Sardar Vallabhbhai National Institute of Technology,Surat

“SOLAR_64”

Members:

Milie Bhanani
Meet Gandhi
Atharva Kalsekar
Rachana Bhagavath
Kalash Tiwari
Anup Phutane
Dhiman Airao
Prachi Agrawal
Pankaj Kumar Vijayvergiya
Rashmi Karnany
Parth Panchal
Mahesh Birajdar
Dhruv Hansaliya
Harshil Singhi
Divya Dashora
Kapil Jain
Devansh Jani

Introduction

The main aim of this project is to build an Extra-terrestrial Rover. Rover is a robot for exploring unknown terrains that can be driven remotely by signals from different locations. It is mainly used as a space exploration vehicle designed to move across the surface of a planet or other celestial body and is also capable of performing various tasks like measuring different physical parameters, collecting soil samples, managing objects, maintenance,astronaut assistance,autonomous task, etc.

MECHANICAL SYSTEM DESIGN & ANALYSIS

A rover, as a space exploration vehicle should be able to cross any difficult terrain. It should be able to overcome bumps and obstacles without jerks and vibrations. It should also be equipped with a robotic manipulator to perform certain tasks such as collecting samples, space shuttle repairing, hardware manipulation, etc. Our arm should be robust enough to sustain hazards such as shocks and should be stable during the motion of the chassis.

Problems encountered during IRC-2018:

1. **Steering Mechanism:** In our first edition of rover we used steering mechanism which is quite slow and had gear assembly which led to problems due to backlash.

2. **Ground Clearance Issue:** Last year's rover encountered problem due to mounting of motors which were prone to collide with obstacles.

3. **Bending of motor shaft:** The design of the rover was such that the max load of the rover was on the motor's shaft this led to poor performance of motor.

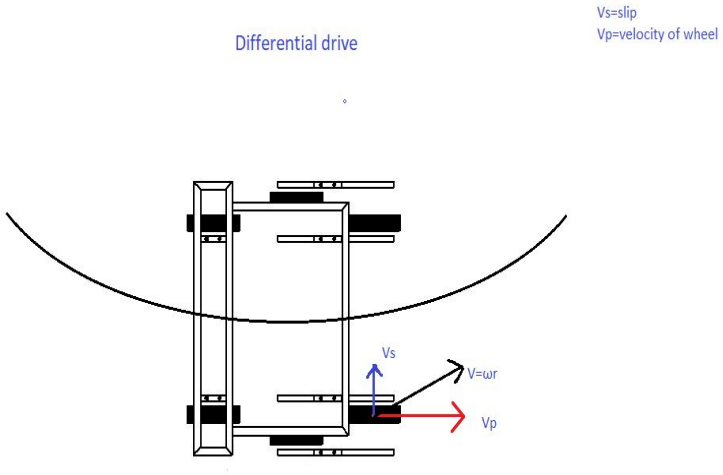
4. **End effector issue:** The end effector was two jaw gripper which was not efficient. Also the claw used for collection of soil had issue due to slipping of soil from the claw.

5. **Manipulator design:** The design of manipulator included mounting of motors and gears .This resulted in backlash error also its functionality was not satisfactory.

6. **Bogie issue:** In previous design pedestal bearing on the rocker part and horizontal shaft support on the bogie was used, this led to motion of both parts of bogie together which is not desired in Rocker bogie mechanism.

Solutions of problems faced in IRC-2018:

1. **Steering Mechanism**: To solve the problem of slow speed we shifted our drive from steering to differential drive and also the gears were eliminated.



2. **Ground Clearance Issue**: We increased the size of wheels to mainly avoid the problem and also changed the mounting of motor in a pipe such that it is not harmed by an obstacle.

3. **Bending of motor shaft**: To reduce the load we changed the motor mounting so that there is no load on the motor shaft and full load is transferred from the pipe to the wheels and finally to the ground without damaging the motor.

4. **End effector issue:** We optimised our gripper to 3 Jaw gripper with lead screw mechanism to get the perfect the grip.

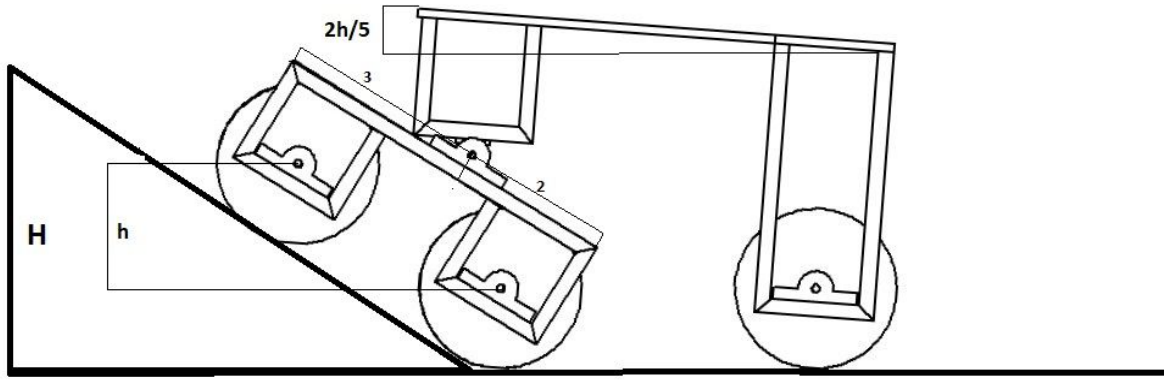
5. **Manipulator Design**: To avoid the use of gears we preferred linear actuators in manipulator which not only solved the problem of backlash but also are more efficient.

6. **Bogie issue:** The position of pedestal bearings and horizontal shaft support were changed, this resulted into free motion of both the parts of the bogie.

Design of Rover 1.1(Testing chassis):

Rocker Bogie Suspension System:

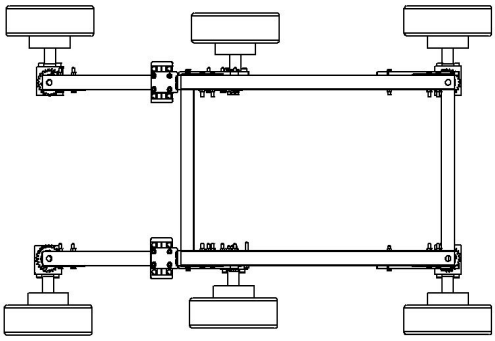
The basic design consists of a rocker and a bogie. Both the parts are hinged at a point which divides the bogie part in a ratio. In most of the cases 1:1 ratio is preferred but with rocker bogie mechanism 3:2 ratio was the most optimised solution.
In 1:1 ratio, if our front wheel stands on an obstacle of 'h' height then the hinge point lifts up to an height of h/2 this results into lifting of platform to a height of h/2. The ratio also provides less torque to lift the front wheel.
Whereas in the case of 3:2 when our front wheel is on 'h' height, then the hinge point is at a height of 2h/5 which provides more stability to the platform. This also serves the purpose by increasing the torque for lifting the front wheel.

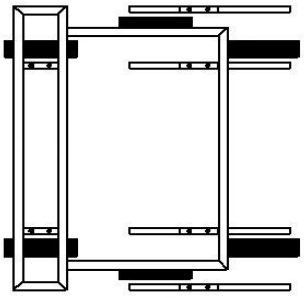


Differential Drive:

The rover is driven by the differential drive which solves the issue of steering mechanism which is slow, whereas differential drive is faster as it turns while moving and is more efficient. Differential drive includes skidding of wheels.

Normally in rover three wheels on the same side are in same plane but to make differential drive more efficient centre wheels are placed a bit outward. This brings all the three wheels of a side on the same arc while turning which results in equal skidding of three wheels, hence supports differential drive.





Wheels Design:

Grip is an integral part of the wheel especially while using differential drive so it is quite important to design the grip which fulfills all the required criteria.While turning in differential drive wheel has a velocity tangential to the arc of turning.There are two components of the velocity one being responsible for slipping and the other responsible for forward motion,so the grip should be as such it benefits both the motions.Fig.1 shows the grip helping the forward motion but as the rover turns at an angle it does help.Changes in fig.1 are shown in fig.2 which helps in forward motion as well as turning in one direction.Fig. 3 is introduced keeping all the problems in mind which supports linear motion and turning.

Fig.1

Fig.2

Fig.3

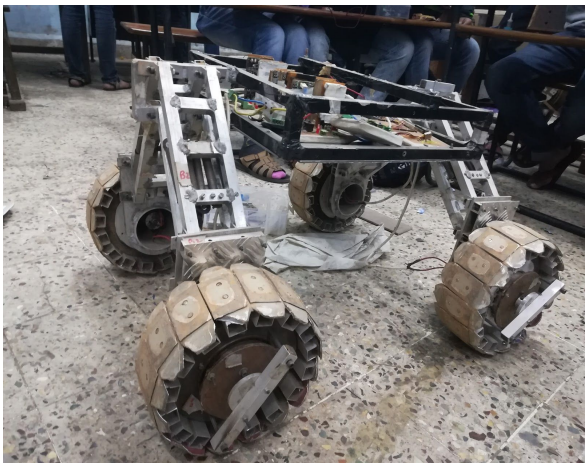
Different motors used in testing chassis:

- Firstly we used tauren motor in the testing chassis.It was providing enough torque and rpm but there was an issue of ground clearance with the motor.
- Then we used mechtex motor which somewhat solved the ground clearance issue.
- Then we tried to use wiper motor which have a high torque and solved the issue of ground clearance but had a low rpm.

Design of Solar-64 (Second edition rover)

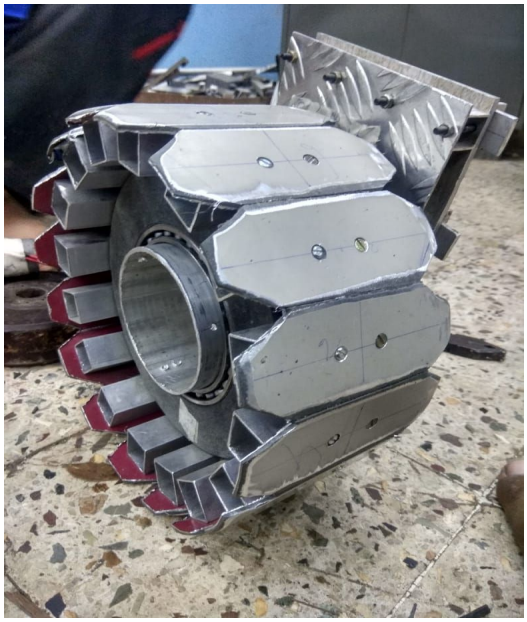
1) Differential drive: The Rover is driven by the differential drive which solves the steering mechanism issue. The differential drive is a two-wheeled drive system with independent actuators for each wheel. The name refers to the fact that the motion vector of the robot is sum of the independent wheel motions, something that is also true of the mechanical differential.Straight-line motion is accomplished by turning the drive wheels at the same rate in the same direction, although that's not as easy as it sounds. In-place (zero turning radius) rotation is done by turning the drive wheels at the same rate in the opposite direction. Arbitrary motion paths can be implemented by dynamically modifying the angular velocity and/or direction of the drive wheels.

2)Four wheel: Differential drive includes skidding of wheels. A shift from six wheel drive to a four wheel drive is implemented to support turning through differential drive. This shift also helped to decrease the weight of the Rover.

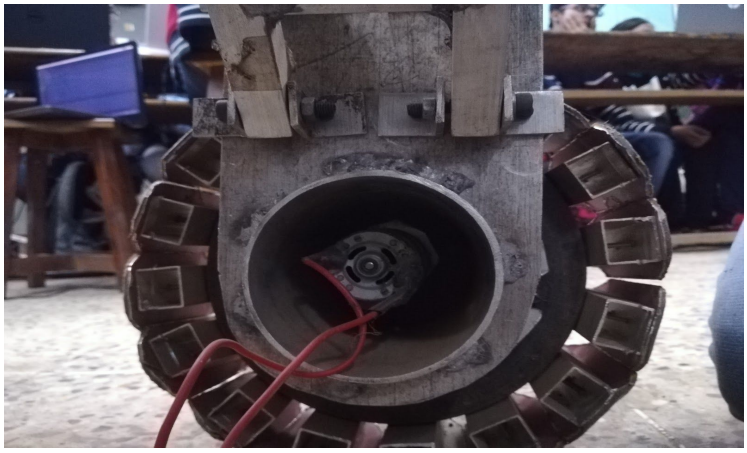


3)Spring suspension system: As the Rocker Bogie Mechanism itself is a self sustained design and doesn't require a suspension system while spring suspensions are implemented on the four wheel drive to fulfill the purpose**.**The suspension system generally comprises of compression springs but due to unavailability of compression springs we altered the mounting of our suspension such that we

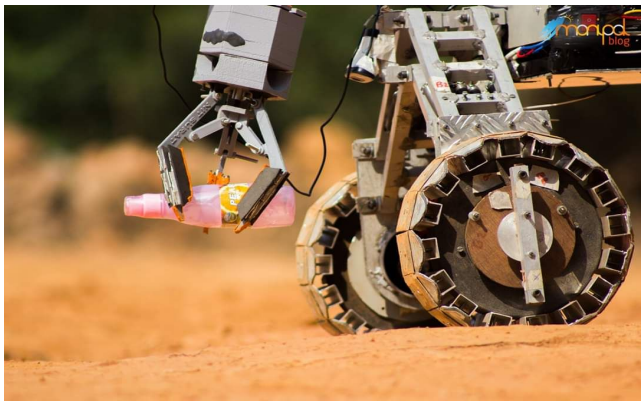
4)Wheel: Lightweight customized wheels are manufactured to provide traction and support the four wheel differential drive. Larger diameter wheels are designed to increase the ground clearance. The wheel was mounted on the aluminium pipe(which is further hinged to the chassis) with the help of bearings which provided effective rotation of wheels. For increasing the diameter of wheel, aluminium box section is used. Now to generate the circular profile composite material was used. This composite sheets were further modified to facilitate turning using differential drive.



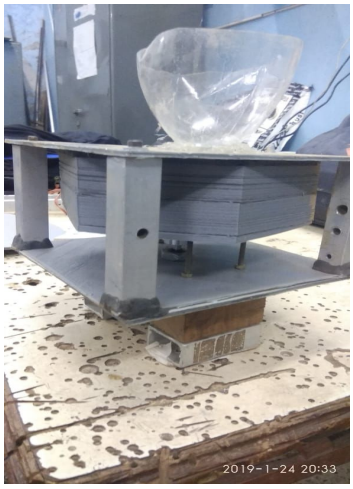
5)Motor Mounting: The motors are mounted inside the wheels. This protects the motors from any potential damage due to the obstacles. This assembly of motor also eliminates the major load on the shaft of motor which was a prominent issue in our previous edition rovers. It also helps to maintain a low centre of gravity and stability of the system. For motor mounting a hollow aluminium pipe is used, the motor is mounted on a wooden plate which is fixed with aluminium pipe. The motor is further coupled with the wheel using a coupler and box section assembly. The aluminium pipe is brazed between aluminium plates and this plates are hinged to the main frame.



6)Manipulator System Design: The Manipulator is actuated using linear actuators. This eliminates the backlash error issue caused by gears in the gear and motor mounting. The linear actuators are mounted so to change the angle between manipulator links hence enhancing the workspace of the manipulator. The first linear actuator is used to change the angle between the round table and first link. The second actuator is used to change the angle between first and second link, it is mounted between second link and round table. The third linear actuator is used for the motion of end effector. This whole assembly of links and linear actuators was mounted on the round table with help of pedestal bearings with proper calculations such as the end effector has wide and effective range of motion. Further the round table is mounted with the help of casters to provide smooth and efficient motion of manipulator.



7)Soil Collection : For soil collection a third link was designed on which the claw mounting was like a JCB. The claw was used to extract the soil. The collected soil is covered using a semicircular composite disc which was rotated with the help of a servo motor used to stop at any required angke which prevents the slipping of soil from claw. For soil collection module 3-D printed hexagon having six different compartments which supports us to collect different sample of soils from various sites of competition was used which was sealed using a composite plate. The sensors were mounted on a T-shaped composite plate and the assembly was actuated using a servo motor helping us to stop 3 different sensors at specific angles which were supposed to enter the hexagon box and measure the respective parameters . The assembly was moved in vertical axis using a rack and pinion assembly.



8)End effector: A three claw mechanism was tested and implemented on the manipulator to get a good grip and handle object with wide range of shapes and sizes.
For equipment servicing task the links of end effector were used having larger contact surface area to ease the task of turning knobs and flipping switches.
For astronaut assistance task the links were designed with pointed ends which helped to make initial contact with the object this helped to break the contact between the ground and object and hence making it easier to grip.

9)Camera mounting: The motion of the rover was guided using 3 cameras. The first and foremost the navigation camera was mounted on a vertical link at a specific height which was capable of viewing in all directions. It was guided using two wega motors having their axis perpendicular to each other.
The second camera was mounted on manipulator to judge the depth and the position of end effector.
The third camera was mounted on the front of the rover to have a closer look at the ground and the type of soil. Also it helped to have a closer look at the objects to be gripped.

10) Circuit Covering: A circuit covering was made using composite sheets to cover the electronic circuits from dust and excessive heating due to hot atmosphere. To avoid heating of circuits due to atmospheric temperature white coloured composite sheet was used which helped to keep circuits cooler.

Task wise approach:

1.Autonomous Task: Rover is required to autonomously traverse between markers and identify the objects in this stage task across the terrain.Teams will be given fixed amount of time for a given stage.

Mechanical : For autonomous task we had to think of a way to setup encoders to the motors for feedback but could not mount it because of lack of space for the encoders .We also removed the manipulator as it had no purpose in the task.

2.Equipment servicing task: In this task the Rover is expected to operate on a mock-up panel and perform a specified set of operations.

Mechanical : For equipment servicing we designed different links with greater area of contact to facilitate turning knobs and flipping switches.

3.Astronaut Assistance Task: In this task teams are expected to traversed from a specified point to another collecting and delivering objects on its way . Teams are expected to moderate to extreme terrain .Teams will have up to 30 minutes to complete this task.

Mechanical : For astronaut assistance task a toolbox was designed, also links having pointed end are used to pick up the objects easily.

4.Science Task: The goal is to collect samples from selected sites in the field ,perform basic science evaluation and store at least on sample in a cache for further scientific analysis

Mechanical : For science task extra link was added in manipulator for mounting of claw. The soil collected in claw was covered using a semi-circular composite disc which was actuated using a servo motor.
For soil collection a 3-D printed hexagon was used which was sealed using an upper plate. The plate had two opening hence facilitated collection if soil and other opening for sensors. The hexagon was actuated using a vega motor. The sensor assembly was in a form of alphabet-T at each end a sensor was mounted. The whole assembly moved in vertical axis with the help of rack, pinion and slider. The sensor plate was actuated using a servo motor which rotated the plate at specific angles and making it easier and faster to collect sensor data.

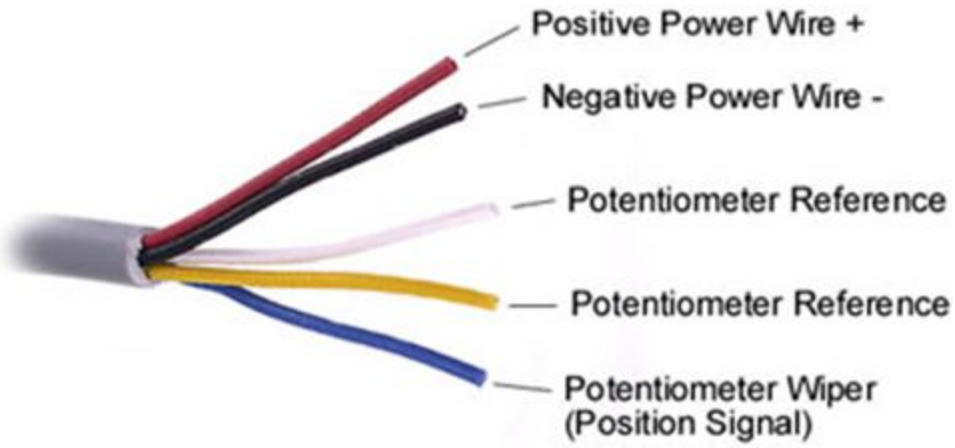
HARDWARE CODING:

LINEAR ACTUATOR:

The linear actuator consists of 5 wires:

1) RED – Motor positive terminal
2) BLACK- Motor negative terminal
3) BLUE- Feedback wire which will go to analog pin of arduino
4) YELLOW- Connected to 5 V.
5) WHITE- Connected to ground.

The actuator is a closed loop actuator, ie, feedback is provided by it in terms of resistor reading. The resistor then can be mapped to its equivalent length of actuator using arduino ‘map’ function. The speed of actuation is controlled by giving it specific pwm. Since actuator draws high current then motors so we have used ‘hercules’ motor driver as its motor driver instead of ‘cytron’.



NOTE:

Motor terminals are always shorted.

Actuation of actuator can be done by BLACK and RED wire, for feedback we require the remaining wires.

If we interchange the terminal (no need to do) yellow and white that is connect yellow to ground and white to 5V then opposite feedback will come, ie , in place of decrease it will increase and vice versa.

To get information related to closed loop actuator use the link:

https://www.youtube.com/watch?v=L5tx64G1ilQ

PWM:

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off.

Arduino PWM pins have a default frequency, but motor drivers does not always work properly on this default frequency. So for proper functioning of motor drivers we had to change its frequency. Refer the below link:

electronics.stackexchange.com/questions/287046/correct-pwm-frequency-for-motor

To change frequency of pwm pin there are 2 method:

1) By changing timer registers -in this we are changing the timer register value by diving it with a prescalar.

https://playground.arduino.cc/Code/PwmFrequency

2) By using arduino <pwm.h > library. Install library from the link

https://code.google.com/archive/p/arduino-pwm-frequency-library/downloads

INVERSE KINEMATICS:

Forward and inverse, it is like a function and its inverse. In robotics for example, this normally refers to calculate the relations between end-effectors and joint angles. So for forward kinematics, the joint angles are the inputs, the outputs would be the coordinates of the end-effectors. On the other hand for inverse kinematics, the given inputs are the coordinates of the end-effectors, the outputs to calculate are the joint angles

For basic concept: **https://drive.google.com/drive/folders/1u0796jgVVZ9bydeno6K_L0w6PPtbDDYY**

Sample code and algorithm:

https://www.circuitsathome.com/mcu/robotic-arm-inverse-kinematics-on-arduino/

(for proper explanation of algorithm see comments too given in the link)

PID ALGORITHM:

for overview about PID refer:

https://www.youtube.com/watch?v=0vqWyramGy8

Algorithm explanation:

For PID we first set a fix speed (base speed) and take 3 variables, ie, kp, ki and kd. For PID we need feedback which will give the current status and helps in error detection.

Basic Relations:

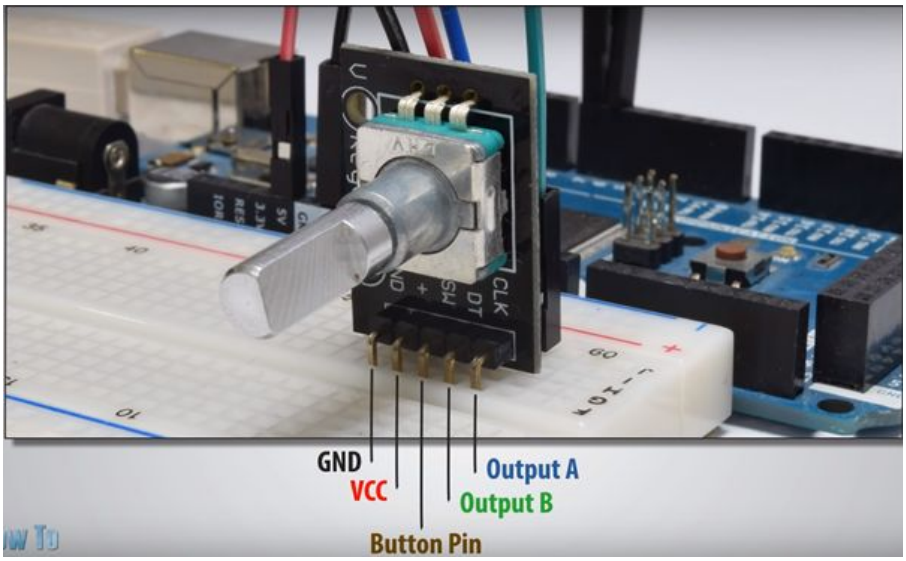
(Current_value=value which we get from feedback
Desired_value= value which we needed
D_error = derivative error
I_error= integration error
Previous_error = preceding current_value)
Error=current_value – desired_value
d_error = error - previous_error;
i_error = error + previous_error;
total_error = kp*error + kd*d_error + i_error*ki

we have to provide value to kp, ki and kd by hit and try, so it is advisable to tune with kp then kd only. If tuning doesn't takes place then go for ki.

ENCODER:

A rotary encoder is a type of position sensor which is used for determining the angular position of a rotating shaft. It generates an electrical signal, either analog or digital, according to the rotational movement.

There are 2 waves coming from encoder which are 90 degree out of phase.
Refer this video for basics: **https://www.youtube.com/watch?v=v4BbSzJ-hz4**
More the ppr (pulse per revolution) more is the efficiency of encoder.
There are total 4 connection, Vcc, ground, output and output (we didn't connect button pin).



To make battery connectors (bullet connectors)

Use this link: **https://www.youtube.com/watch?v=3oBy-jV9ajE&feature=youtu.be**

HERCULES MOTOR DRIVER:

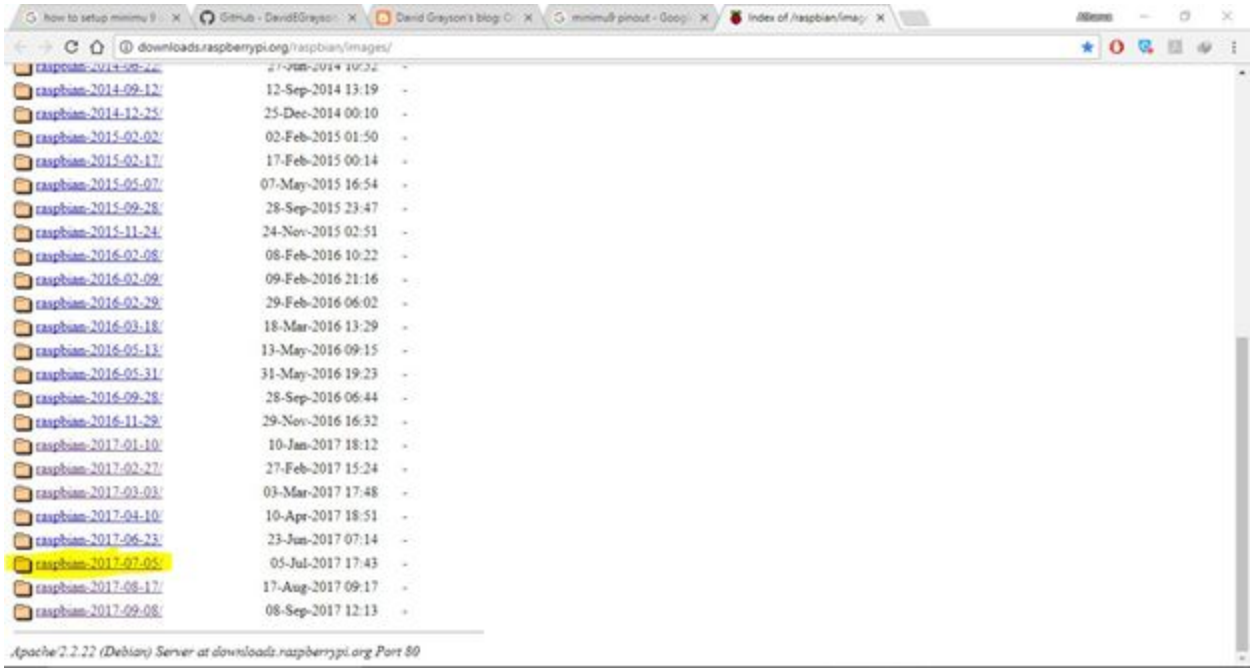
- 1st pin is ground, 2nd is direction pin1, 4th is pwm pin, 6th pin is direction pin2.
- Data sheet and basic information:

http://www.nex-robotics.com/images/downloads/Hercules%206V-36V,%2015Amp%20Motor%20V2.pdf

For all related codes refer the github link: https://github.com/pkvv/manipulator

Setting up the Raspberry Pi 3

1.Download Raspbian OS from here: https://www.raspberrypi.org/downloads/raspbian/



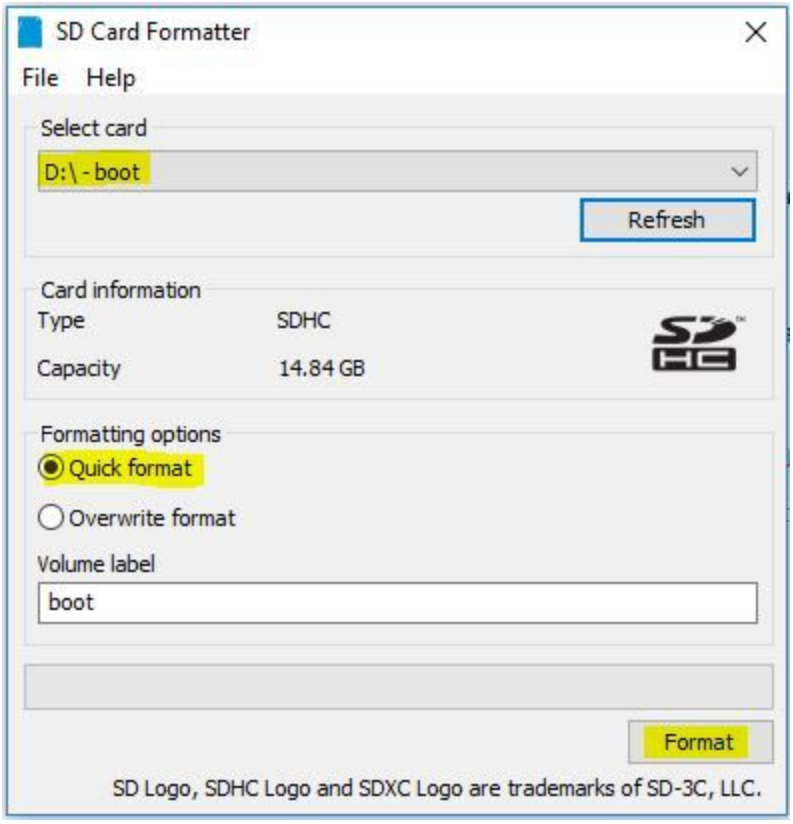
(I downloaded this one because it was the latest jessie image)

2.Download SD card Formatter from here :
https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html

3. Download win32diskimager from here : https://sourceforge.net/projects/win32diskimager/

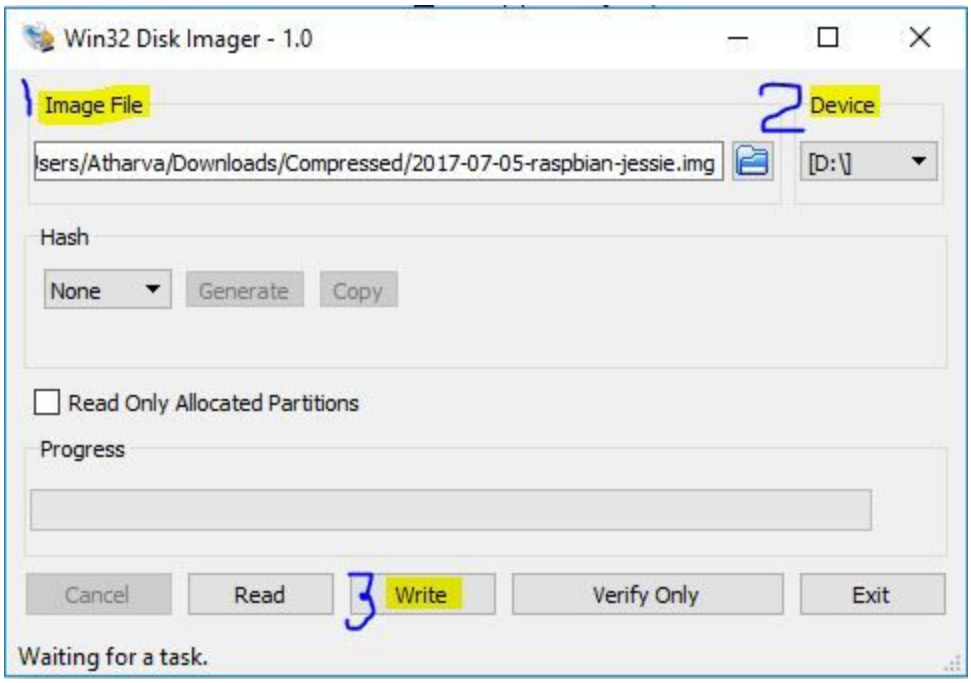
4.Download putty from anywhere. I did it from Filehippo.

5.Mount the SD card into the card reader and connect it to computer. Open SD card formatter and select the card and click “format” button.



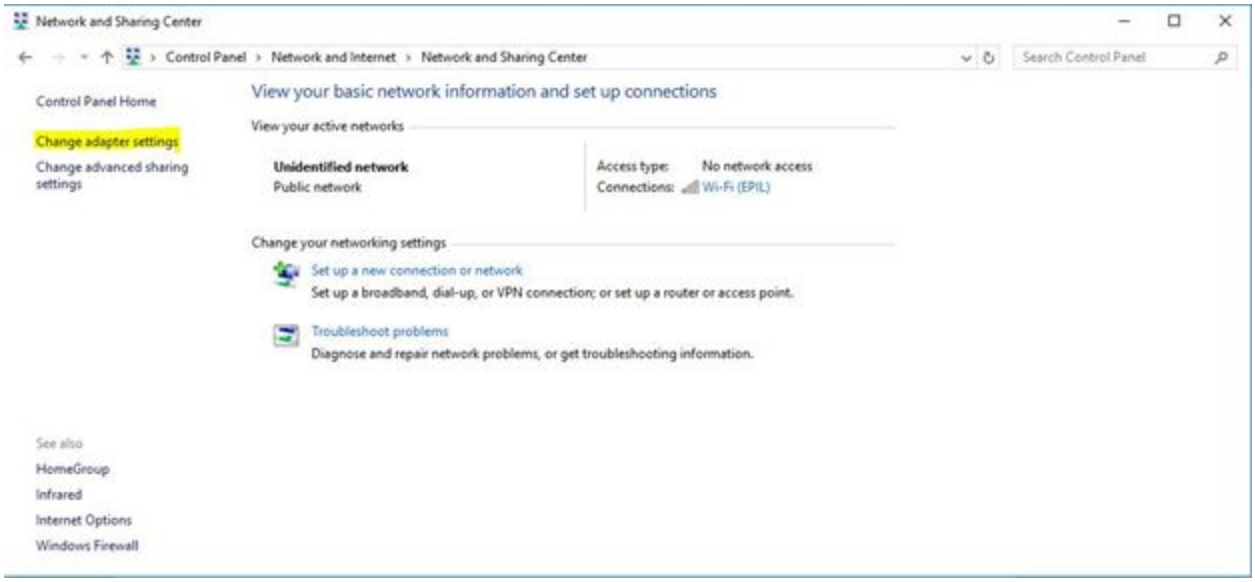
6.Now extract the Downloaded Raspbian OS

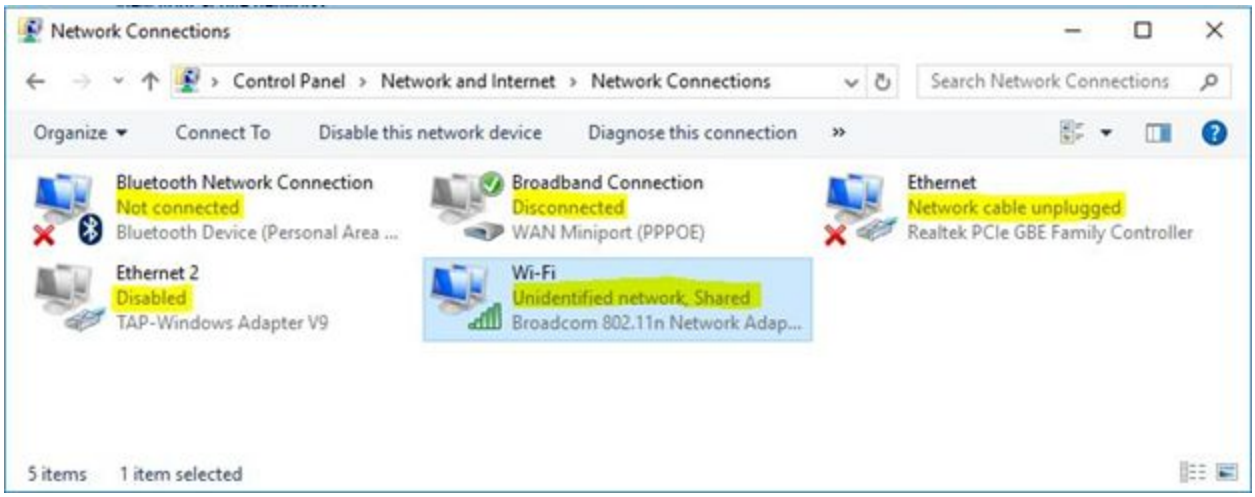
7.Open win32diskimager and browse for the extracted file and set the destination device for the SD card and click “write” button.



8.Now open windows command prompt and type the following command
echo>D:\ssh
here D = drive for SD card (in my case, this can be different for different users).
This creates a ssh file in the SD card. If we don't do this would get an error while using it from putty later on.

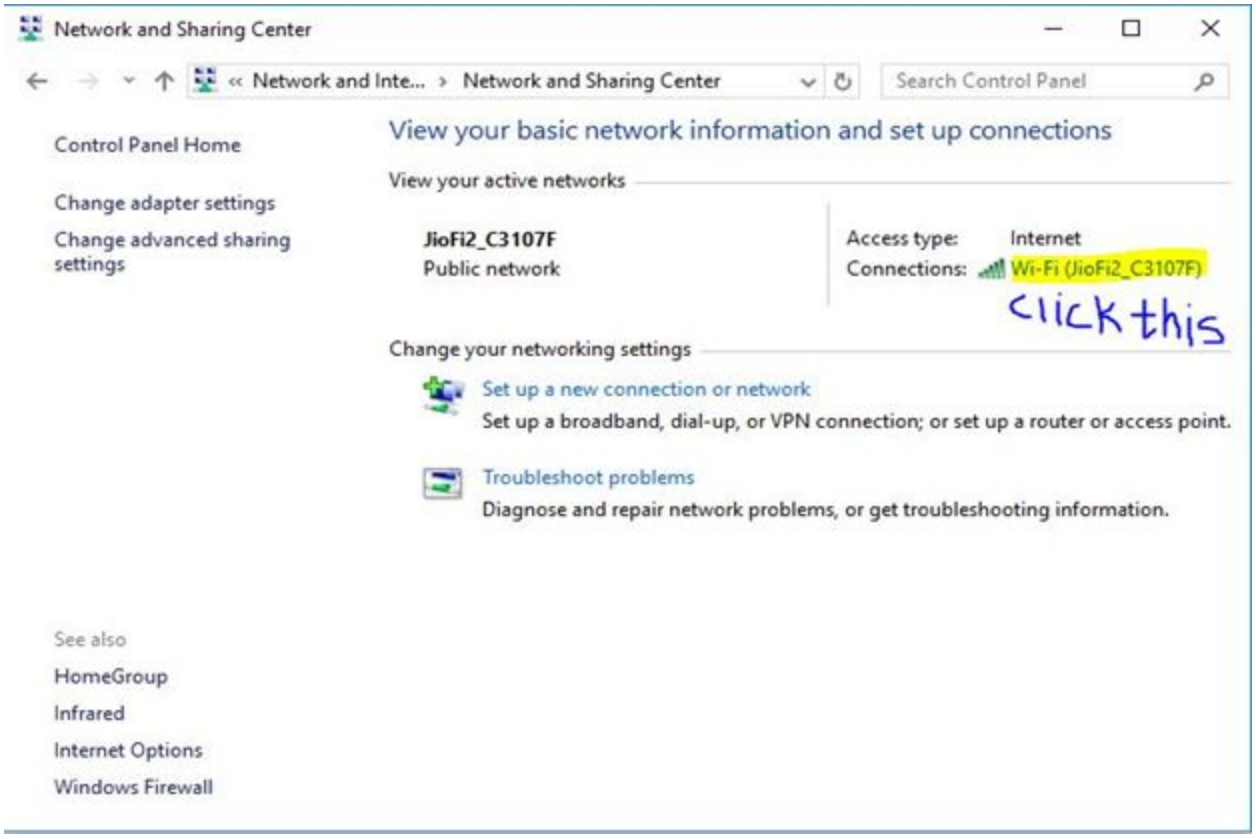
9.Eject the SD card and insert it into the Raspberry pi. And don't connect anything else to it.

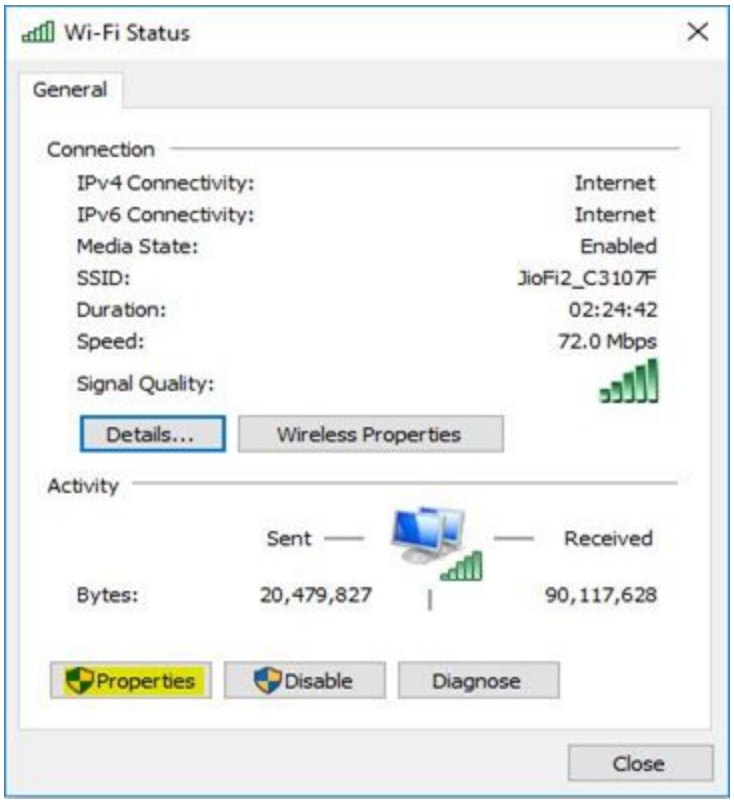


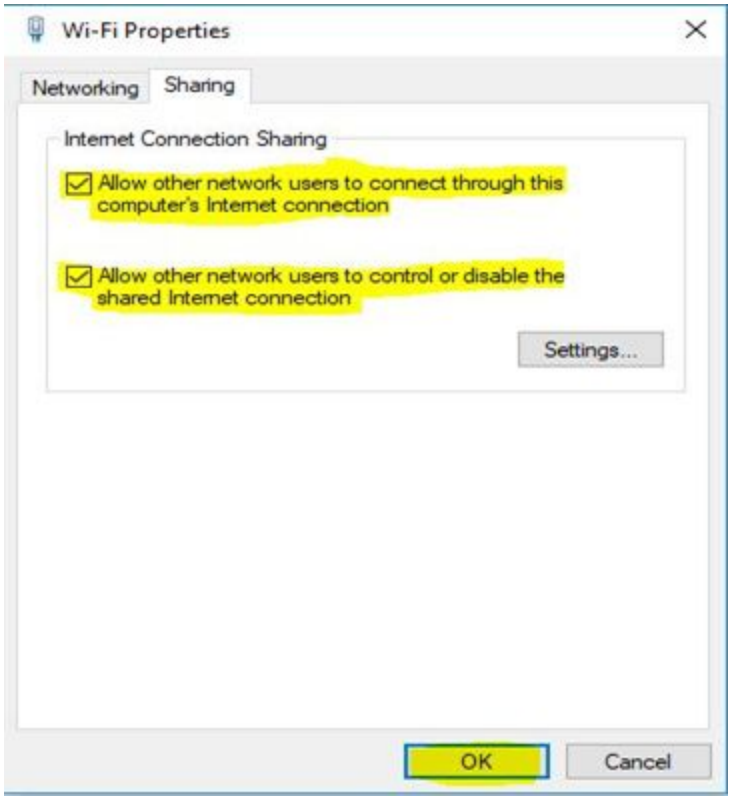


10.Now you should be connected to the internet either via WIFI or LAN. Open Network and sharing and check the adapter settings.

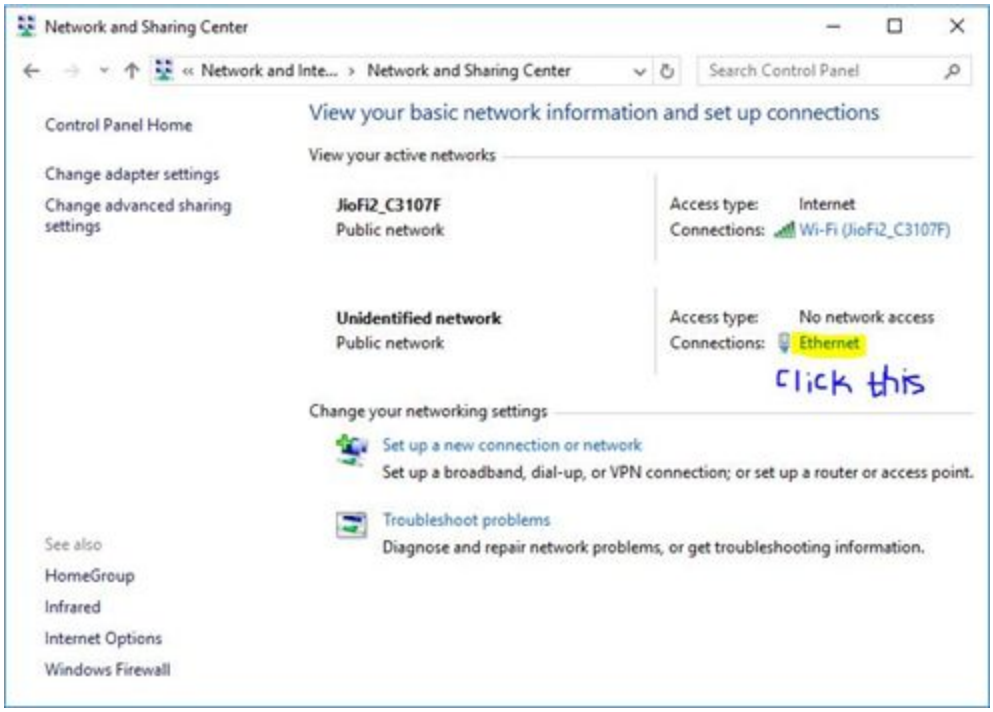
Now again go back to network and sharing and

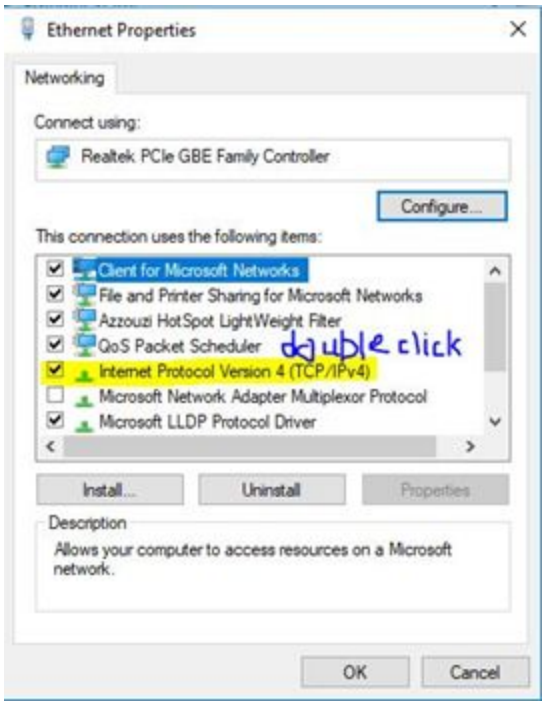


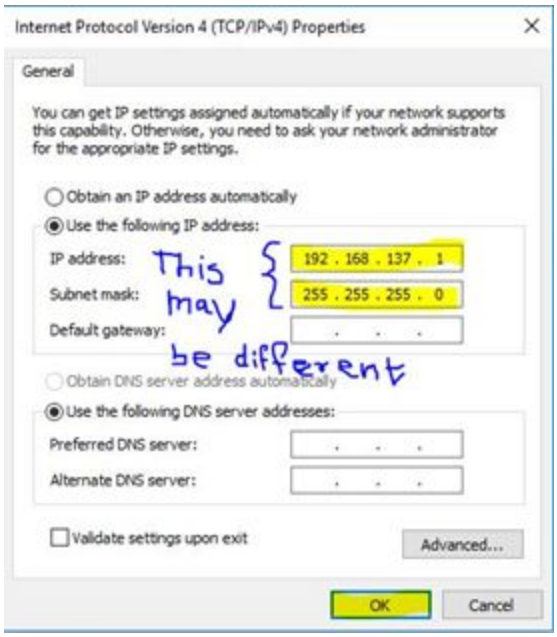




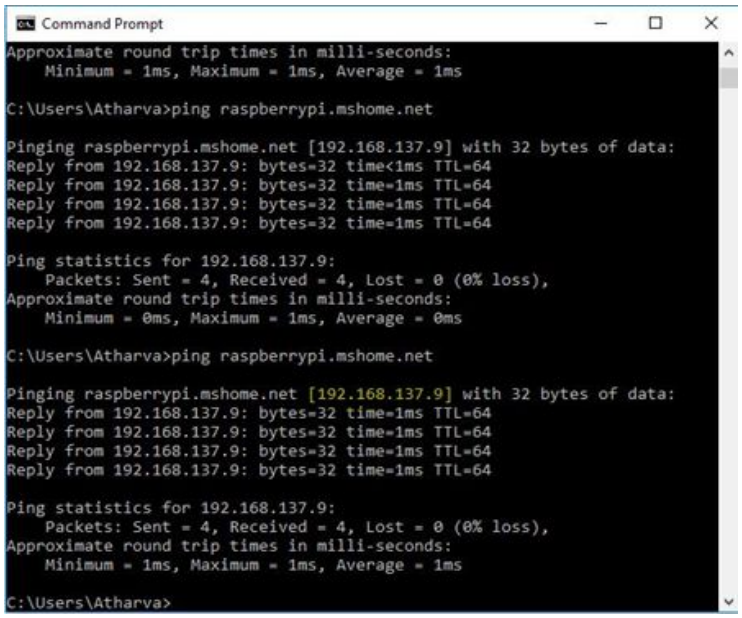
11.Insert SD card into the Raspberry Pi and now connect LAN port of laptop and Raspberry Pi with the LAN cable and power it up. Then open Network and Sharing again and follow along.







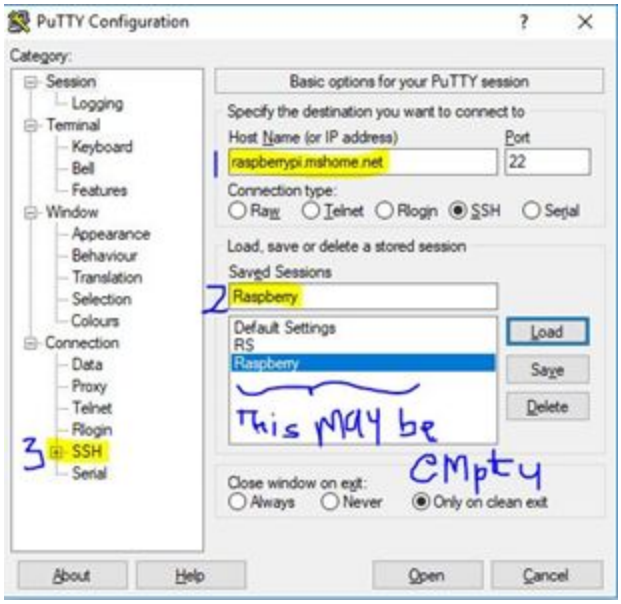
12.Now open windows command prompt and type “ping raspberrypi.mshome.net”. You will see something Like this

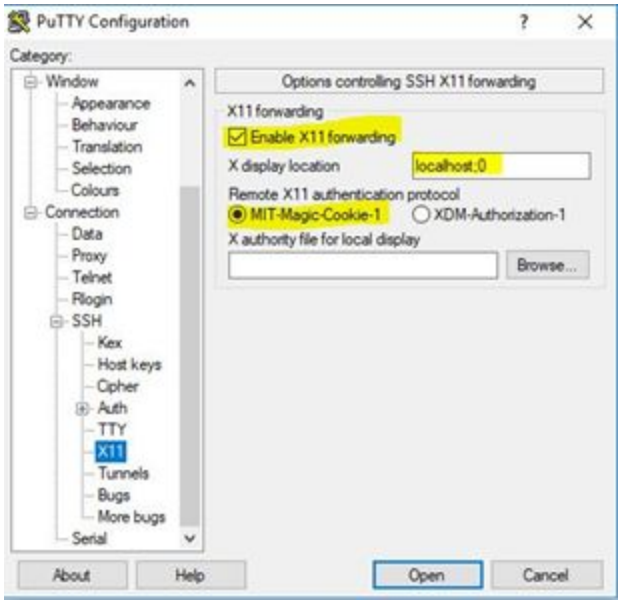


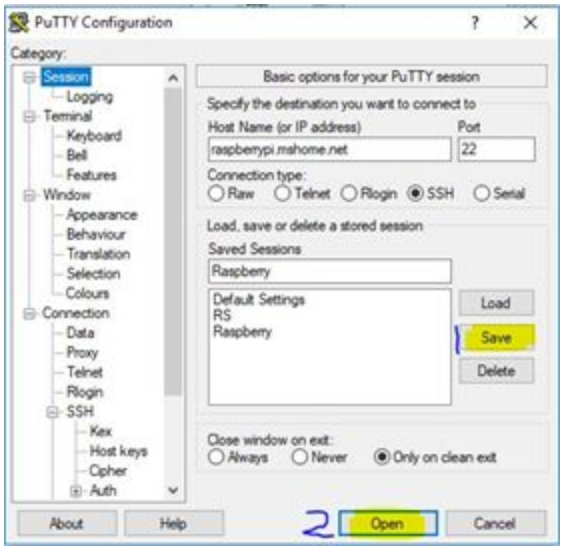
Note the highlighted IP.

13. Now download VNC viewer from here: https://www.realVNC.com/en/connect/download/viewer/ and install it.

14.Now open putty and follow along.

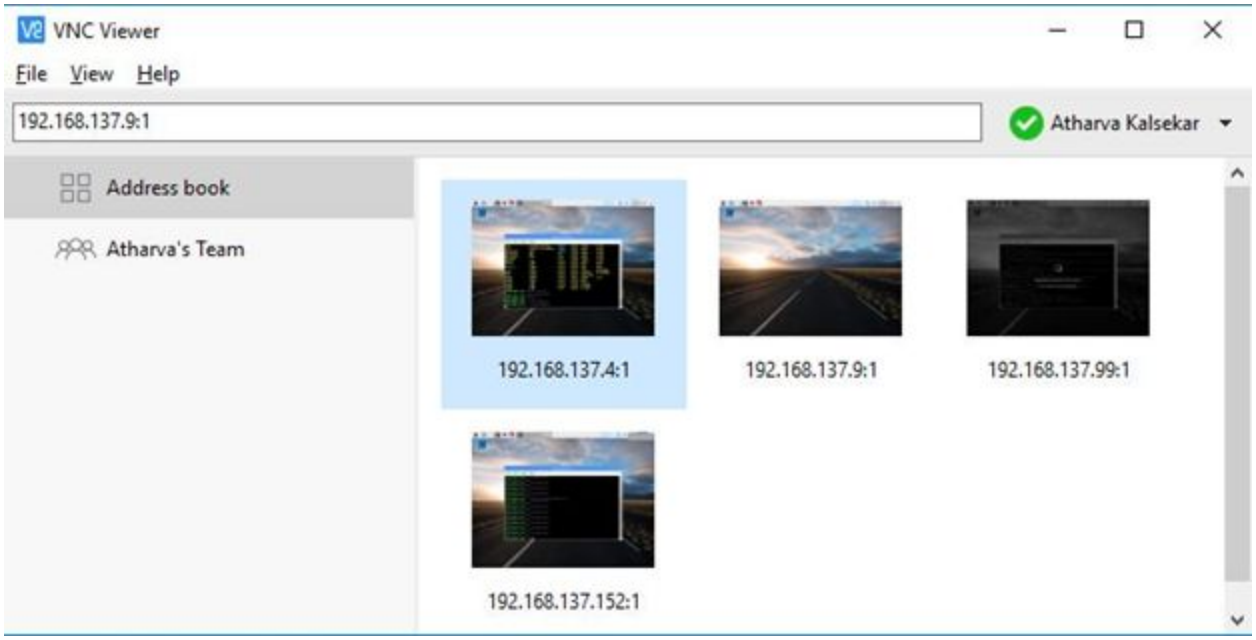






15. After that the putty session will be opened. Then **login as = pi**, **password = raspberry**.
Then type command “VNCserver“ which will give you a IP address as = <IP of pi obtained from command prompt>: 1.

16.Copy the IP of VNC server. Open VNC viewer and paste it there.
And then username = pi , password = raspberry.



NOTE: For ubuntu mate the entire procedure is same except after writing the os file connect the Raspberry pi to a hdmi screen and set up ubuntu for rpi and then enable ssh via sudo raspi-config. You can download ubuntu mate file from the ubuntu website.

Setting up wifi for ubuntu mate
Follow the steps listed in the link below:
https://gist.github.com/vfdev-5/10b9801db56e71ba17988683f863e84e

Setting up MJPG streamer in raspberry pi
Follow the steps in the link below:
https://github.com/cncjs/cncjs/wiki/Setup-Guide:-Raspberry-Pi-%7C-MJPEG-Streamer-Install-&-Setup-&-FFMpeg-Recording

Image Processing Module:

The competition included tasks like :

Object Detection : According to the guidelines, the bot should be capable of recognising some specified objects such as blue bottle,yellow disc and red box which can be present multiple at a time.To accomplish this, initially we were training our rover using the haar-cascade method, but because of the fact that we have to detect multiple objects at a time,we were unable to complete that using the same method as in that process only single positive image can be used to train the bot.The task was finally completed by using OPENCV module and the concept that each colour have its specific HSV values,which can be used to identify and differentiate between different objects.The algorithm includes capturing the frame and adjusting its HSV values according to the requirements using trackbars, and bounding the recognised object into a rectangle mentioning the object's name above the rectangle.

The following links can be followed to have a brief idea about basics of OPENCV module :

https://courses.learnopencv.com/p/opencv-for-beginners

For haar-cascading, a dataset of images is taken in which there is a probability that our processor can get confused are used as negative images and multiple positive images of the object which is to be detected, is used to train our processor using several command lines in command prompt. The process includes converting all negative images into a text file and also converting all negatives and positive ones into a XML file having parameters which can be further used by the classifier.

Wide Angle Panoramic View : For the attainment of panoramic view, we employed a method utilizing invariant feature detection based approach - ORB and finding the matches using KNN matcher, this data is then used for stitching the sampled images. The use of ORB allows robust matching of pictures irrespective of camera zoom, rotation and illumination. From feature descriptor, some key points are extracted among which four best keypoints are used to find a homography matrix between two images.Using this homography, one image is warped to be in the same perspective as the other and a new resultant image is achieved. As the images are aligned in the same frame,they are combined to form a stitched panorama image.

Recognising Digital Values : Detection of digital values is achieved by mnist_loader. It includes training our processor from an image having all values from 0 to 9 in different handwritten formats as well as in digital form. After gray-scaling the test image, various operations such as blurring the pixels, finding contours, and drawing a rectangle around the detected digit and then comparing that with the image from which the system is trained, we finally get our result above the rectangle.

Data transmission

For the continuous propagation of rover on the field, we need to monitor the motion of the rover at every instant of time. Thus, we require a module from which we can access an uninterrupted data transmission of camera feed in order to steer the bot according to the necessities.To fulfill this, we had used a receiver-transmitter module RC-832 and TS-832.

RC-832 :

The RC-832 module was the receiver module used to receive data at the base station.To visualise the camera feed, we interfaced receiver module along with EasyCAP USB 2.0 Video Capture.The EasyCAP USB 2.0 Video Capture with

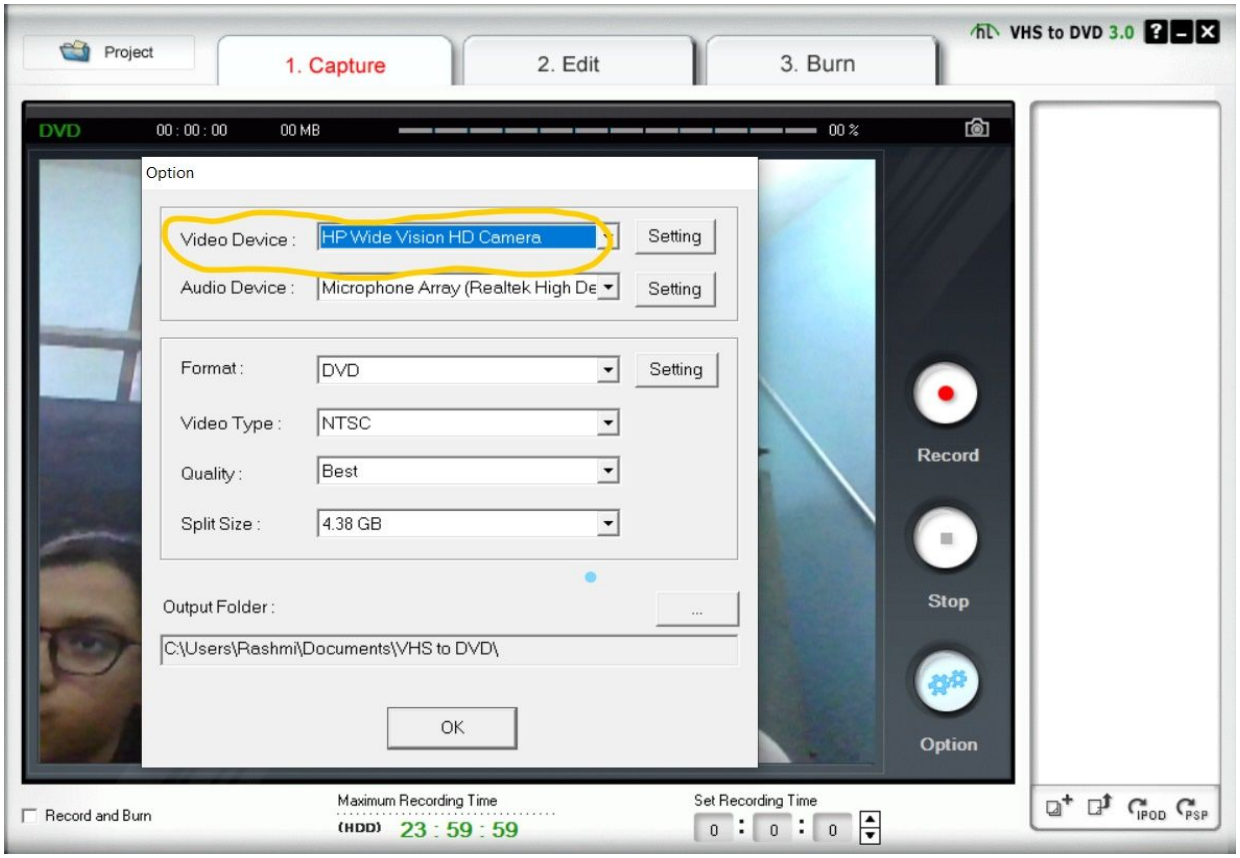
Audio, can capture High-quality video and audio file direct by USB 2.0 interface without sound card. However, the installation is very simple and the external power is unnecessary.

Installation Process :

1. Insert the CD-ROM into your computer, and select “Install Application”.

2. In the next windows, we can choose setup language, and click “next”. ”I accept the terms of the license agreement”. Click ”next” , “Finish” to complete application software installation.

3. Please double click “honestech VHS to DVD 2.0 SE” icon on your desktop, you will get a window, and you must enter your product key in the window , then click “OK” icon.(You can find the product key on the CD-ROM bag).



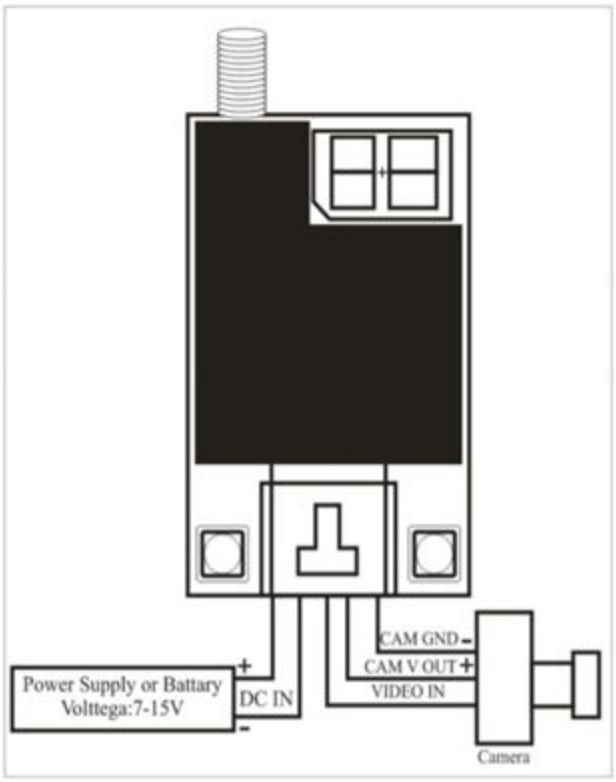
4. We should change our setting of ‘Video Device’ from front camera to ‘AVR TO USB 2’ and then press ‘OK’ to save the changes.

Technical specifications of RC-832 :

RC832 5.8GWireless receiver	Sensitivity	≤-90dBm
	Working Frequency	ISM 5.8GHz
	Available Channel	32CH
	Power supply	DC 12V
	Consumption Current	200mA, Max.
	Antenna Input Impedance	50ohm Typ.
	Antenna Connector	SMA
	Antenna Gain	2dBi
	Video Output Level	1.0Vp-p Typ, 75Ω
	Audio Output Level	1.0Vp-p Typ, 10KΩ
	Audio Carrier	6.5MHz
	Type Standard	NTSC/PAL
	Dimension (L x W x H)	80x65x15 mm
	Weight	85 g

RC-832 and EasyCAP are coupled by linking their corresponding yellow cables. We had used ‘41’ as our frequency-channel.A 12V continuous power supply is required to keep the receiver module switched ON ,which was provided by LiPO battery and can also be provided by adapter using AC supply.No such external power supply is required for EasyCAP ,as it takes power directly from the processor it is connected to.

TS-832 :



The transmitter module or TS-832 was used to transfer the data of camera from field to base station.It also requires a steady 12V supply to be in a ON condition.Care must be taken of polarity of supply while connecting the power source .The ‘CAM V OUT’ cable and ‘CAM GND’ cable are further connected to the audio jack of Raspberry-pi which is used as a processor and whose screen is accessible on the receiver side or base station.If we have to make any alteration on the screen visualised on base station, we have to setup a keyboard or mouse (according to the needs) at the transmitter side means in the processor used.

Using QT CREATOR

1. Here are the different ways to installing the QT CREATOR.
   - https://www.ics.com/blog/getting-started-qt-and-qt-creator-linux from this link you can able to install QT in your PC.
   - https://www.youtube.com/watch?v=1jdjcF7rUDU&t=185s from this link also you are able to install the QT.
   - https://media.readthedocs.org/pdf/ros-qtc-plugin-forked/4.4/ros-qtc-plugin-forked.pdf you can also be able to install the QT with ROS plugin by following the step given in above pdf.

2. If you have installed the QT creator with any of the 1st two links the to ad the ROS plugin in it you should have to do setup in your QT which is provided in the link given below https://github.com/Levi-Armstrong/ros_qtc_plugins/wiki/Setup-Qt-Creator-for-ROS .

3. In order to learn basic QT watch the following videos

- https://www.youtube.com/playlist?list=PL2D1942A4688E9D63
- https://www.youtube.com/watch?v=EkjaiDsiM-Q&list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA
- https://www.youtube.com/playlist?list=PLS1QulWo1RIZjrD_OLju84cUaUlLRe5jQ
- https://www.youtube.com/watch?v=-K-VU9I4ffY&list=PLzHdTn7Pdxs7H5w9L_tzdDU1sZ74eb0CS

These video series clear your doubts about the QT.

4. If you want to learn more about the QT you can visit the following sites

- Qt C++ GUI Development for Beginners : https://bit.ly/2DjLEWj
- Qt Quick and QML For Beginners : https://bit.ly/2RY5Kxo
- Qt Quick and QML Intermediate : https://bit.ly/2ASJxXw
- QtQuick and QML Advanced : https://bit.ly/2W1w2xU
- https://learnqtguide.teachable.com/

5. I had made the simple GUI which takes the data from the Arduino board and prints its data on the GUI. for doing this I watched the 2 videos which are

- https://www.youtube.com/watch?v=QrCbyMnoTEU
- https://www.youtube.com/watch?v=AX-HhBXBzGg&t=916s

6. The file of QT which I made is available on my GITHub account

- https://github.com/DhimanAirao/GUI

7. The error which you might get during making of this is one library is missing which is -lpulse which is rectified by downloading it from here

- https://stackoverflow.com/questions/42141004/qt-multimedia-cannot-find-lpulse

8. Now let's see how to plot the graph on the GUI. There is the widget in QT C++ known as QCustom Plot which help us to plot the graph in QT. For more idea refer the following site

- https://www.qcustomplot.com/

9. For learning how to plot the graph in QT refer the following videos.

- https://www.youtube.com/watch?v=xWGEvlDWokQ&t=2s
- https://www.youtube.com/watch?v=y9MJLUmswEM
- https://www.youtube.com/watch?v=5ENvNtYhoPM&t=2s
- https://www.youtube.com/watch?v=ST6OdlO6rRU&t=247s

10. I tried to plot the real time graph of the temperature vs time I it didn't succeeded because I got the issue of data. Because the data taken by the graph is single value by when i get the data of temperature on my gui,it is in the form of list.

11. I refer the following site to convert my data from list to single valued but It doesn't help me.
    - https://www.geeksforgeeks.org/converting-strings-numbers-cc/

12. There is the widget QWebView in QT which help you to open any type of the site in your GUI whose tutorial you can see in the series of videos provided for tutorial.

13. The problem that i get while using QWebView is that this widget is only available in version of QT Creator below 5.0 so if you want to use you must install QT version less than 5.0.

14. I found on some site that if you have version of QT 5.0 or above you can add QWebView widget by downloading some missing file I tried it but don't work in my laptop.

15. Following are the sites from where you can get the missing files
    - https://askubuntu.com/questions/831860/how-to-install-the-missing-shared-library-libqtwebkit-so-4
    - https://www.linuxquestions.org/questions/linux-software-2/qwebview-widget-missing-from-display-widgets-in-qt-creator-4175505688/
    - https://bugs.launchpad.net/ubuntu/+source/qtwebkit-source/+bug/674367

16. If you want to install Chrome browser in your linux follow the steps given in following site
    - https://askubuntu.com/questions/79280/how-to-install-chrome-browser-properly-via-command-line

Sensors

1. FC-28:

It is very simple sensor having two pins only and for getting the data from it. We use the ------ sensor to connect it with arduino to get the desired value. It gave us the two type of output digital and analog both further it depends on our use which type of data we want. The code for sensor is available in the GITHUB link provided at end.

If you want to calibrate the sensor you should visit the following site,

- https://greensense.github.io/Blog/2017/02/17/Arduino-Soil-Moisture-Sensor-Calibration/

2. MQ-135:

This is the gas sensor which simply gives us the combine reading of different gases like CO,NH3,benzene,alcohol etc. present in the atmosphere.this sensor also provides the data in two ways digital and analog.The code for sensor is available in the GITHUB link provided at end.

3. BMP180:

This sensor provide us with the data of temperature and pressure of atmosphere. This sensor works on the I2C communication mode. I had used the library for this sensor whose link is provided here

- https://github.com/sparkfun/BMP180_Breakout_Arduino_Library

The basic example codes are also provided in the library.

4. IMU(GY-87):

This sensor is having 10 degrees of freedom which are as listed below

1. 3 Accelerometer
2. 3 Gyroscope
3. 3 Magnetometer
4. Temperature and pressure sensor (BMP180)

The libraries for this sensor is available at the following site

- http://kom.aau.dk/~jdn/edu/doc/arduino/sensors/gy80gy87gy88/

5.GPS:

The GPS module that I had used is NEO-M8N the product by the ublox. The library that we had used for the same is provide in the link given below.

- https://github.com/SlashDevin/NeoGPS/blob/master/extras/doc/Installing.md
- https://github.com/SlashDevin/NeoGPS

These are two ways to use library one as explained in 1st link and secondly you git clone it and add to libraries folder.

The problem that I get during the use of GPS module is even though I keep it connected for more than Half an hour it don't provide any kind of data. The reason behind this might be that your module's firmware is not updated. The following link shows how to update firmware

- http://www.rei-labs.net/how-to-update-neo-m8n-firmware/

In order to update the firmware you need to have installed the software developed by the ublox whose link is provided below

- https://www.u-blox.com/en/product/u-center

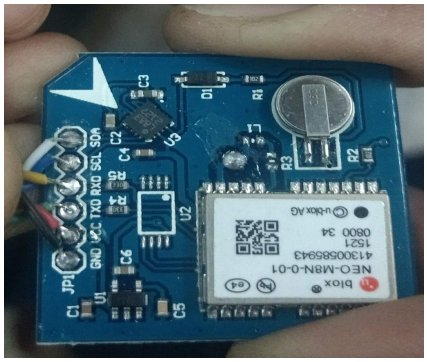
connect your GPS module with the TTL and the perform the steps provided in the above link to update firmware.

You can have the different data of your sensor using the above software and the viideos for the same is provided below

- https://www.youtube.com/watch?v=I0RaJZbccqw&t=299s
- https://www.youtube.com/watch?v=_tht-IpAALY

The pin out of NEO-M8N GPS module is as below

1. RED: VCC(+5V)
2. BLACK: GROUND
3. GREEN: TX0(TRANSMITTER)
4. YELLOW: RX0(RECEIVER)
5. BLUE: SCL
6. WHITE: SDA



The GITHUB link for all codes is given below:

- **https://github.com/DhimanAirao**

ROS(Robot Operating System):

While Installing ROS Kinetic on ubuntu 16.04 LTS following errors were encountered on various command lines.

user@user:~/kalash/ROS$ sudo apt-get update
Reading package lists... Done
E: Could not get lock /var/lib/apt/lists/lock - open (11: Resource temporarily unavailable)
E: Unable to lock directory /var/lib/apt/lists/

```
user@user:~/Ros$ sudo apt-get install ros-kinetic-desktop-full
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
```

WHILE DOING BY ANOTHER METHOD OF DOWNLOADING ROS JADE FROM GIT REPOSITORY THE PROCESS WAS COMPLETED BUT THE ROS WAS NOT COMPLETELY INSTALLED

```
user@user:~/ROS/Documents/workspace/RoundRockRobotRoundup$ cd
.git/                  Python_Sensors_Arduino/ ROS/
user@user:~/ROS/Documents/workspace/RoundRockRobotRoundup$ ros
rosco                      rosdistro_reformat
rosdistro_build_cache      rosinstall
rosdistro_freeze_source    roslocate
rosdistro_migrate_to_rep_141  rosws
rosdistro_migrate_to_rep_143

user@user:~/ROS/Documents/workspace/RoundRockRobotRoundup$ export| grep ROS
declare -x OLDPWD="/home/kalash/ROS/Documents"
declare -x PWD="/home/kalash/ROS/Documents/workspace/RoundRockRobotRoundup"
```

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. ROS is similar in some respects to 'robot frameworks.

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. These are explained in greater detail in our Conceptual Overview.

ROS is a distributed framework of processes **(aka *Nodes*)** that enables executables to be individually designed and loosely coupled at runtime. These processes can be grouped into *Packages* and *Stacks*, which can be easily shared and distributed. ROS also supports a federated system of code *Repositories* that enable collaboration to be distributed as well. This design, from the filesystem level to the community level, enables independent decisions about development and implementation, but all can be brought together with ROS infrastructure tools.

The ROS framework is easy to implement in any modern programming language. We have already implemented it in Python, C++, and Lisp, and we have experimental libraries in Java and Lua.

ROS currently only runs on Unix-based platforms. Software for ROS is primarily tested on Ubuntu and Mac OS X systems, though the ROS community has been contributing support for Fedora, Gentoo, Arch Linux and other Linux platforms.

Basic Concepts of ROS:

- **Nodes**: Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one Node provides a graphical view of the system, and so on. A ROS node is written with the use of a ROS client library, such as roscpp or rospy.
- **Master**: The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
- **Parameter Server**: The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.
- **Messages**: Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).
- **Topics**: Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by *publishing* it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will *subscribe* to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from its consumption. Logically, one can think of a topic as a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type.
- **Services**: The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply. ROS client libraries generally present this interaction to the programmer as if it were a remote procedure call.

Differential Code :

```
if __name__ == '__main__':
        rospy.init_node('JOYstick')
        twist_pub = rospy.Publisher('cmd_vel', Twist, queue_size=50)
        rospy.Subscriber('joy',Joy, keys_cb, twist_pub)
        rospy.spin()
```

In the above given code:

rospy.init_node('JOYstick')

initialises a node with name JOYstick.

twist_pub = rospy.Publisher('cmd_vel', Twist, queue_size=50)

This code fragment initialises a publisher which publishes TWIST message on cmd_vel topic with the buffer queue size of 50.

rospy.Subscriber('joy',Joy, keys_cb, twist_pub)

this code fragment initialises a subscriber which subscribes to the topic ‘joy’ having the message type Joy and as the message is received a keys_cb callback function called which publishes on the cmd_vel topic.

```
def keys_cb(msg, twist_pub):

        base=10

        #forward movement
        if msg.buttons[0]==1 and msg.axes[5]==1:
                t.linear.x=base*0.30
                t.angular.z=base*0.0
        if msg.buttons[1]==1 and msg.axes[5]==1:
                t.linear.x=base*0.7
                t.angular.z=base*0.0
        if msg.buttons[2]==1 and msg.axes[5]==1:
                t.linear.x=base*0.80
                t.angular.z=base*0.0
        if msg.buttons[3]==1 and msg.axes[5]==1:
                t.linear.x=base*0.9
                t.angular.z=base*0.0
        if msg.buttons[7]==1 and msg.axes[5]==1:
                t.linear.x=base*1.0
                t.angular.z=base*0.0

        #backward movement
        if msg.buttons[0]==1 and msg.axes[5]==-1:
                t.linear.x=-(base*0.30)
                t.angular.z=base*0.0

        if msg.buttons[1]==1 and msg.axes[5]==-1:
                t.linear.x=-(base*0.7)
                t.angular.z=base*0.0

        if msg.buttons[2]==1 and msg.axes[5]==-1:
                t.linear.x=-(base*0.80)
                t.angular.z=base*0.0
```

```
if msg.buttons[3]==1 and msg.axes[5]==-1:
        t.linear.x=-(base*0.9)
        t.angular.z=base*0.0

if msg.buttons[7]==1 and msg.axes[5]==-1:
        t.linear.x=-(base*1.0)
        t.angular.z=base*0.0



#movement left
if msg.buttons[0]==1 and msg.axes[4]==1:
        t.angular.z=(base*0.20)
        t.linear.x=(base*0.80)

if msg.buttons[1]==1 and msg.axes[4]==1:
        t.angular.z=(base*0.80)
        t.linear.x=(base*0.20)

if msg.buttons[2]==1 and msg.axes[4]==1:
        t.angular.x=(base*0.20)
        t.linear.z=base*0.70

if msg.buttons[3]==1 and msg.axes[4]==1:
        t.angular.z=(base*0.70)
        t.linear.x=base*0.20

if msg.buttons[7]==1 and msg.axes[4]==1:
        t.angular.z=-(base*0.8)
        t.linear.x=0


#movement right
if msg.buttons[0]==1 and msg.axes[4]==-1:
        t.angular.z=-(base*0.20)
        t.linear.x=base*0.80

if msg.buttons[1]==1 and msg.axes[4]==-1:
        t.angular.z=-(base*0.80)
        t.linear.x=base*0.20

if msg.buttons[2]==1 and msg.axes[4]==-1:
        t.angular.z=-(base*0.20)
```

```
        t.linear.x=base*0.70

    if msg.buttons[3]==1 and msg.axes[4]==-1:
        t.angular.z=-(base*0.70)
        t.linear.x=base*0.20

    if msg.buttons[7]==1 and msg.axes[4]==-1:
        t.angular.z=base*0.8
        t.linear.x=0


    #stop code
    if msg.buttons[8]==1:
        t.angular.z=0
        t.linear.x=0

    twist_pub.publish(t)
```

For further codes the following link can be approached,

https://drive.google.com/open?id=1s0ix1qBA3BQyxmsHQxfKNQ7gmzO01ik0

MAPVIZ:

Mapviz is a ROS-based visualization tool with a plug-in system.

You can install mapviz using apt-get from the ROS apt repository:

```
sudo apt-get install ros-$ROS_DISTRO-mapviz ros-$ROS_DISTRO-mapviz-plugins
ros-$ROS_DISTRO-tile-map ros-$ROS_DISTRO-multires-image
```

The source code we have used for mapviz is:

`https://github.com/swri-robotics/mapviz.git`

Mapviz uses various plugins like coordinate picker,laser_scan etc.The details of various plugin can be referred from above github link.

The plugins which we used were tile_map , coordinate picker and gps.The plugin coordinate picker was used for taking coordinates. GPS plugin was used for taking the gps coordinates on map and tile_map was used for generating the desired map on mapviz.We have generated the offline google map on mapviz.
For generating the map we used the following link:
https://github.com/danielsnider/MapViz-Tile-Map-Google-Maps-Satellite

The problem we faced was taking gps coordinates on mapviz and it was resolved by setting up proper resolution. The second problem faced by us was displaying the google map on mapviz.For this issue the following link was referred: https://github.com/danielsnider/MapViz-Tile-Map-Google-Maps-Satellite/issues/3

RVIZ:
Rviz is a 3D visualization tool in ROS.
For this following documentation was referred:
https://github.com/ros-visualization/rviz
The URDF file used for visualization was exported from solidworks.