eYSIP2016

Autonomous Drone

Akshit Gandhi
Keyur Rakholiya
Mentors:
Pushkar Raj
Akshat Jain
Rama Kumar

Duration of Internship: 21/05/2016 – 10/07/2016

2016, e-Yantra Publication

Autonomous Drone

Abstract

The latest innovation in the aviation industry is the Drone Technology. But, the drones that require manual input are very susceptible to radio interference or radio connection losses which can lead to disastrous results with your drone, such as crashes, etc. So, we came up with a solution, the solution is to make it **Autonomous!** i.e no manual input.

The drone has a companion computer on-board (Raspberry Pi) which guides the drone. These kind of drones can be used for security surveillance, perimeter defense, mapping unreachable or hard to reach sites and even archaeological sites!, aerial photography, military purposes, forest survey, etc.

So, as far as the project is concerned the drone we have designed has the following key features.

- Autonomous Take-off and Land
- Manual control using keyboard via remote login with Raspberry Pi
- Autonomous Take-off and Land (indoors) with ultrasonic sensors
- Autonomous flight between multiple points

Completion status

The project has been **completed** successfully. The tasks which we have accomplished are noted below.

1. Calculations for deciding the components to be used for drone.
2. Assembly of Drone.

3. Calibrating APM.
4. First Manual Test Flight.
5. Interfacing GPS with APM and reading values from it.
6. Interfacing Ultrasonic Sensor with APM and reading values from it.
7. Setting up the Raspberry Pi.
8. Create a Power distribution system for the drone, RPi and APM.
9. **Interfacing the Raspberry Pi with APM.**
10. Manual flight using Raspberry Pi (remote login).
11. Auto take-off and landing using Raspberry Pi.
12. Semi-Autonomous Flight.
13. Completely Autonomous Flights.

1.1 Hardware parts

- F450 Frame: Datasheet, Vendor link.
- 30 A ESC (DYS): Datasheet.
- D2282 1100 Kv Brushless Motors (DYS): Datasheet.
- 10x4.5 Proppellers 2 Clockwise and 2 Counter-clockwise rotation.
- Anti-vibration mount for APM 2.6.
- Fly-Sky CT-6B Transmitter Receiver for manual flight: Datasheet.
- Ardupilot flight controller v2.6: Datasheet, Official Documentation
- Raspberry Pi B+: Datasheet.
- Lithium Polymer 3 Cell, 11.1V, 5000mAh, 20C discharge Battery: Datasheet and Vendor.
- GPS Module: Datasheet.
- Ultrasonic Sensor HC-SR04: Datasheet.
- Connection diagram

1.2 Software used

1. Ardupilot Copter Firmware

   This is a firmware for the APM 2.6 board.

   - Detail of software: ACv3.2.1 and it can be downloaded from the mission planner itself or follow the link.
   - To load the firmware follow the tutorials section of Autonomous Drone or follow this link

2. Mission Planner (Ground Station System)

   Details: Mission Planner is a ground control station for Plane, Copter and Rover. It is compatible with Windows only. Mission Planner can be used as a configuration utility or as a dynamic control supplement for your autonomous vehicle.

   - Load the firmware (the software) into the autopilot (APM, PX4...) that controls your vehicle.
   - Setup, configure, and tune your vehicle for optimum performance.
   - Plan, save and load autonomous missions into you autopilot with simple point-and-click way-point entry on Google or other maps.
   - Download and analyze mission logs created by your autopilot.
   - Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.
   - Detail of software: **Version 1.3.30**, Download link
   - Installation steps:
     (a) Download the software from above link.
     (b) Follow the detailed installtion procedure on this link and also you can visit the documentation on this link

3. MobaXterm (For SSH with Raspberry Pi)

   Details: MobaXterm is your ultimate toolbox for remote computing. In a single Windows application, it provides loads of functions that are tailored for programmers, webmasters, IT administrators and pretty much all users who need to handle their remote jobs in a more simple fashion.

MobaXterm provides all the important remote network tools (SSH, X11, RDP, VNC, FTP, MOSH, ...) and Unix commands (bash, ls, cat, sed, grep, awk, rsync, ...) to Windows desktop, in a single portable exe file which works out of the box.

- Detail of software: Version 9.1 Home Edition, Download link
- Installation steps:
  (a) Download MobaXterm from website link above.
  (b) Launch MobaXterm installer.
  (c) Click Next to start installation. Accept the license agreement and click Next.
  (d) Choose installation folder, then click Next. Then click Install.
  (e) After the installation, click Finish.
  (f) To create a session, Open MobaXterm
  (g) Click Session to create a new session.
  (h) Configure the new session for user root.

1.3 Assembly of hardware

1.3.1 Assembly for manual flight

Assembling the Motors

Once you get the motors, the very first job is to mount the motors on the F450 frame. For this you will need M3 screws, so with the help of a screw-driver or L-key mount the motors on the arm of the quad like this:

Figure 1.1: installing motors

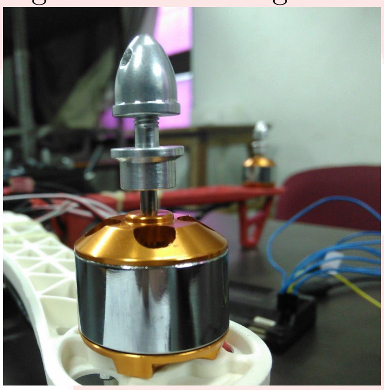


Figure 1.2: Motor Assembly.

Esc mount

After the motors have been attached on the arms, next job is to attach the esc's, you can tie individual esc's on the arms using cable ties as shown below.

Also connect 3 wires from the esc's to the 3 wires on the motor, don't worry about the order in which the wires have to be connected we will look upon that in later tutorials.on ce your arms are ready with the escs and motors it will look something like this:

Figure 1.3: Esc Assembly.

Connecting Esc to Power distribution board

The F450 frame comes with an inbuilt power distribution board so you just need to solder the esc wires (both positive and ground) to the pcb on the base plate. The pcb would look something like this.

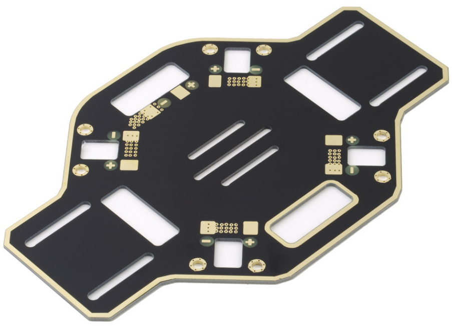


Figure 1.4: Power distribution Board.

You will see that there are some points to attach the esc's marked as + or - and also on extra pair of points will be there for connecting the battery to the distribution board. First you need to ensure that there are no connectors on the esc(on the +ve and -ve wires), if there are any connectors desolder them and then solder the wires onto the pcb. Special care to ensure that +ve wires is soldered onto the +ve of the pcb and respectively for negative.

Once you have soldered the wires onto the assembly would look something like this:

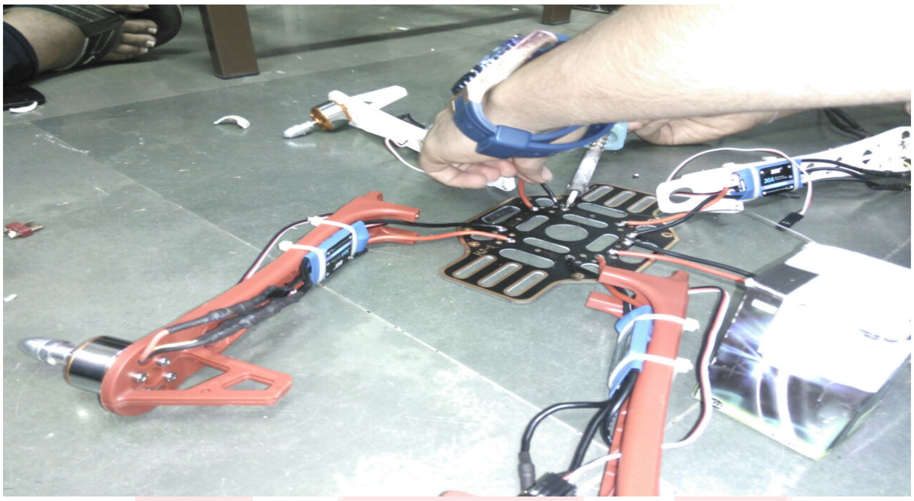


Figure 1.5: Soldering.

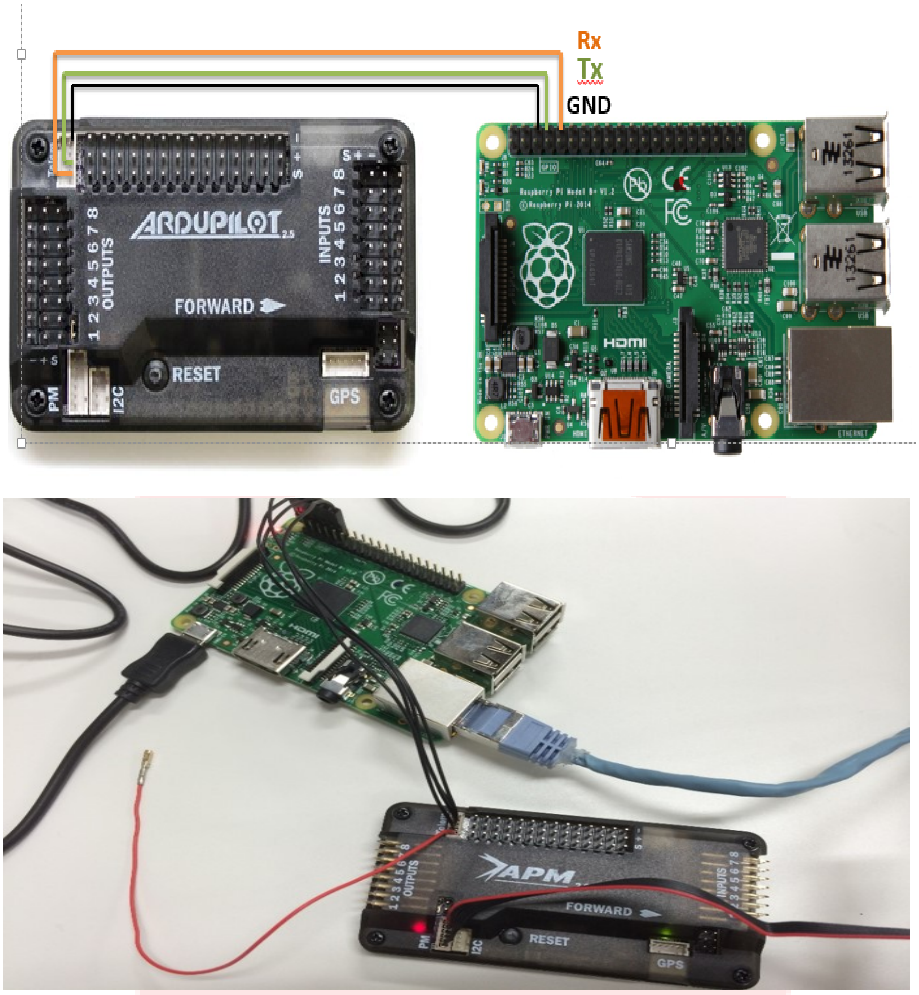


Figure 1.6: Wiring Schematic.

Assembling the frame together

Now you have your frame ready the next step is to use the screws provided with the kit to mount the arms on the lower plate and after that attach the upper plate. After this step your quad should look like this:

But an IMPORTANT note: To make it easy for you to know the orientation of the quad you can setup the frame in such a way that 2 adjacent white/red arms point to the forward direction of the quad.

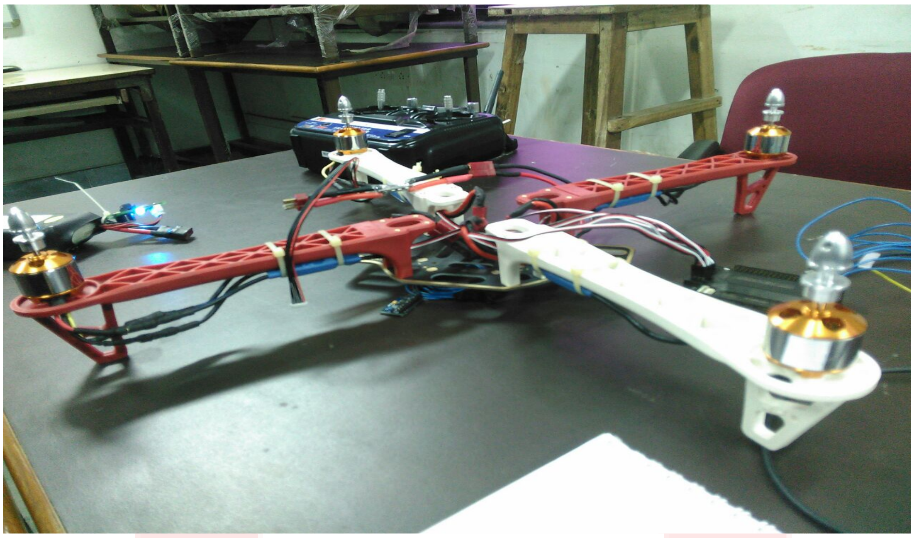


Figure 1.7: Frame Assembly w/o upper plate mounted.

Mounting your Flight Controller

For the autonomous drone application we are usin APM 2.6 as our flight controller, but you can use any flight controller of your choice like KK 2.x board, CC3D, Naze 32, Pixhawk, KISS,etc. The basic step to mount any flight controller on the deck of the frame is by mounting it by using an anti-vibration mount. Using it is very important as it reduces the amount of vibrations that reach the controller. As unwanted vibrations can lead to erroneous results.

So were are using the antivibration mount for the APM. So depending on your flight controller you can mount it on the mount by using double sided tapes. But before mounting the vibration mount on the deck ensure that the forward direction of the FC matches with the forward direction of your quad decided by you.

Wiring the flight controller

Now depending on your flight controller you have to wire up your Flight controller with the receiver and the ESC's. Here we will demonstrate the connections for the APM 2.6 .

Another important note is to make the motor connections properly i.e. the first motor connects to port 1 on the APM and son on. The below pictures will make it very clear.

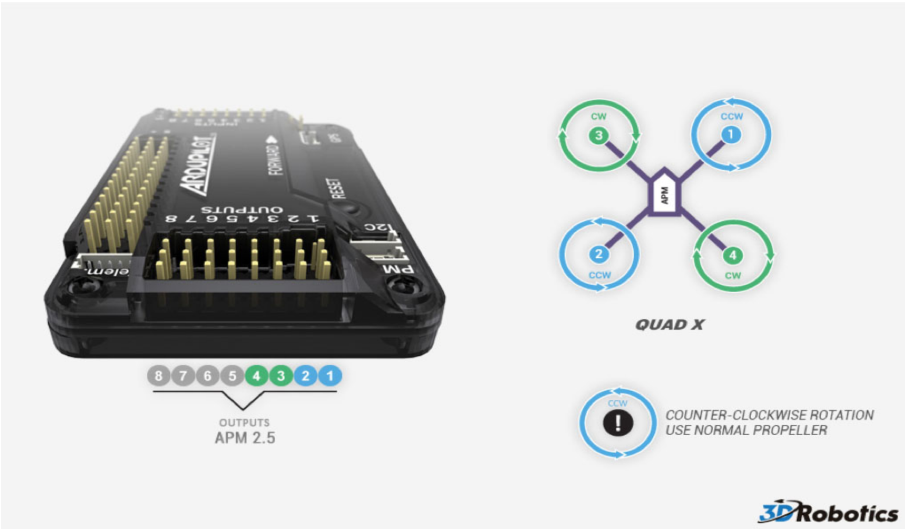


Figure 1.8: Motor Layout For quad X config.

Don't worry about the motor spin direction we will solve it in a moment. Now let's see how to make connections with the APM. Refer the below figure. Don't bother about the GPS and Gimbal, they are just extra accessories.

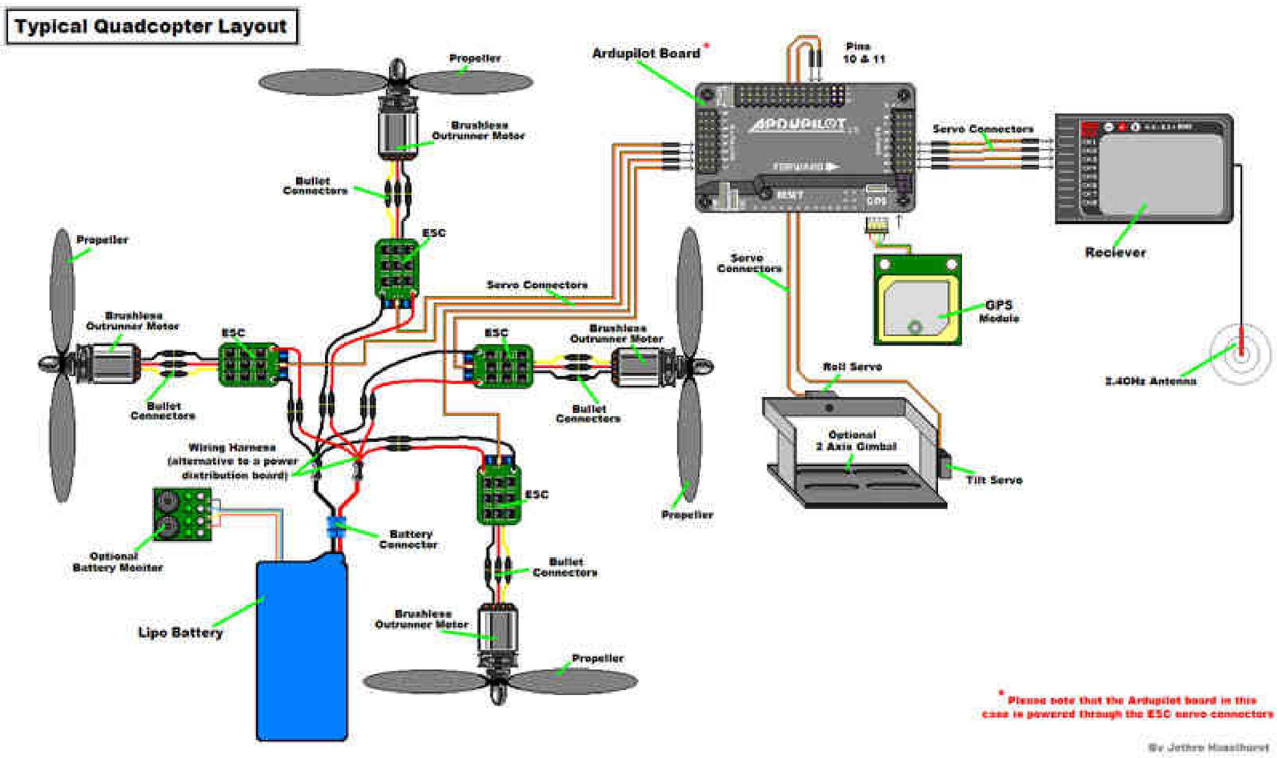


Figure 1.9: APM wiring

Now if you have followed the above steps properly, you are ready for the calibration of your flight controller. Now here we are done with the assembly.

Note: for calibration of APM refer to the tutorials on autonomous drone repository on this link

1.3.2 Assembly for Autonomous flight

This section explains how to connect and configure a Raspberry Pi (RPi) so that it is able to communicate with a APM flight controller using the MAVLink protocol[2] over a serial connection

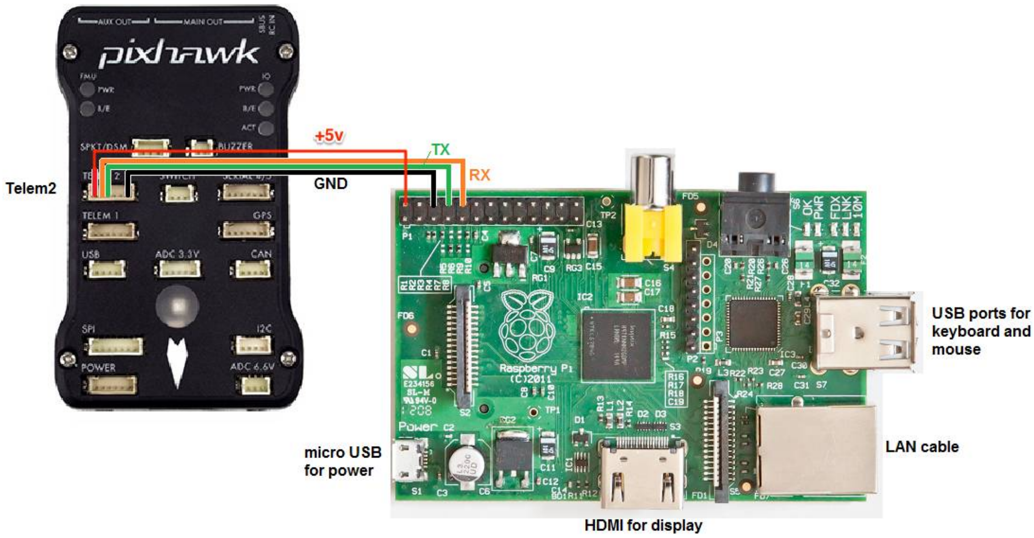


Figure 1.10: Connection diagram.

The picture shown above is for pixhawk controller but the same connection works for APM also.

Connecting to RPi with an SSH/Telnet client

NOTE: These steps assume that you have already set-up your RPi so that it is running Raspbian.

To avoid the requirement to plug a keyboard, mouse and HDMI screen into your RPi it is convenient to be able to connect from your Desktop/Laptop computer to your RPi using an SSH/Telnet client such as PuTTY.

- Connect the RPi to your local network by one of the following methods:
  - Connecting an Ethernet LAN cable from the RPi board to your Ethernet router, or
  - Use a USB dongle to connect your RPi to the local wireless network, or
  - Connect the Ethernet LAN cable between the RPi and your desktop/laptop computer. Open the control panels Network and Sharing Center, click on the network connection through which your desktop/laptop is connected to the internet, select properties and then in the sharing tab, select Allow other networks to connect through this computers Internet connection

- Determine the RPis IP address:
  - If you have access to the RPis terminal screen (i.e.you have a screen, keyboard, mouse connected) you can use the /sbin/ifconfig command.

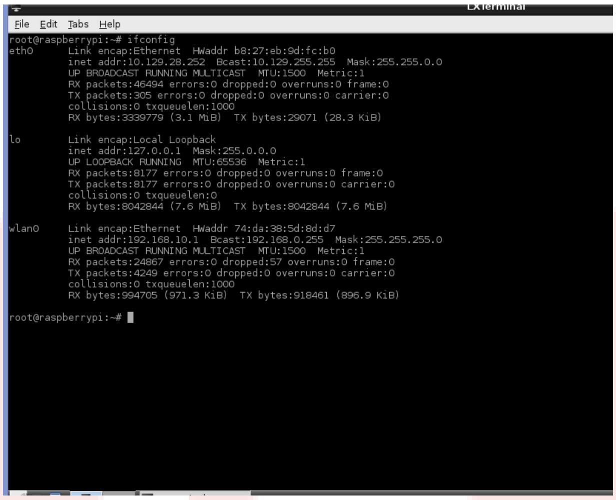


Figure 1.11: IP address on terminal.

  - If your Ethernet router has a web interface it may show you the IP address of all connected devices

- Connect with Putty:

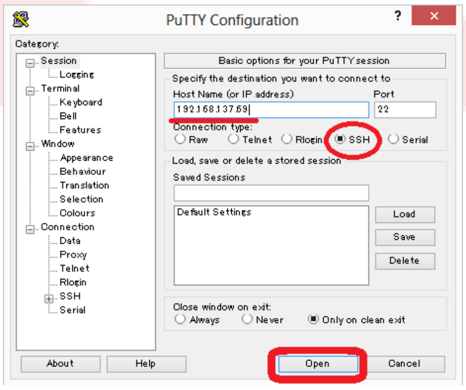


Figure 1.12: Connect screen with Putty.

If all goes well you should be presented with the regular login prompt to which you can enter the username/password which defaults to pi/raspberry

Install the required packages on the Raspberry Pi

Log into the RPi board (default username password is pi/raspberry) and check that its connection to the internet is working. If there's some problem then troubleshooting guide can be found on the documentation site http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html

After the internet connection is confirmed to be working install these packages:

- sudo apt-get update
- sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv python-pip python-numpy python-dev libxml2-dev libxslt-dev
- sudo pip install pymavlink
- sudo pip install mavproxy

Disable the OS control of the serial port

Use the Raspberry Pi configuration utility for this.

Type:

- sudo raspi-config
- And in the utility, select Advanced Options:

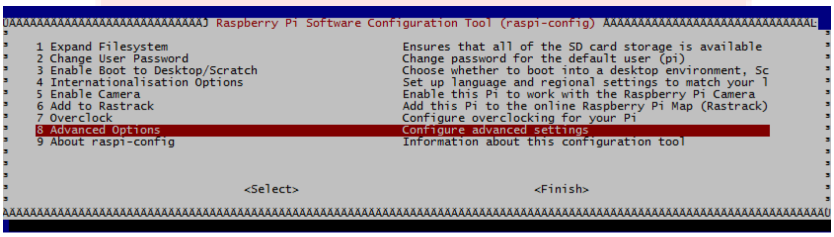
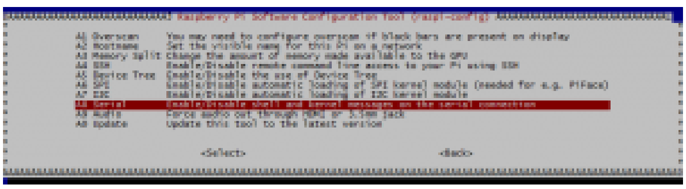


Figure 1.13: RasPiConfiguration Utility: Serial Settings: Advanced Options.

  - And then Serial to disable OS use of the serial connection:



Reboot the Raspberry Pi when you are done.

Testing the connection

To test the RPi and APM are able to communicate with each other first ensure the RPi and APM are powered, then in a console on the RPi type:

- sudo -s
- mavproxy.py –master=/dev/ttyAMA0 –baudrate 57600 –aircraft My-Copter

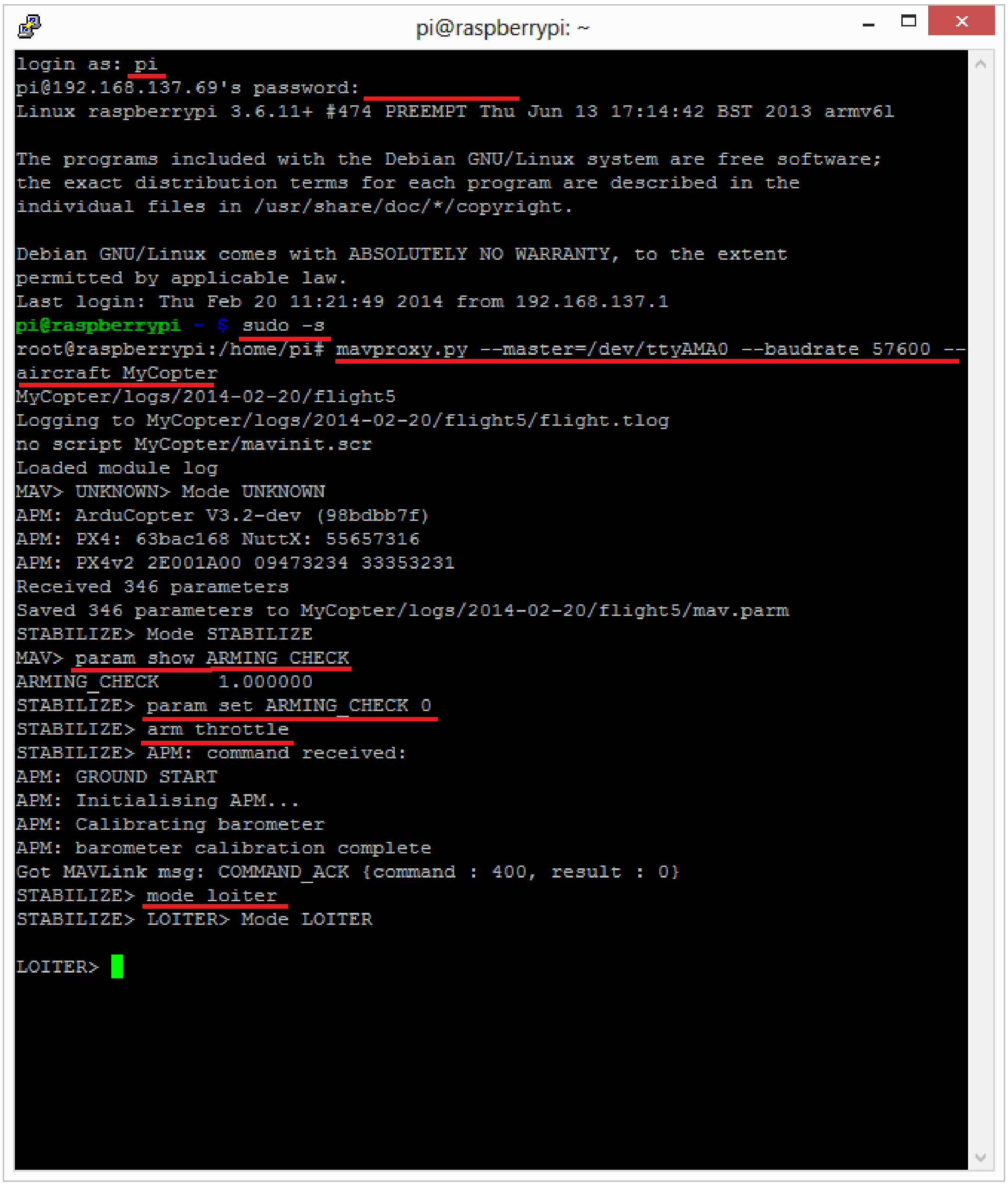
  raspberry pi's default port is connected with console input/output. so use this port we have to disable it first. now we have connected our device via UART port ot R-Pi. so "/dev/ttyAMA0" is a port for serial communication between R-pi and APM.

  You configure BaudRate as bits per second. The transferred bits include the start bit, the data bits, the parity bit (if used), and the stop bits. However, only the data bits are stored.

  The baud rate is the rate at which information is transferred in a communication channel. In the serial port context, "9600 baud" means that the serial port is capable of transferring a maximum of 9600 bits per second. If the information unit is one baud (one bit), the bit rate and the baud rate are identical. If one baud is given as 10 bits, (for example, eight data bits plus two framing bits), the bit rate is still 9600 but the baud rate is 9600/10, or 960. You always configure BaudRate as bits per second. Therefore, in the previous example, set BaudRate to 9600.

  Note: In the above command it's MyCopter and My-Copter

- Once MAVProxy has started you should be able to type in the following command to display the `ARMING_CHECK` parameters value
  - param show `ARMING_CHECK`
  - param set `ARMING_CHECK` 0
  - arm throttle
    * And then Serial to disable OS use of the serial connection:



```
pi@raspberrypi: ~
login as: pi
pi@192.168.137.69's password:
Linux raspberrypi 3.6.11+ #474 PREEMPT Thu Jun 13 17:14:42 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 20 11:21:49 2014 from 192.168.137.1
pi@raspberrypi ~ $ sudo -s
root@raspberrypi:/home/pi# mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --
aircraft MyCopter
MyCopter/logs/2014-02-20/flight5
Logging to MyCopter/logs/2014-02-20/flight5/flight.tlog
no script MyCopter/mavinit.scr
Loaded module log
MAV> UNKNOWN> Mode UNKNOWN
APM: ArduCopter V3.2-dev (98bdbb7f)
APM: PX4: 63bac168 NuttX: 55657316
APM: PX4v2 2E001A00 09473234 33353231
Received 346 parameters
Saved 346 parameters to MyCopter/logs/2014-02-20/flight5/mav.parm
STABILIZE> Mode STABILIZE
MAV> param show ARMING_CHECK
ARMING_CHECK      1.000000
STABILIZE> param set ARMING_CHECK 0
STABILIZE> arm throttle
STABILIZE> APM: command received:
APM: GROUND START
APM: Initialising APM...
APM: Calibrating barometer
APM: barometer calibration complete
Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
STABILIZE> mode loiter
STABILIZE> LOITER> Mode LOITER

LOITER>
```

Figure 1.14: Terminal screen.

NOTE: If you get an error about not being able to find log files or if this example otherwise doesnt run properly, make sure that you havent accidentally assigned these files to another username, such as Root.

Entering the following at the Linux command line will ensure that all files belong to the standard Pi login account:

* sudo chown -R pi /home/pi

Configure MAVProxy to always run

To setup MAVProxy to start whenever the RPi is restarted open a terminal window and edit the /etc/rc.local file, adding the following lines just before the final exit 0 line:

```
(
date
echo $PATH
PATH=$PATH:/bin:/sbin:/usr/bin:/usr/local/bin
export PATH
cd /home/pi
screen -d -m -s /bin/bash mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft MyCopter
) > /tmp/rc.log 2>&1
exit 0
```

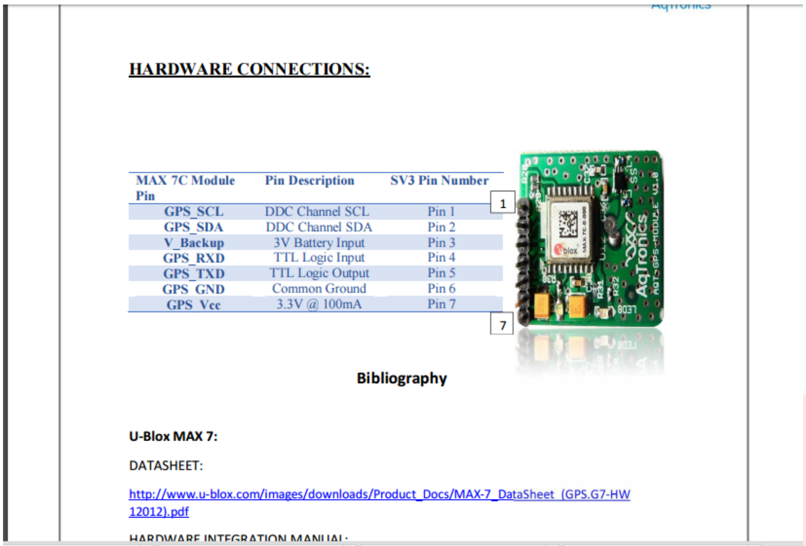
To open /etc/rc.local file type in sudo nano /etc/rc.local and type the code below and save it in the file

If you wish to connect to the MAVProxy application that has been automatically started you can log into the RPi and type: sudo screen -x

Note: For powering the Raspberry Pi from the drone refer this.

1.3.3 Interface GPS sensor with APM 2.6

To connect the GPS to APM via GPS port which is given in APM. Follow this image for connections.



HARDWARE CONNECTIONS:

MAX 7C Module Pin	Pin Description	SV3 Pin Number
GPS_SCL	DDC Channel SCL	Pin 1
GPS_SDA	DDC Channel SDA	Pin 2
V_Backup	3V Battery Input	Pin 3
GPS_RXD	TTL Logic Input	Pin 4
GPS_TXD	TTL Logic Output	Pin 5
GPS_GND	Common Ground	Pin 6
GPS_Vcc	3.3V @ 100mA	Pin 7

Bibliography

U-Blox MAX 7:

DATASHEET:

http://www.u-blox.com/images/downloads/Product_Docs/MAX-7_DataSheet_(GPS.G7-HW 12012).pdf

Figure 1.15: GPS module pin Diagram

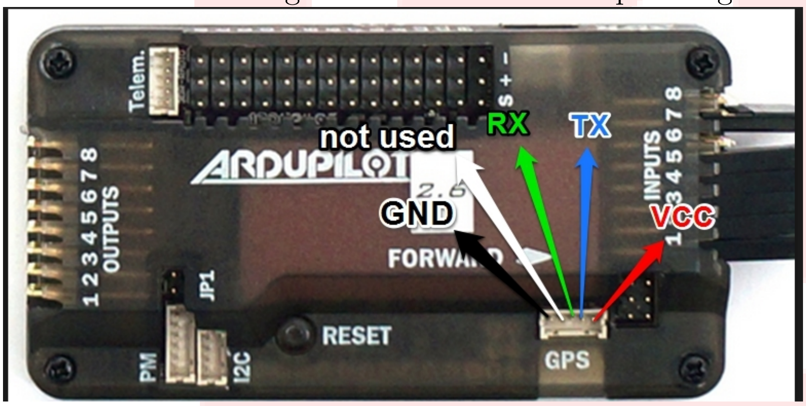


Figure 1.16: GPS Pin on APM

connect the module as shown in figure. connect APM with Mission planner and go to under open sky. for perfect gps location you GPS module should be under open sky. after 2 or 3 minutes GPS satellites will be connected to module. and you will find your Location on map in MISSION PLANNER software. you can also see no. of satellites connected to your gps module.

You will also get value of longitude,latitude,altitude on terminal of mission planner software.

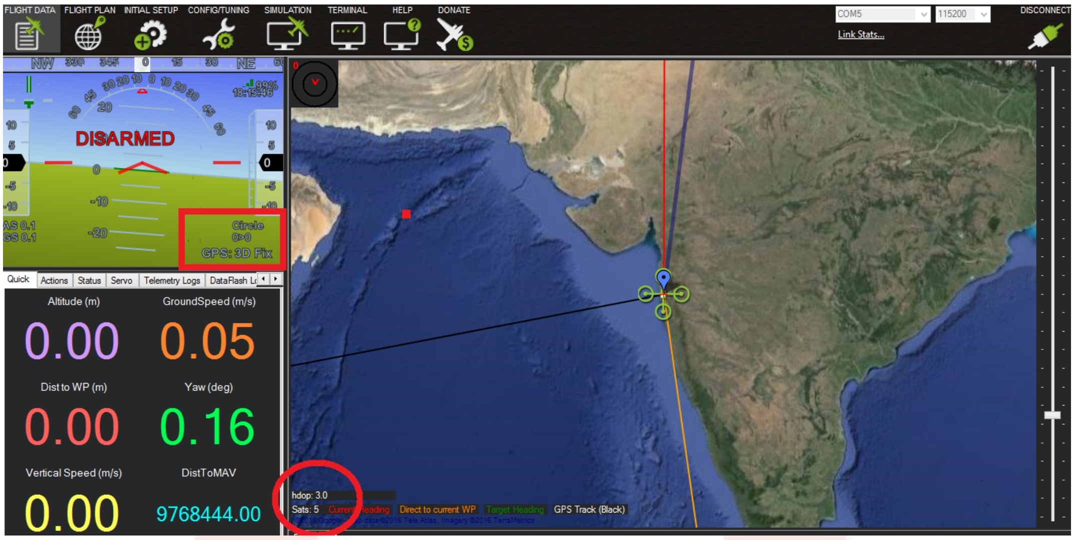


Figure 1.17: Current location

As you can see in the figure. mission planner shows 3D fix and no. of satellites are 5. As you move from one place to another you will be tracked as you can see in image.

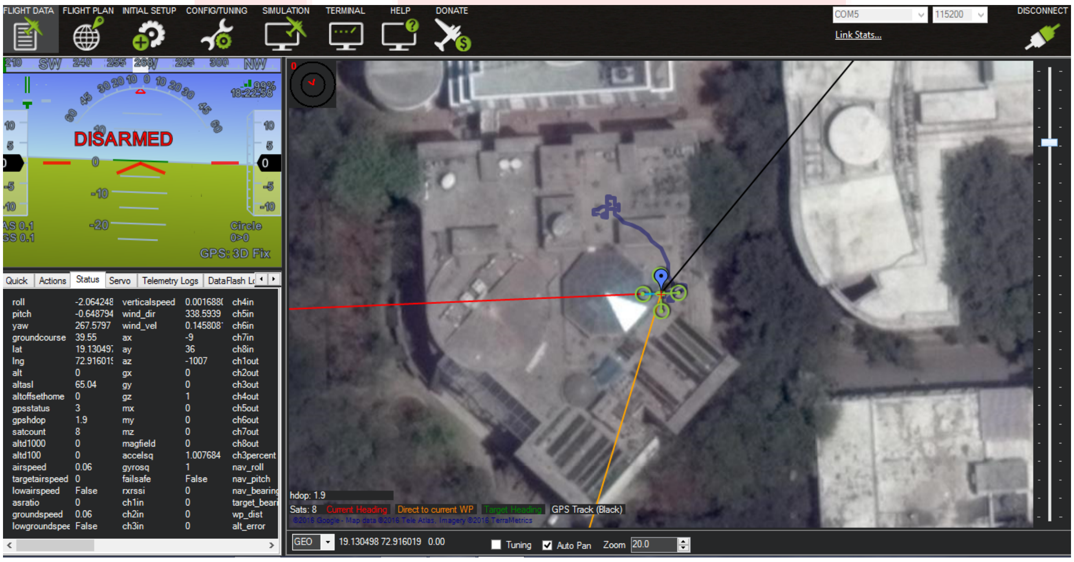


Figure 1.18: Closer look to location

You can see LAT , LON ,ALT and SATS on terminal.

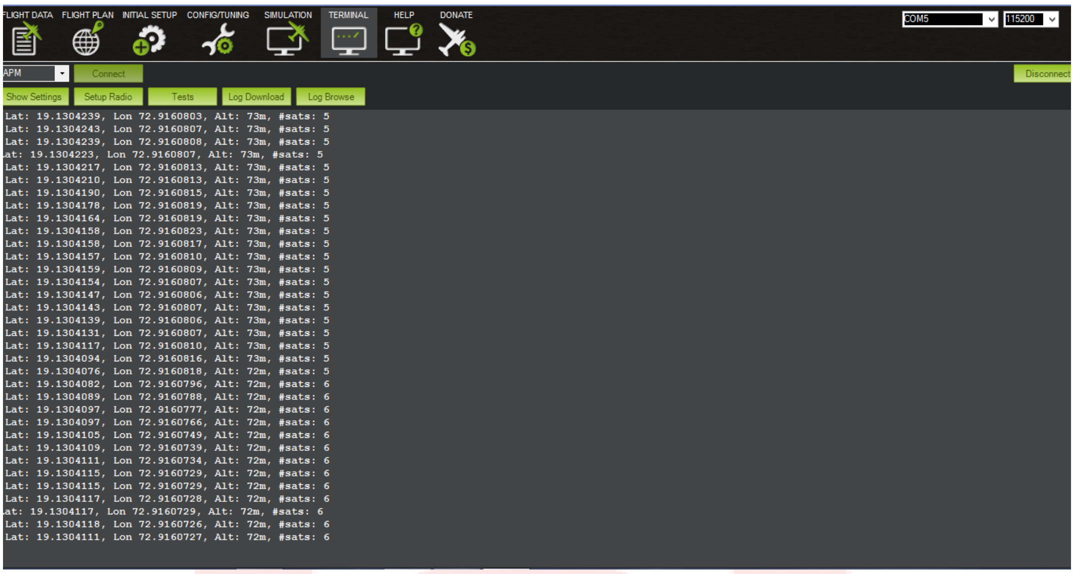


Figure 1.19: DATA from terminal

1.3.4 Interface Ultrasonic sensor with Raspberry Pi

Connecting to R-Pi

Powering the module is easy. Just connect the +5V and Ground pins to Pin 2 and GPIO21 on the Pis GPIO header.

Figure 1.20: HC-SR04 Sensor

The input pin on the module is called the trigger as it is used to trigger the sending of the ultrasonic pulse. Ideally it wants a 5V signal but it works just fine with a 3.3V signal from the GPIO. So we connected the trigger directly to GPIO20 on our GPIO header.

You can use any GPIO pins you like on your RPi but you will need to note the references and amend your Python script accordingly.

The modules output is called the echo and needs a bit more thought. The output pin is low (0V) until the module has taken its distance measurement. It then sets this pin high (+5V) for the same amount of time that it took the pulse to return. So our script needs to measure the time this pin stays high. The module uses a +5V level for a high but this is too high for the inputs on the GPIO header which only like 3.3V. In order to ensure the Pi only gets hit with 3.3V we can use a basic voltage divider. This is formed with two resistors.

If R1 and R2 are the same then the voltage is split in half. This would give us 2.5V. If R2 is twice the value of R1 then we get 3.33V which is fine. So ideally you want R2 to be between R1 and R1 x 2. But we used 1000 and 680 ohm resistors (Because it worked the best for us!).

Here is a diagram of our final circuit. We chose GPIO20 and GPIO16 [echo], but you can use any of the available GPIO pins on the GPIO header. Just remember to update the script.

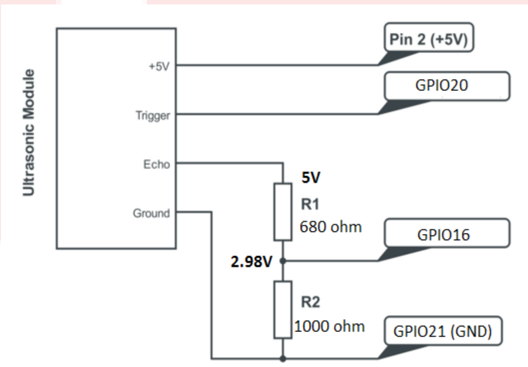


Figure 1.21: Circuit Diagram

Python Script

Once you have followed above connection diagram you can test the sensor you are ready with the next part i.e to access the sensor values using the python script. The script is in the codes folder.

Accuracy

- The accuracy of the distance measurement is dependent on timing. Python under Linux is not ideal for precise timing but for general messing about it will work OK. To improve accuracy you would need to start looking at using C instead.
- When the GPIOs are configured the module needs some time before it is ready to take its first reading so we added a 0.5 second delay to the start of the script.
- The transducers have a wide angle of sensitivity. In a cluttered environment you may get shorter readings due to objects to the side of the module.
- Measurements work down to about 2cm. Below this limit the results can give strange results.
- If the ultrasonic transducers touch anything the results are unpredictable.

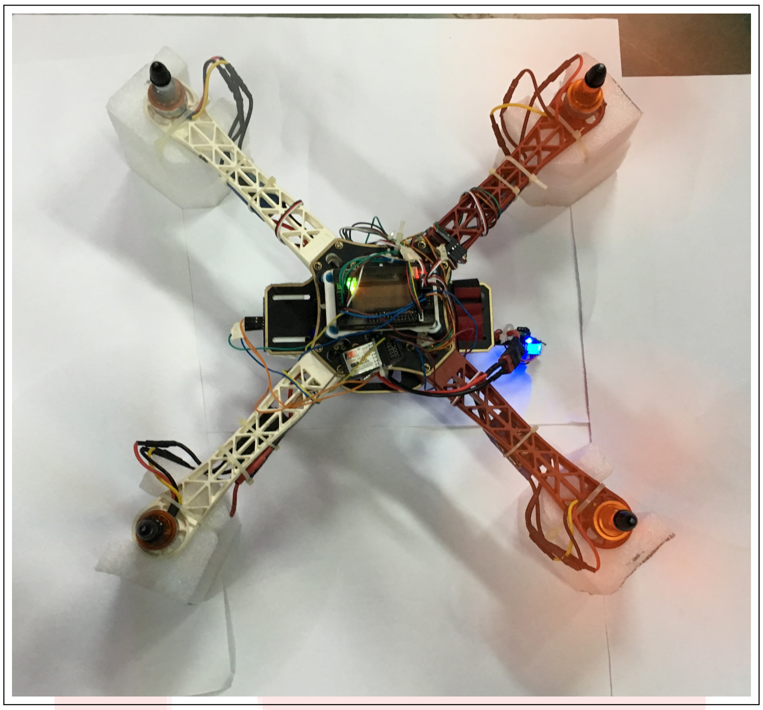
1.4 Software and Code

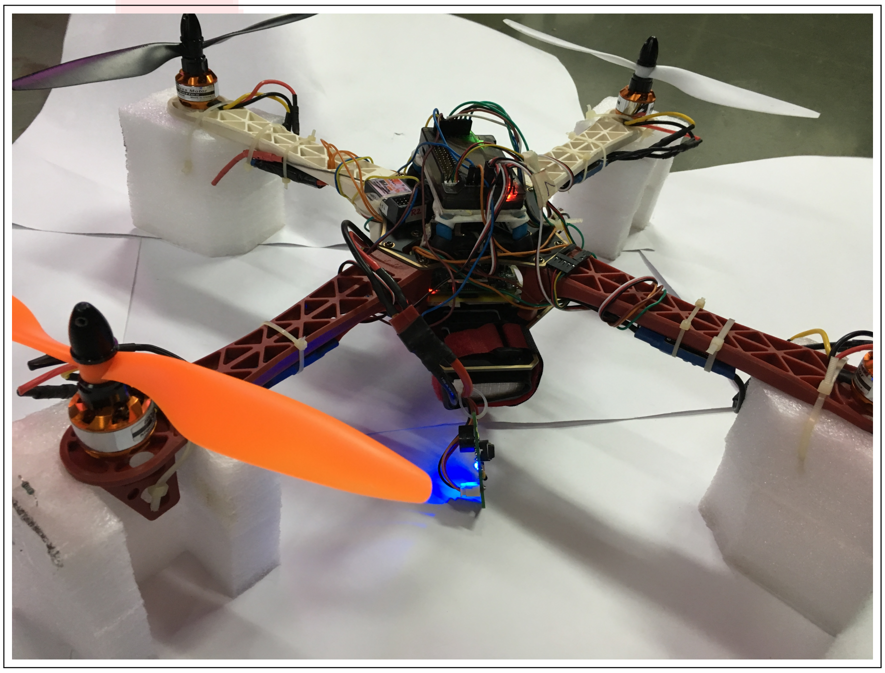
1. Autonomous Takeoff and Landing
2. Ultrasonic sensor take off
3. Quadcopter using keyboard control
4. A to B point Autonomous

Brief explanation of various parts of code is given in comments of code.

1.5 Use and Demo

Actual Drone Images





Videos of Flights:

1. Test flight 1
2. Test flight 2
3. Test flight 3

1.6 Future Work

1. Interfacing camera and gimbal
2. Object tracking drone
3. Obstacle avoidance
4. Get GPS data via WiFi and map places

1.7 Bug report and Challenges

Issues with hardware:

1. Noise in ultrasonic sensor
2. Propellers are not balanced

Challenges faced during project:

1. Interfacing Raspberry Pi and APM

1.8 Refrences

1. http://python.dronekit.io/
2. https://discuss.dronekit.io/
3. https://github.com/dronekit
4. http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html
5. http://www.hobbyking.com/hobbyking/store/index.asp
6. http://www.rcbazaar.com/default.aspx

7. http://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python

8. http://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

9. http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html

10. http://qgroundcontrol.org/mavlink/start

11. http://ardupilot.org/copter/docs/common-installing-3dr-ublox-gps-compass-module.html

12. http://ardupilot.org/planner/docs/common-install-mission-planner.html

13. http://ardupilot.org/planner/docs/common-loading-firmware-onto-pixhawk.html

14. http://ardupilot.org/planner/docs/common-connect-mission-planner-autopilot.html

15. http://ardupilot.org/copter/docs/connecting-the-apm2.html

16. https://multicopterbuild.wordpress.com/how-tos/quadcopter/

17. http://dronesforsaleclassified.com/product/dji-hot-wheels-diy-same-paragraph-quadcopter

18. http://www.flyingtech.co.uk/frames-props-parts-accessories/tarot-firefly-450-quadcopter-fr

19. http://www.diydrones.com

20. http://www.robu.in

21. http://www.quadkopters.com