must consider the chassis design as a whole. For each transverse tie-in we create a system that becomes more like a single large tube spanning the whole width of the chassis- the ultimate in efficiency. We have integrated 7 transverse members along our main rails in such a way that the chassis has much more torsional stiffness than the tubes taken individually. We even put extra braces on our central "X" member to make it even stronger.

The stiffness of an ideal unitized structure is proportional to the square of the distance of the components from the centerline. Double the distance and you have four times the overall stiffness. While practical automotive considerations eliminate an ideal connection between the rails, widely spaced tubes that are tied together well work more efficiently than the same tubes on a narrower base. The original 427 Cobras' rails were only 20 inches apart. Ours are spaced at 27 inches on center through the middle of the chassis, one of the widest spacing in the industry. And we still are one of the few in the industry that have left room for an under car exhaust outside the rails.

5.1 Chassis Design primitives:

Avoid complexity wherever possible. Strive for lower rather than higher component densities and use multifunctional components that reduce the total number of components needed for any given assembly. Minimize variety, such as the number of different types of component or values. Shun attributes known to have caused problems on previous projects. And keep an active database of new problems as they occur. Know the limits of your company's production capabilities and design in margins for error. It is not always necessary to have an exact knowledge of the process capability. As a matter of fact, it is often best to underestimate the process' capabilities (i.e., assume more restrictive process tolerances than actually exist).

An Analysis report was generated using Ansys Software solutions. When a 20 KN load was applied on the main joints of chassis, it could bear a force of 25 kg of weight. Figure 5.2 Shows a three dimensional isometric view of the chassis. Figure 5.1 Shows a screenshot of anveshak chasis analysis. Figure 5.3 shows the dimensions of chasis.

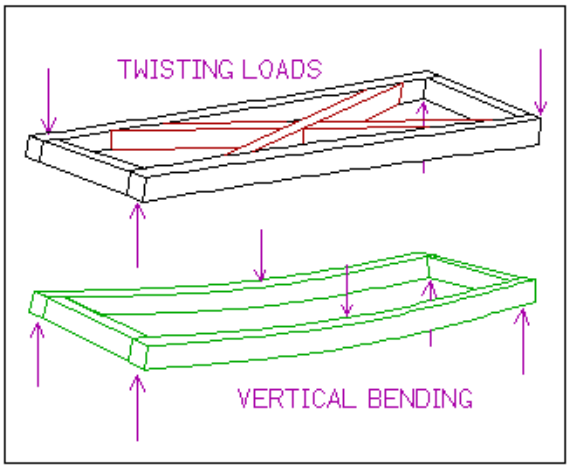

Figure 5.1 Applying loads

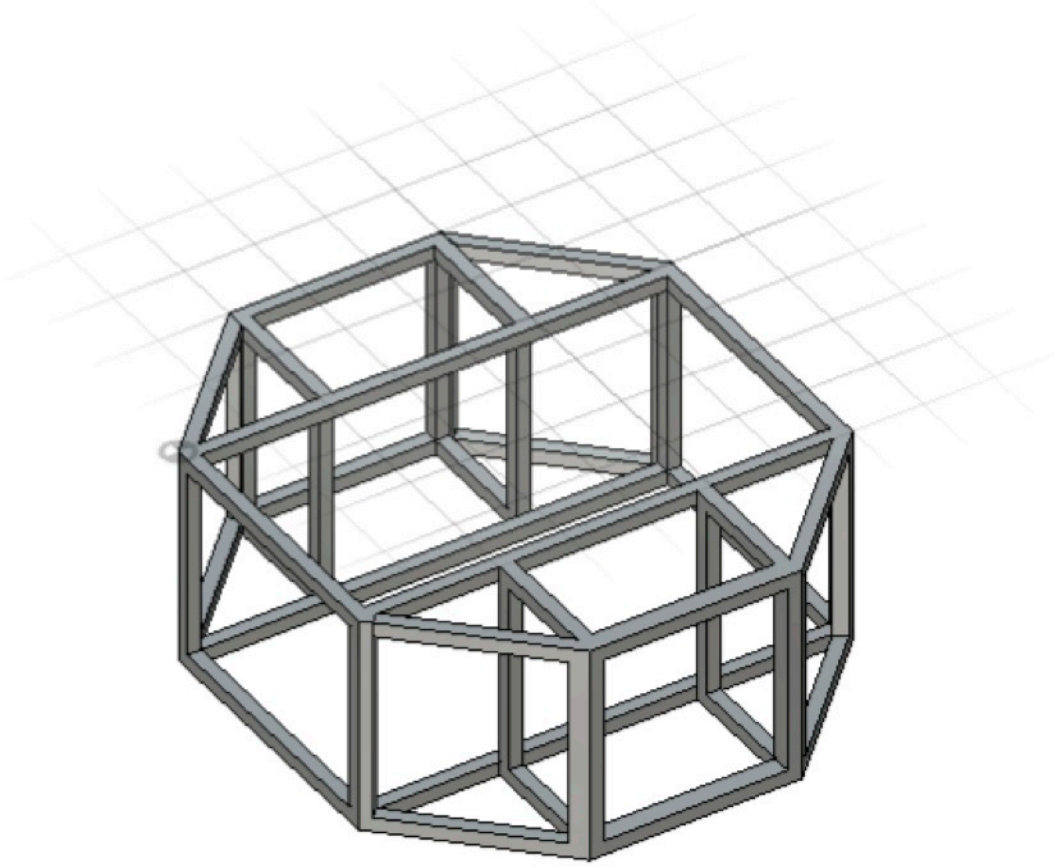

Figure 5.2 Three Dimensional view

Figure 5.3 Dimensions

Chapter-6

CIRCUIT DESIGN

Anveshak features a modular circuitary, which consists of smat controling devices, voltage supplies and a microcontroller. Figure 6.1 shows the over all circuit block diagram. Various parts and their roles are discribed below:

Arduino: It is a microcomputer, based on Atmega 2560 microprocessor. It monitors sensor data and allows control from serial command.

Motor control: It uses 12v power supply. It is a bidirectional, Pulse Width Modulating control system.

Voltage Supply: It consists of a solar panel charged battery array and a regulator for constant voltage supply needed for essential parts of the rover.

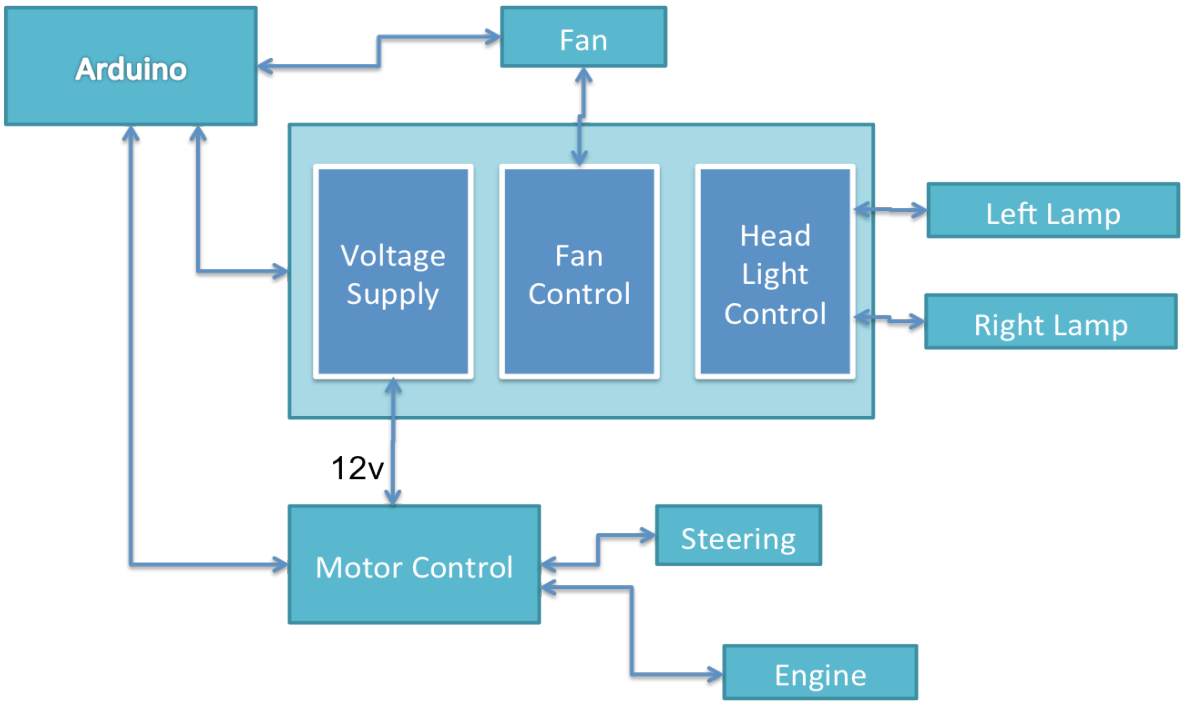

Figure 6.1 Circuit Block diagram

6.1 Bidirectional /pwm high power motor control

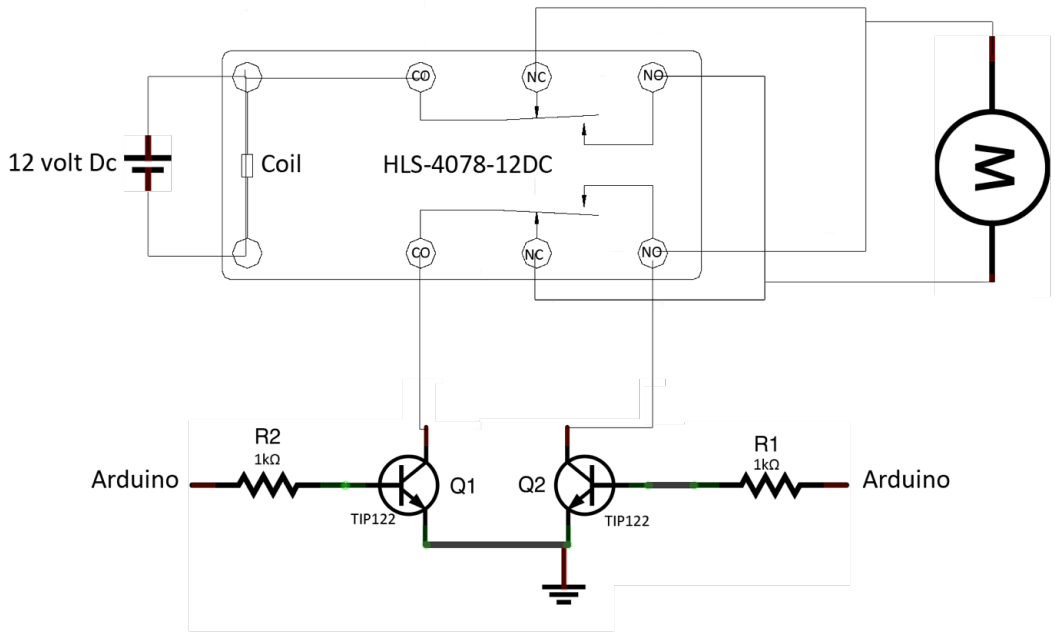

Figure 6.2 Motor controls

Description

Normally motors are controlled using H-Bridges, which has four transistors. The transistors are used as switches to control the motor polarity. Motor speed can be controlled using Pulse Width Modulation.

As the robot requires High voltage motors, conventional H-bridge Integrated circuits can't be used to deliver high current. Thus a new control technique has been implemented in the robot. Figure 6.2 shows the circuit diagram of the control mechanism. When a high pulse is applied to R2, it enables the relay. This allows the external 12-volt supply to energies DPDT relay coil. A TIP122 (Q1 with R2) is used to switch coil. Another TIP122 (Q2 with R1) is used to Switch Common (DPDT relay pin 'CO') terminal. Motor turns clockwise when Q2 receives logic HIGH. Thus the Control mechanism is Bidirectional. Table 6.1 shows the properties of parts shown in Figure 6.2

If short 10 milliseconds pulses are applied to Q2, speed of motor can be controlled. As the circuit switch on/off for every time Δt, increasing Δt increases the speed. This technique is called as Pulse Width Modulation control (PWM).

Observations and Comments

- Although Q1 can also be used to apply PWM signal, but it was ignored as switching relay for such short pulses is impractical (But possible) and inaccurate. As the mechanical switch life expectancy of limited (10,000,000), switching DPDT relay using PWM is not optimal.
- To be able to control to work, linear circuit paradigm must be respected, and thus microcontroller's ground should be in common with control circuit.
- Value of R1 and R2 is not mandatory, to deliver more power to motor (High than 24 volt), components can be replaced but underlying concept remains unchanged.
- Value of R1 and R2 is not mandatory, to deliver more power to motor (High than 24 volt), components can be replaced but underlying concept remains unchanged.

TABLE 6.1

BILL OF MATERIALS

Assembly List		
Label	Part Type	Properties
K1	Relay	Package THT; contact rating 125VAC @ 1AMP / 24VDC @ 2 AMP; Voltage 12V; switching circuit DPDT; part # HLS-4078-H-12VDC
M1	DC Motor	
Q1	TIP122	Package TO220 [THT]
Q2	TIP122	Package TO220 [THT]
R1	1k Ω Resistor	Package THT; tolerance ±5%; bands 4; resistance 1kΩ; pin spacing 400 mil
R2	1k Ω Resistor	Package THT; tolerance ±5%; bands 4; resistance 1kΩ; pin spacing 400 mil

6.2. Regulated voltage Supply, Exhaust Fan and Lamp Control

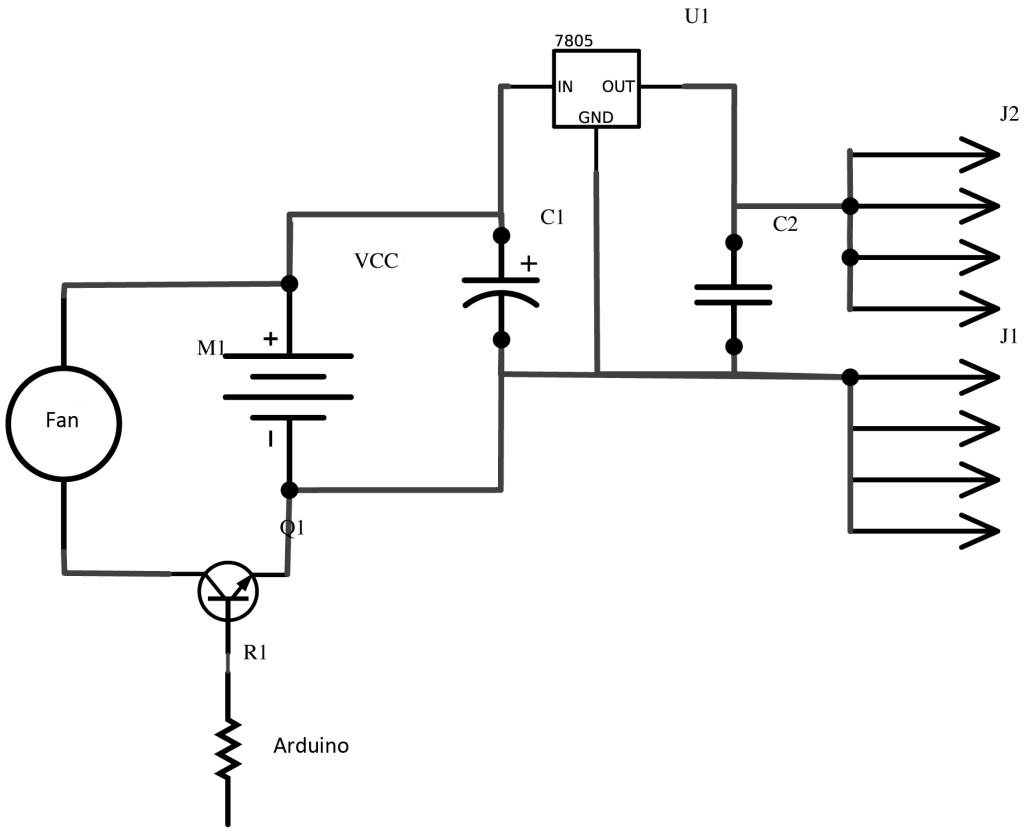

Figure 6.3 Regulated voltage Supply circuit

Description

Figure 6.3 shows the 5 volts regulated power supply, which can be used to power small sensor modules. These modules are designed to plug and play, thus are automatically detected by the system in programed fashion. Table 6.2 shows the properties of parts shown in Figure 6.3

TABLE 6.2

BILL OF MATERIALS

Assembly List		
Label	Part Type	Properties
C1	Electrolytic Capacitor	Package 100 mil [THT, electrolytic]; capacitance 10µF; voltage 6.3V
C2	Ceramic Capacitor	Package 100 mil [THT, multilayer]; capacitance 0.1µF; voltage 6.3V
J1	Generic male header - 4 pins	Package THT; hole size 1.0mm, 0.508mm; row single; form ♂ (male); pins 4; pin spacing 0.1in (2.54mm)
J2	Generic male header - 4 pins	Package THT; hole size 1.0mm, 0.508mm; row single; form ♂ (male); pins 4; pin spacing 0.1in (2.54mm)
M1	DC Motor	
Q1	TIP122	Package TO220 [THT];

R1	1 Ω Resistor	Package THT; tolerance ±5%; bands 4; resistance 1Ω; pin spacing 400 mil
U1	Voltage Regulator - 5V	Package TO220 [THT]; voltage 5V
VCC1	Battery	Voltage 6V
1	1 Ω Resistor	Package THT; tolerance ±5%; bands 4; resistance 1Ω; pin spacing 400 mil
1	TIP122	Package TO220 [THT]
1	Voltage Regulator - 5V	Package TO220 [THT]; voltage 5V
1	Battery	Voltage 6V

Chapter-7

SOFTWARE

The software follows the 4-layer architecture as discussed in chapter 4.

7.1. Parameter setup window

Interface is written on Linux native GUI design tool Glade interface designer. Following code results the parameter setup window of the software. Parameter setup window renders ability to tweak pinouts of Arduino on the fly. This ability requires parallel programming. Figure 7.1 shows the Parameter setup window.

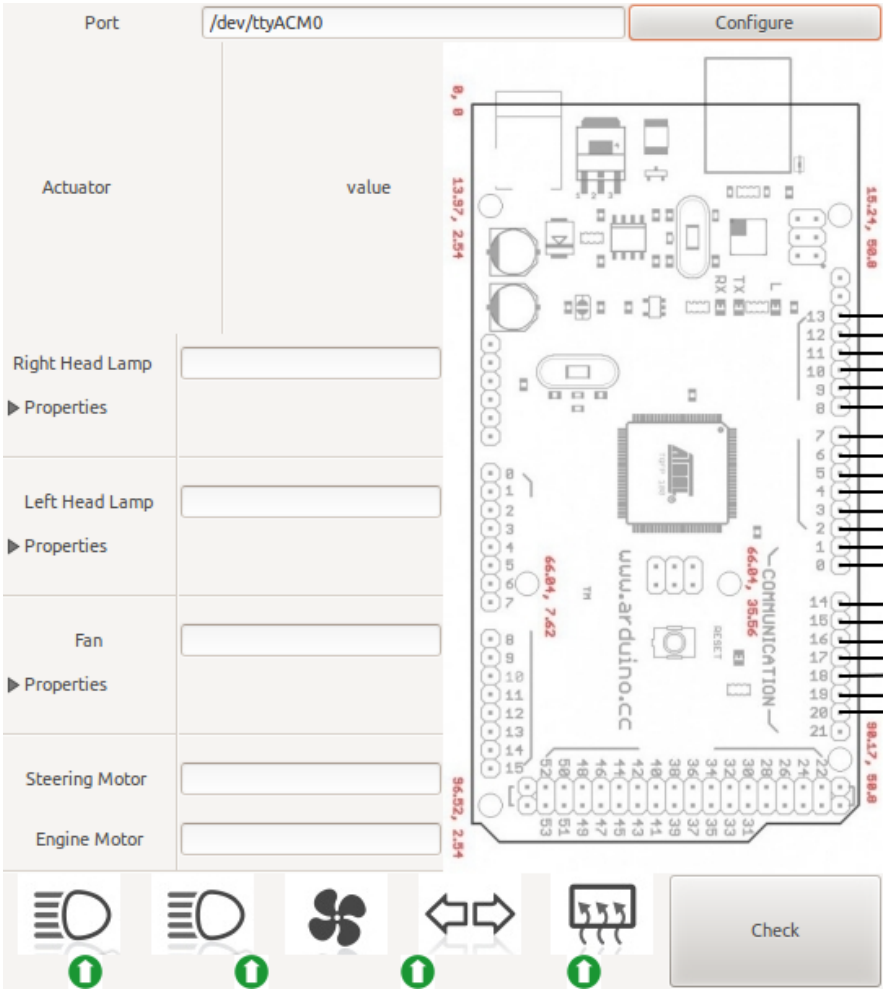

Figure 7.1 Parameter setup window

Code for gui of parameter setup window:

```
<?xml version="1.0" encoding="UTF-8"?>

<interface>

<requires lib="gtk+" version="2.24"/>

<!-- interface-naming-policy project-wide -->

<object class="GtkWindow" id="window1">

<property name="can_focus">False</property>

<property name="title" translatable="yes">Parameter Setup</property>

<property name="resizable">False</property>

<property name="icon">arduino-mega2560-holes-coordinates.jpg</property>

<child>

<object class="GtkVBox" id="vbox1">

<property name="visible">True</property>

<property name="can_focus">False</property>

<child>

<object class="GtkHBox" id="hbox9">

<property name="visible">True</property>

<property name="can_focus">False</property>

<child>

<object class="GtkLabel" id="label14">

<property name="visible">True</property>

<property name="can_focus">False</property>
```

<property name="ypad">2</property>

<property name="label" translatable="yes">Port</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">0</property>

</packing>

</child>

<child>

<object class="GtkEntry" id="entry9">

<property name="visible">True</property>

<property name="can_focus">True</property>

<property name="invisible_char">•</property>

<property name="primary_icon_activatable">False</property>

<property name="secondary_icon_activatable">False</property>

<property name="primary_icon_sensitive">True</property>

<property name="secondary_icon_sensitive">True</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">1</property>

</packing>

</child>

<child>

<object class="GtkButton" id="Conf">

<property name="label" translatable="yes">Configure</property>

<property name="visible">True</property>

<property name="can_focus">True</property>

<property name="receives_default">True</property>

<property name="use_action_appearance">False</property>

<signal name="clicked" handler="on_Conf_clicked" swapped="no"/>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">2</property>

</packing>

</child>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">1</property>

</packing>

</child>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">0</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image1">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="xalign">0</property>

<property name="yalign">0</property>

<property name="pixbuf">arduino-mega2560-holes-coordinates.jpg</property>

</object>

<packing>

<property name="expand">False</property>

<property name="fill">False</property>

<property name="position">1</property>

</packing>

</child>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">1</property>

</packing>

</child>

<child>

<object class="GtkHBox" id="hbox1">

<property name="visible">True</property>

<property name="can_focus">False</property>

<child>

<object class="GtkVBox" id="vbox3">

<property name="visible">True</property>

<property name="can_focus">False</property>

<child>

<object class="GtkHBox" id="hbox7">

<property name="visible">True</property>

<property name="can_focus">False</property>

<child>

<object class="GtkImage" id="image2">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">left.jpg</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">0</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image3">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">left.jpg</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">1</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image4">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">fan.jpg</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">2</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image5">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">steering.jpg</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">3</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image6">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">engine.jpg</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">4</property>

</packing>

</child>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">0</property>

</packing>

</child>

<child>

<object class="GtkHBox" id="hbox8">

<property name="visible">True</property>

<property name="can_focus">False</property>

<child>

<object class="GtkImage" id="image7">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">arrow_circle_green_right.png</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">0</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image8">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">arrow_circle_green_right.png</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">1</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image9">

<property name="can_focus">False</property>

<property name="pixbuf">arrow_circle_green_right.png</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">2</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image10">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="pixbuf">arrow_circle_green_right.png</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">3</property>

</packing>

</child>

<child>

<object class="GtkImage" id="image11">

<property name="visible">True</property>

<property name="can_focus">False</property>

<property name="xalign">0</property>

<property name="pixbuf">arrow_circle_green_right.png</property>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">False</property>

<property name="position">4</property>

</packing>

```
</child>
</object>
<packing>
<property name="expand">True</property>
<property name="fill">True</property>
<property name="position">1</property>
</packing>
</child>
</object>
<packing>
<property name="expand">True</property>
<property name="fill">True</property>
<property name="position">0</property>
</packing>
</child>
<child>
<object class="GtkButton" id="button1">
<property name="label" translatable="yes">Check</property>
<property name="visible">True</property>
<property name="can_focus">True</property>
<property name="receives_default">True</property>
<property name="use_action_appearance">False</property>
```

<signal name="clicked" handler="on_button1_clicked" swapped="no"/>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">1</property>

</packing>

</child>

</object>

<packing>

<property name="expand">True</property>

<property name="fill">True</property>

<property name="position">2</property>

</packing>

</child>

</object>

</child>

</object>

</interface>

7.1.1 Backend for Parameter Setup window

Glade only provides an interface window, a frontend. To menifest the interface a language is needed. Glade supports many languages like C, C++, java, Python. To implement parameter setup window as shown in Figure 17, Python was used.

Following code implements parallel programming code, it consists of a base program, which run bash scripts int.sh, int2.sh and internal.sh. These bash scripts run the ROSCORE process and a ROSPY process, before the program actually starts working.

Code for Int.sh:

```
#!/bin/bash

# Call this script with at least 3 parameters, for example

# sh scriptname 1 2 3

rosrun rosserial_python serial_node.py $
```

Code for internal.sh:

```
#!/bin/bash

# Call this script with at least 3 parameters, for example

# sh scriptname 1 2 3

gnome-terminal --tab -e "sh Int.sh $1" --tab -e "sh Int2.sh"
```

Code for internal.sh:

```
#!/bin/bash

# Call this script with at least 3 parameters, for example

# sh scriptname 1 2 3

gnome-terminal –tab –e “sh Int.sh $1” –tab –e “sh Int2.sh”
```

Python Code:

```
#!/usr/bin/env python
from gi.repository import Gtk, GdkPixbuf, Gdk
import rospy
from std_msgs.msg import UInt16
from std_msgs.msg import String
import os, sys
import time
class GUI:
        def __init__(self):
                self.gladefile = "tutorial-5c.glade"
                self.builder = Gtk.Builder()
                self.builder.add_from_file(self.gladefile)
                self.builder.connect_signals(self)
                window = self.builder.get_object('window1')
                window.show()
        def  on_button1_clicked(self, widget):
                pub = rospy.Publisher('servo', UInt16)
                rospy.init_node('nh')
                if not rospy.is_shutdown():
os.system("sh Internal.sh " + port.get_text())
        def destroy(window, self):
```

```
                Gtk.main_quit()
def main():
        app = GUI()
        Gtk.main()
if __name__ == "__main__":
   sys.exit(main())
#!/usr/bin/env python
from gi.repository import Gtk, GdkPixbuf, Gdk
import rospy
from std_msgs.msg import UInt16
from std_msgs.msg import String
import os, sys
import time
class GUI:
        def __init__(self):
                self.gladefile = "tutorial-5c.glade"
                self.builder = Gtk.Builder()
                self.builder.add_from_file(self.gladefile)
                self.builder.connect_signals(self)
                window = self.builder.get_object('window1')
                window.show()
        def  on_button1_clicked(self, widget):
```

```
                pub = rospy.Publisher('servo', UInt16)
                rospy.init_node('nh')
                if not rospy.is_shutdown():
os.system("sh Internal.sh " + port.get_text())
        def destroy(window, self):
                Gtk.main_quit()
def main():
        app = GUI()
        Gtk.main()
if __name__ == "__main__":
   sys.exit(main())
```

Chapter-8

STEPS TO TEST THE ROVER USING PARAMETER SETUP

Step 1: Connections: Pins of Arduino are connected according to Figure 16. Table 8.1 Shows a basic pin connections.

TABLE 8.1

CONFIGURATIONS

Subsystem	Arduino pin number
Right Head lamp	9
Left Head lamp	10
Fan	8
Steering Motor (relay enable)	12
Engine Motor (relay enable)	11
Steering Motor (Direction)	3
Engine Motor (Direction)	2

Step 2: Connect the Arduino cable to Computer.

Step 3: Open a Terminal window:

Type:

cd <Directory-of-python-file>

ls

Python <Python-File>

Figure 8.1 shows the terminal window with above code.

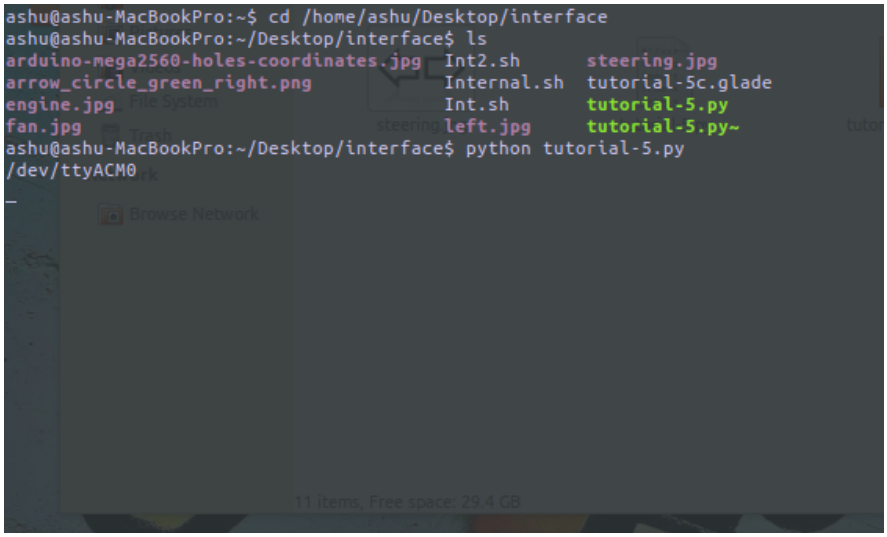

Figure 8.1 Terminal Window

Step 4: When the Parameter Shows up, Type the Arduino device address and hit configure button as shown in Figure 8.2.

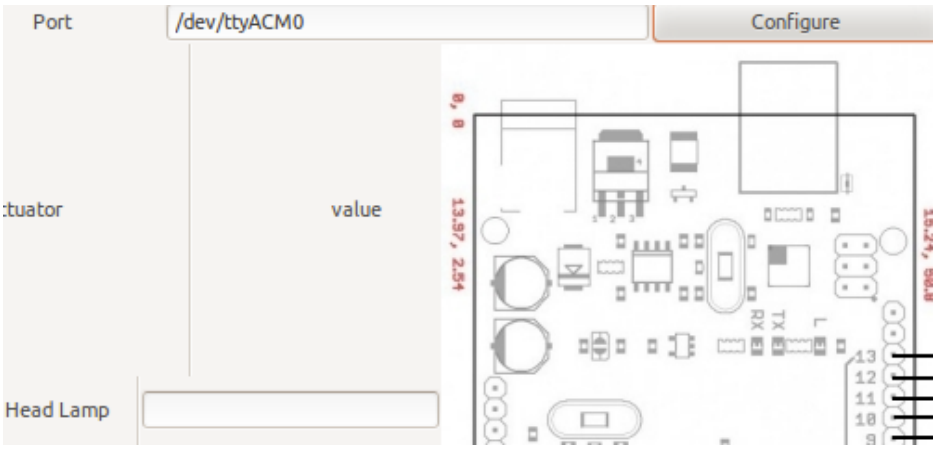

Figure 8.2 Enter port address

When the configure button is pressed, it opens a terminal window automatically, wakes up ROSCORE process and ROSPY child process. ROSCORE process is shown in figure 8.3. Rospy child process is show in figure 8.4.

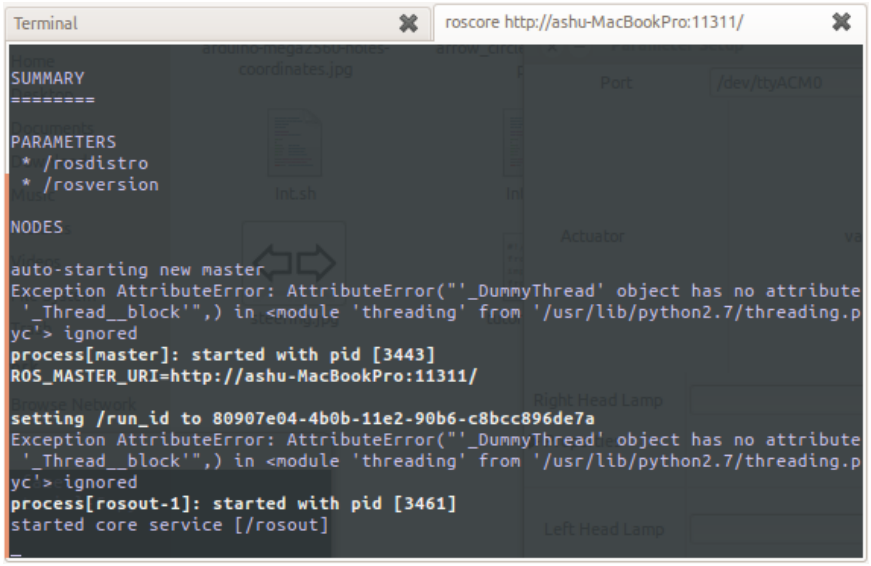

Figure 8.3 Terminal window

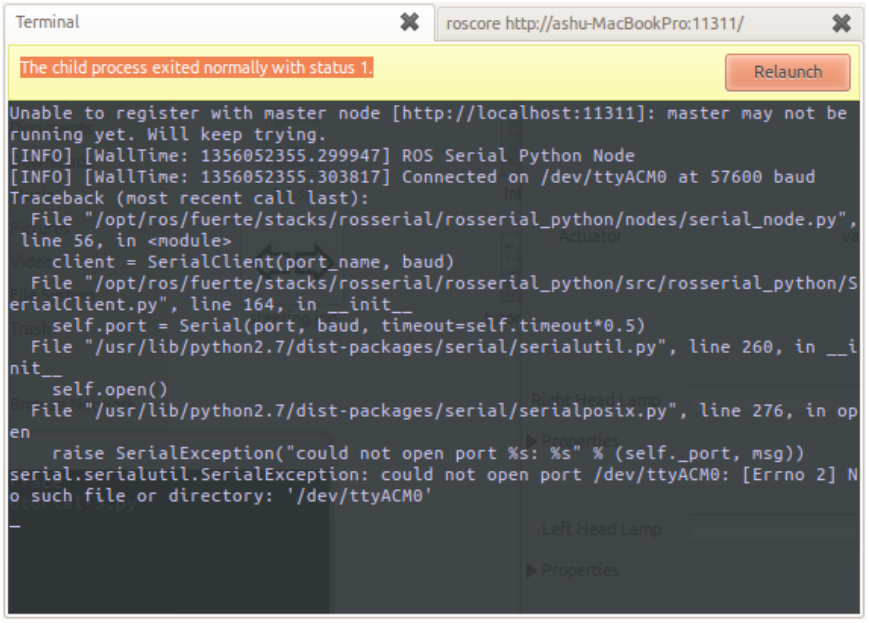

Figure 8.4 Terminal window

Step 5: Go back to Parameter setup window, and type pin numbers as per table 1. This will start Headlights, motors and fan one by one.

CONCLUSION

It is part of our nature, as human beings that we want to explore and try to better understand the world in which we live. Space exploration is the logical extension of that need. The space program provides an opportunity for nations to work together in international cooperation through participation in joint missions, thereby promoting peace and understanding between governments by their people. In future, leading nation would be a spacefaring country. As project Anveshak is open source at heart, it allows other researchers to modify and tweak designs. Instead of a strict development cycle, Anveshak has a flexible and evolving architecture. As the developer can pick between alternatives of a same module, many versions of rover (based on environment) can be designed. Looking to the future, Anveshak would evolve into more complex robot. 3d printing is one feature (yet to be encorporated), which can reduce cost of missions. In future Anveshak would be able to print itself out of materials available readily on mars. This could be a major breakthrough in space industries.

REFERENCES

[1] Presentation by A. Bhaskaranarayana, www.oosa.unvienna.org/pdf/icg/2008/expert/2-3.pdf .

[2] GPS-aided geo-augmented navigation (GAGAN), en.wikipedia.org/wiki/GPS-aided_geo-augmented_navigation

[3]. Mars Science Laboratory, http://en.wikipedia.org/wiki/Mars_Science_Laboratory