Chapter 1

Introduction

1.1 Motivation and Background

In today's world, there is a lot of traffic in roads which leads to congestion in the whole city. So in the time of emergency critical situation. An ambulance which travels via road may not be able to reach the destination in time and the patient might lose his or her life. Thus, it is necessary to introduce a distinct means that would take the objective of saving human life one step closer. A drone or a hex copter takes aerial route and it's driven by human. Using more number of motors and propellers will produce thrust. The hex copter which consists of six BLDC Motors and propellers attached to it make it the optimal design and provide the necessary thrust. Power supply is 5200Mah battery is provided to the drone. The drone comprises of a medicine box which is capable reaching emergency situations faster than ambulance to help in emergency situation. The use drones in health is the purpose of the proposed prototype. For this reason, the first step is to develop a hexacopter. Both the thrust and the torque are produced by every hexacopter and it is produced about it's COR (centre of rotation). In addition to this a drag force is also produced in opposite direction to its flight.

1.2 Objective and Goals

The goal of our project, the air ambulance comes to the rescue situation, equipped with first aid and basic surgical kit. The air ambulance can reach the accident area within minutes and deploy necessary emergency supply. A GPS module will be used to determine the current position and SD card will be used to store the information needed. The use of drones in healthcare is the purpose of the proposed prototype. For this reason, the first step is to develop a hexacopter. Both the thrust and the torque are produced by every hexacopter and it is produced about it's COR (Centre of Rotation).In addition to this; a drag force is also produced in the opposite direction to its flight. Every hexacopter tries to achieve lift, yaw, roll and pitch via the thrust produced by the four motors attached to it. This way, the propellers fixed on the motors can be used for the flight of the hexacopter in all directions by differentiating the four rotors' thrust, the pitch and roll of the hexacopter can be controlled. The moment arm of each rotor's thrust about the CG, in steady state of the UAV should be equal.

1.3 Design of Hexacopter

1.3.1 Hexacopter configuration

Hexacopter has six motors attached to the frame with arms, these motors rotate the propeller at very high speeds to generate thrust. To achieve stability, three motors are made to rotate in clockwise(CW) and counterclockwise(CCW) directions. With this setup, it is able to cancel the net moment about the drone's yaw axis. The type of motors used in multi-rotors is BLDC motors, the rotations speed is controlled by using 3 different phases of current. The set speed of rotation of the motors is maintained with the help of ESC(Electronic Speed Controllers). ESCs constantly monitor the feedback current from the motor and make minute changes to the input current for the motors. Inertial Measurement Unit(IMU) which comprises of accelerometer and gyroscope, is used to measure short displacements and determine the orientation of the drone. This is done based on accelerations experienced by the body. Pixhawk is open-sourced flight controller, with on-board sensors that has many inbuilt algorithms to filter the raw sensor data and provides a precise data estimate. It also supports high level protocols such as MavLink which is used to communicate with a computer that may send commands to move the drone to a certain location in world's GPS coordinates. Batteries are used to power the on-board computer, sensors and motors. Optical flow sensor is used to measure very short displacements at very high frequency, this can be thought of as an odometry present in the wheeled robots. The model of optical flow sensor used is the PX4Flow, this sensor is chosen because it is widely used in UAV applications and has robust filtering algorithms like the EKF(Extended Kalman Filter), which has gained traction in recent years and finding many applications in real world implementations. Along providing the displacement along the ground plane, it also measures the altitude of the drone with the help on inbuilt sonar sensor. The ultrasonic sensor measures the distance from ground to the UAV's frame to determine the altitude of the flight. Moreover, the sensor is also capable of providing correct readings in indoor as well as outdoor environments, even without the need of illumination of LEDs. The optical flow methods attempt to calculate the motion at each voxel position between two image frames, which are taken at t and $t+\Delta t$. These methods are referred to as differential because they are based on local image signal approximations of the Taylor series; that is, partial derivatives are used with regard to spatial and temporal coordinates. Jetson nano is a computer that has powerful computation capabilities given its small footprint and weight profile. It is capable of running Neural Networks for image classification, object detection, semantic segmentation. The power consumption of this computer is as low as mere 5 watts. Lithium Polymer batteries are most commonly used due to their ability to provide high discharge rates and relatively procurable battery type. The specification of the battery chosen is 11.4 volts provided by three cells placed in series, with a discharge rate of 30C. The main frame chosen for the hexacopter design is the DJI F550. This frame is intended

towards hobbyist which enables to attach custom components such as GPS, telemetry, LIDARs(Light Detection and Ranging), on-board computers and attach custom motors. Based on these mentioned components.

It is implementation of Quad copter for more stability and strength. The use of drone as air ambulance. For this reason , the first step is to develop a hexacopter. Both the torque and thrust are produced by every hexacopter and it is produced about its COR (center of rotation). In addition to this a drag force is also produced in opposite direction to its Flights. Every hexacopter tries to achieve lift yaw , roll and pitch via the thrust produced by the six motors attached to it. This way, the propellers fixed on the motors can be used for the flight of the hexacopter in all direction. By the differentiating the six rotor's thrust the pitch and roll of the hexacopter can be controlled. The moment arm of each rotor's thrust about the CG, in steady state of the UAV should be equal.

$M = MI \times \alpha;$

Simplifying, $T = mr\alpha + mg;$

Differentiating, $dT/d\alpha = mr.$

Additionally, to increase stability, mass at the

Rotor (m) or the distance between the rotor and CG(r) have to be increased.

For Yaw stability, two of the hexacopter's rotors are counter-rotating, thus the reaction torque of two rotors have been countered. Equating thrust and weight, $3T\cos\theta = Mo \times g;$

Impulse-momentum principle, [3] MI rotor x $+3T\sin\theta.r = 0;$

Simplifying, $\tan\theta = [mrm2\omega/2] / grMo;$

From the technical specifications of the motor, the Maximum rotational speed of the motor is $\omega = 1900$

Rads-1. Thus $\theta = 0.43°$.

1.4 Construction

The main components are used in hexacopter drone is as follows

1.4.1) Frame :-HJ550mm Plastic Fibber Frame

1.4.2) Motors : Six 1000KV BLDC motors

1.4.3) Electronic speed controller(ESC): 30amp Simon

1.4.4) Flight Controller : Ardupilot flight controller

1.4.5) Battery :5200mAH Lipo

1.4.6) GPS module : M7N

1.4.7) Transmitter: FSi6 Flysky

1.4.8) Receiver:6 channel fsi6

1.4.9) Propeller:10*4.5

1.4.1 HJ550mm Plastic Fibber Frame

- Material: glass fibber + ABS plastic
- Color: red + black + white
- Weight: 424 grams
- Wheelbase: 550 mm
- Take off weight: 1200 - 2400 grams
- Propeller: 9443, 1045

The hexacopter is simple design with six rotor propeller with controller. The flight controller is one of the major part of this vehicle. It works on the principle of Newton's "3rd" law of motion " For every action there is an equal and opposite reaction". Hexacopter is a device with a intense mixture of Electronics, Mechanical and mainly on the principle of Aviation. The Hexacopter has 6 motors whose speed of rotation and the direction of rotation changes according to the users desire to move the device in a particular direction (i.e. Takeoff motion, Landing motion, forward motion, backward motion, right and left motion). The six rotors creates differential thrust and the hexacopter hover and move accordance with the speed of those rotors. Hexacopter movement mechanism Hexacopter can have described as a small vehicle with six propellers attached to rotor located at the cross frame. To control the hexacopter motion fixed pitch rotors are used. The speed of these six rotors are independent. By independent pitch, roll, and yaw attitude of the vehicle can be control easily. The basic three motions in the Hexacopter are: Hover: All the propellers are operated at approximately the same speed and therefore produce approximately the same thrust. Since all the propellers are equally spaced from the center of gravity, the thrust of the propellers produces no net rotating torque on the aircraft. Additionally, the hexacopter uses three clockwise (CW) rotating propellers and three counter-clockwise (CCW) rotating propellers so that the propeller torque is canceled when they are operating at equal speeds. In hover, the total upward thrust balances the downward gravitational force, and the multicopter maintains zero pitch and roll angles in zero wind (drag) conditions. Roll Control : A hexacopter can

be controlled about its roll axis by increasing the speed of the propellers on one side and decreasing the speed of the propellers on the other side. When the thrust increase on one side is the same as the thrust decrease on the opposite side, the net thrust remains the same. Similarly, the net effect of torque remains the same.

1.4.2 Motors :

1.4.2.1 Six 1000KV BLDC motors

A motor converts supplied electrical energy into mechanical energy. Various types of motors are in common use. Among these, brushless DC motors (BLDC) feature high efficiency and excellent controllability, and are widely used in many applications. The BLDC motor has power-saving advantages relative to other motor types. As their name implies, brushless DC motors do not use brushes. With brushed motors, the brushes deliver current through the commutator into the coils on the rotor. So how does a

brushless motor pass current to the rotor coils? It doesn't—because the coils are not located on the rotor. Instead, the rotor is a permanent magnet; the coils do not rotate, but are instead fixed in place on the stator. Because the coils do not move, there is no need for brushes and a commutator. With the brushed motor, rotation is achieved by controlling the magnetic fields generated by the coils on the rotor, while the magnetic field generated by the stationary magnets remains fixed. To change the rotation speed, you change the voltage for the coils. With a BLDC motor, it is the permanent magnet that rotates; rotation is achieved by changing the direction of the magnetic fields generated by the surrounding stationary coils. To control the rotation, you adjust the magnitude and direction of the current into these coils.

1.4.2.2 Advantages of BLDC Motors

A BLDC motor with three coils on the stator will have six electrical wires (two to each coil) extending from these coils. In most implementations three of these wires will be connected internally, with the three remaining wires extending from the motor body (in contrast to the two wires extending from the brushed motor described earlier). Wiring in the BLDC motor case is more complicated than simply connecting the power cell's positive and negative terminals; we will look more closely at how these motors work in the second session of this series. Below, we conclude by looking at the advantages of by BLDC motors.

One big advantage is efficiency, as these motors can control continuously at maximum rotational force (torque). Brushed motors, in contrast, reach maximum torque at only certain points in the rotation. For a brushed motor to deliver the same torque as a brushless model, it would need to use larger magnets. This is why even small BLDC motors can deliver considerable power.

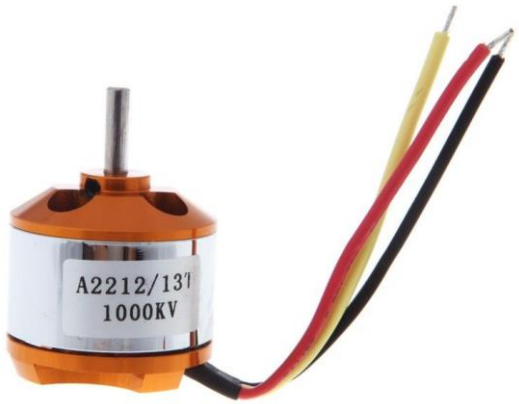
The second big advantage—related to the first—is controllability. BLDC motors can be controlled, using feedback mechanisms, to delivery precisely the desired torque rotation speed. Precision control in turn reduces energy consumption and heat generation, and—in cases where motors are battery powered—lengthens the battery life.

BLDC motors also offer high durability and low electric noise generation, thanks to the lack of brushes. With brushed motors, the brushes and commutator wear down as a result of continuous moving contact, and also produce sparks where contact is made. Electrical noise, in particular, is the result of the strong sparks that tend to occur at the areas where the brushes pass over the gaps in the commutator. This is why BLDC motors are often considered preferable in applications where it is important to avoid electrical noise.

1.4.2.3 BLDC Motor for UAV

1.4.2.4 A2212 1000KV BLDC

The A2212 1000Kv motor is a BLDC brushless outrunner motor specifically made for quadcopters and multirotor. It provides high performance and brilliant efficiency. These brushless BLDC motors are perfect for medium size quadcopters with 8-inch to 10-inch propellers. The A2212 brushless motors can be uses to build powerful and efficient quadcopters. They come complete with mounting bolts, prop adapters and power leads.



The A2212 1000Kv brushless motor is best used with an F550 quadcopter frame and a 30A ESC motor can be used to drive the motor.

Model	A2212
Motor KV (RPM/V)	1000
Compatible LiPO Batteries	2S ~ 3S
Shaft Diameter (mm)	3.17
Max. Efficiency Current (A)	4 ~ 10
Maximum Power (W)	150
No-Load Current (mA)	500
Internal Resistance(mΩ)	90

Maximum Efficiency	80%
Poles	14
Length (mm)	27.5
Width (mm)	27
Weight (gm)	64
Model	XT60

1.4.3 Electronic speed controller ESC

The term ESC stands for "electronic speed control is an electronic circuit used to change the speed of an electric motor, its route, and also to perform as a dynamic brake. These are frequently used on radio-controlled models which are electrically powered, with the change most frequently used for brushless motors providing an electronically produced 3-phase electric power low voltage source of energy for the motor. An ESC can be a separate unit that lumps into the throttle receiver control channel or united into the receiver itself, as is the situation in most toy-grade R/C vehicles. Some R/C producers that connect exclusive hobbyist electronics in their entry-level vehicles, containers, or aircraft use involved electronics that combine the two on a sole circuit board.

1.4.3.1 ESC Design

An electronic speed controller can be designed with three essential components like a voltage regulator/ BEC (Battery Eliminator Circuit)), a Processer & the switching includes FETs. The BEC is a separation of the electronic speed control that will transmit power back to your receiver after that to servos.

This also includes one secondary function like when the motor is operated through a battery then the motor gets its smallest voltage, then the BEC will keep some power for the flight control in dangerous situations so the motor doesn't consume total the power from the battery. At present, the processor is completely enclosed within a single Si semiconductor chip.

The main function of this processor is to decode the data being provided to it from the receiver within the model as well as to regulate the power toward the motor using FETs. In an ESC, this transistor plays a key role by performing all the works. It observes the

complete current & voltage of the motor as well as a battery. This transistor works like a switch to control the current flow to throttle the electric motor.

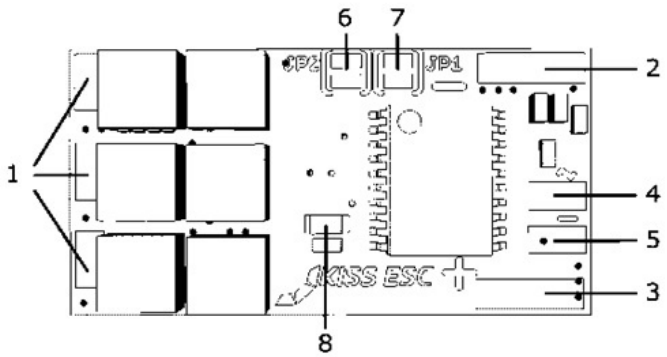
1.4.3.2 The Function of Electronic Speed Control

An ESC or electronic speed control mainly follows a speed reference signal to change the speed of a switching network of field-effect transistors. The motor speed can be changed by changing the switching frequency or the duty cycle of the transistors. For BLDC motors, different kinds of speed controls are necessary because this motor speed can be controlled by changing the voltage on its armature. This kind of motor needs a diverse operating rule like the motor speed can be changed by varying the timing of pulses for current transmitted to the different motor windings. Generally, the Brushless ESC systems make 3-phase AC power such as a variable frequency drive (VFD) to make the brushless motors work. These kinds of motors are more popular due to their power, efficiency, lightweight, longevity as compared to usual brushed motors. BLDC motor controllers are very complex as compared to brushed ones. The exact phase changes through the rotation of the motor, which can be taken into account using the electronic speed control. Generally, the rotation of this motor can be detected through back EMF, but variations that exist will utilize optical detectors otherwise separate Hall Effect sensors. In general, the speed controllers based on programming mainly include some options which are specified by the user that permits braking, acceleration, timing & direction of revolution. The reversed motor's direction can be accomplished by switching any 3 leads of the ESC toward the motor.

1.4.3.3 Components used in ESC

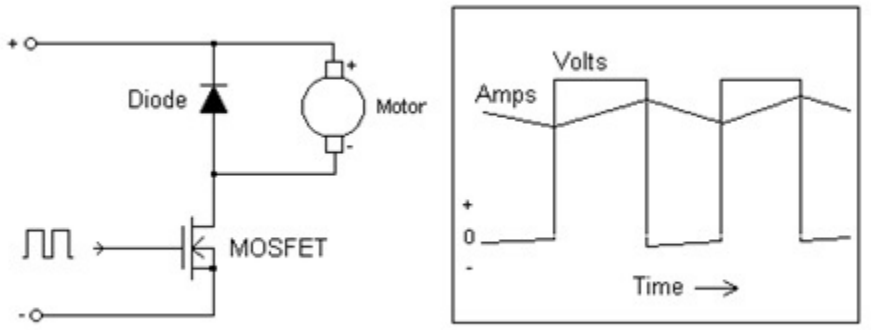
The components used in ESC mainly include the following

- Solder pads for the 3-BLDC motor phases
- Negative (-) LIPO connections
- Positive (+) LIPO Connection
- Servo signal or input of the PWM signal
- GND reference of PWM Signal
- Solder jumper, for altering the direction of Rotation (CW/CCW)
- Solder jumper, for varying the type of the PWM input signal State LED



1.4.3.4 Electronic Speed Controller Circuit

The term ESC is frequently used as a contraction for ‘electronic speed controller. The basic function of ESC is to change the amount of power to the electric motor from the aircraft battery based upon the location of the throttle stick. In earlier, speed controllers are mainly used in remote control boats and cars which use a variable resistor with a wiper that was stimulated by a servo motor.



This technique works reasonably at full throttle as the battery is associated straight to the motor, though at part throttle situations the flow of current through the resistor producing power to be lost in the form of heat. As a model, the aircraft will use most of its time at the portion of the throttle. This is not a very practical means of power control.

Current speed controllers differ the power to the motor by fast switching the power ON and OFF. Here, MOSFET Transistor is used as a switch instead of a mechanical device, and the amount at which it is switched is about 2000 times a second. So, the power to the motor is diverse by changing the amount of ON time, against off time in a specified cycle. Here is the simple ESC circuit with a waveform diagram that may help with the description.

When the MOSFET is switched ON, the current rises as the magnetic field in the windings of the motor increases. When the MOSFET is switched OFF, magnetic energy stored in the windings has to be absorbed by the ESC. By cabling a diode across the motor, we return the energy into the motor as current, which rises down as the magnetic field fails.

1.4.3.5 ESC Firmware

Most of the modern electronic speed controllers mainly include a microcontroller to understand the input signal & controlling the electric motor properly through an inbuilt program which is known as firmware.

In some conditions, it is achievable to modify the factory in-built program for an alternate. So this can be done by adapting the ESC toward a specific application. Some kinds of ESCs are inbuilt with the upgradeable firmware by the user whereas some types need soldering. Generally, these are sold like black boxes through proprietary firmware.

1.4.3.6 Electronic Speed Controller for Drone

In most of the drone's applications, the spread is constantly rising in a different range of applications which ranges from the hobbyist, industrial and commercial & in most superior military applications.

The main benefits of Drones are, they can be operated remotely, so flying over regions is very difficult, inconvenient, or dangerous to attain in person. The commercial applications of the drone are many like monitoring buildings, plants, agriculture, shooting areas & delivery of medicines, packages otherwise essential goods.

Generally, the high-range of drones mainly equipped with BLDC motors but these motors need cautious & continuous speed regulation for the relative direction of revolution. For that, the ESC circuit is responsible. So the ESC design mainly includes the following features.

- The topology utilized for controlling the motor
- Compromise among efficiency & cost
- The kind of battery used on the drone
- Necessary performance
- EMC (electromagnetic compatibility) and resistance to interference

In drones, there are two kinds of brushless motors are used like BLDC, BLAC and also called PMSM or permanent magnet synchronous motors. So the choice of motor mainly depends on the preferred control algorithm like trapezoidal otherwise FOC (field-oriented control).

1.4.3.7 Esc 30a

Simonk 30A BLDC ESC Electronic Speed Controller can drive motors which consume current up to 30A. It works on 2S-3S LiPo batteries. This electronic speed controller offers a battery eliminator circuit (BEC) that provides 5V and 2A to the receiver so no extra receiver battery is required. This version of the ESC also includes backward-

polarity protection and protection on the 5V receiver line, this means that if you accidentally attach a battery backward it won t destroy your motor controller and other

BECs won t affect the ESC.

1.4.4 Ardupilot flight controller

Copter is an advanced open-source autopilot system for multicopters, helicopters, and other rotor vehicles. It offers a wide variety of flight modes from fully manual to fully autonomous.

As part of the wider ArduPilot software platform it works seamlessly with a variety of Ground Control Station programs that are used to setup the vehicle, monitor the vehicle's flight in real-time and perform powerful mission planning activities. It also benefits from other parts of the ArduPilot ecosystem, including simulators, log analysis tools, and higher level APIs for vehicle control.

ArduPilot is already a preferred platform for numerous commercially available autopilot systems but you can also use it to enhance the abilities of your own DIY multirotor.

ArduPilot runs on many different autopilot boards, the most important of which are linked from the topic AutoPilot Hardware Options.

Selecting the right board depends on the physical restraints of the vehicle, features desired, and the applications that you want to run. Factors to consider are:

1. Sensor Redundancy: ArduPilot supports redundant IMUS, GPS, etc. Many controllers have multiple IMUs integrated on board.
2. Number of Servo/Motor Outputs
3. Number of UARTs: Telemetry radios, GPS's, Companion Computers, etc can be attached via these ports
4. External Buses: I2C, CAN, etc. allow many types of devices, such as airspeed sensors, LED controllers, etc. to be attached to the autopilot.

5. Number of Analog I/O: Some controllers have analog I/O available for such features as inputting receiver signal strength (RSSI) or battery voltage/current or other analog sensors.
6. Integrated Features: Such as on-board OSD (On Screen Display), integrated battery monitoring sensors
7. Size: Many vehicles have limited space for the autopilot.
8. Expense: Controller prices range from ~$25 to much more, depending on feature set.

Broadly speaking:

- Pixhawk is highly recommended for general use where sensor redundancy is desired and flexible external expansion.
- The "CUBE", CUAV V5, and Holybro Durandal/KakuteF7 AIO series of controllers offer mechanical vibration isolation of the IMUs built-in.
- Pixracer is recommended for small frames that require no more than 6 PWM outputs, with sensor redundancy.
- Emlid NAVIO2 Linux Autopilots should be considered for UAV Vision applications

1.4.4.1 Choosing a Ground Station

A ground station is typically a software application, running on a ground-based computer, that communicates with your UAV via wireless telemetry. It displays real-time data on the UAVs performance and position and can serve as a "virtual cockpit", showing many of the same instruments that you would have if you were flying a real plane. A GCS can also be used to control a UAV in flight, uploading new mission commands and setting parameters. It is often also used to monitor the live video streams from a UAV's cameras.

There are at least ten different ground control stations.

On desktop there is (*Mission Planner*, *APM Planner*

2, *MAVProxy*, *QGroundControl* and *UgCS*. For Tablet/Smartphone there is *Tower* (DroidPlanner

3), *MAVPilot*, *AndroPilot* and *SidePilot* that can be used to communicate with ArduPilot (i.e. Copter, Plane, Rover, AntennaTracker).

The decision to select a particular GCS often depends on your vehicle and preferred computing platform:

- **Ready-to-fly** users may prefer the portability and ease of use of *Tower* (Droid Planner 3), or another GCS running on a tablet or phone.
- **DIY/Kit** users and developers often have to access configuration and analysis tools, and would therefore need (at least initially) *Mission Planner*, *APM Planner 2* or another more full-featured GCS.

1.4.4.2 Comparison Desktop:

1.4.4.2.1 Mission Planner

Full featured and widely used GCS.

The Mission Planner, created by Michael Oborne, does a lot more than its name. Here are some of the features:

- Point-and-click waypoint/fence/rally point entry, using Google Maps/Bing/Open street maps/Custom WMS.
- Select mission commands from drop-down menus
- Download mission log files and analyze them
- Configure autopilot settings for your vehicle
- Interface with a PC flight simulator to create a full software-in-the-loop (SITL) UAV simulator.
- Run its own SITL simulation of many frames types for all the ArduPilot vehicles.

1.4.4.2.2 What is Mission Planner

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. Here are just a few things you can do with Mission Planner:

- Load the firmware (the software) into the autopilot board (i.e. Pixhawk series) 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.
- With appropriate telemetry hardware you can:
    - Monitor your vehicle’s status while in operation.
    - Record telemetry logs which contain much more information the the on-board autopilot logs.
    - View and analyze the telemetry logs.
    - Operate your vehicle in FPV (first person view)

1.4.4.2.3 Installing Mission Planner:

Mission Planner was designed for native Windows installation. However, it is possible to use it under Linux (with some occasional issues) and there is a Beta version for Android OS.

1.4.4.2.4 Loading Firmware:

These instructions will show you how to download the latest firmware onto the autopilot hardware that already has ArduPilot firmware installed. This process will use the Mission Planner ground control station.

Most often, these boards have another flight controller software pre-installed. (If the board has ArduPilot already installed, see Loading Firmware for firmware loading instructions.

Installing ArduPilot to these autopilot involves:

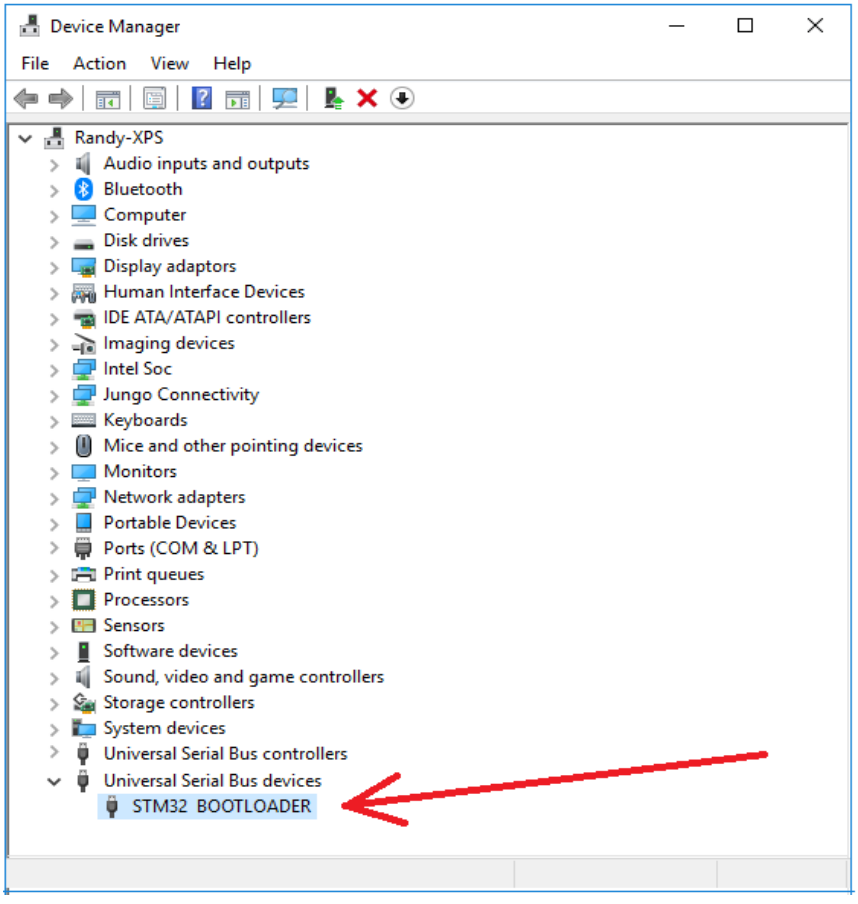
- Installing the required driver and flashing tool
- Downloading the appropriate ArduPilot firmware
- Loading ArduPilot to the board

1.4.4.2.5 Download the ArduPilot firmware

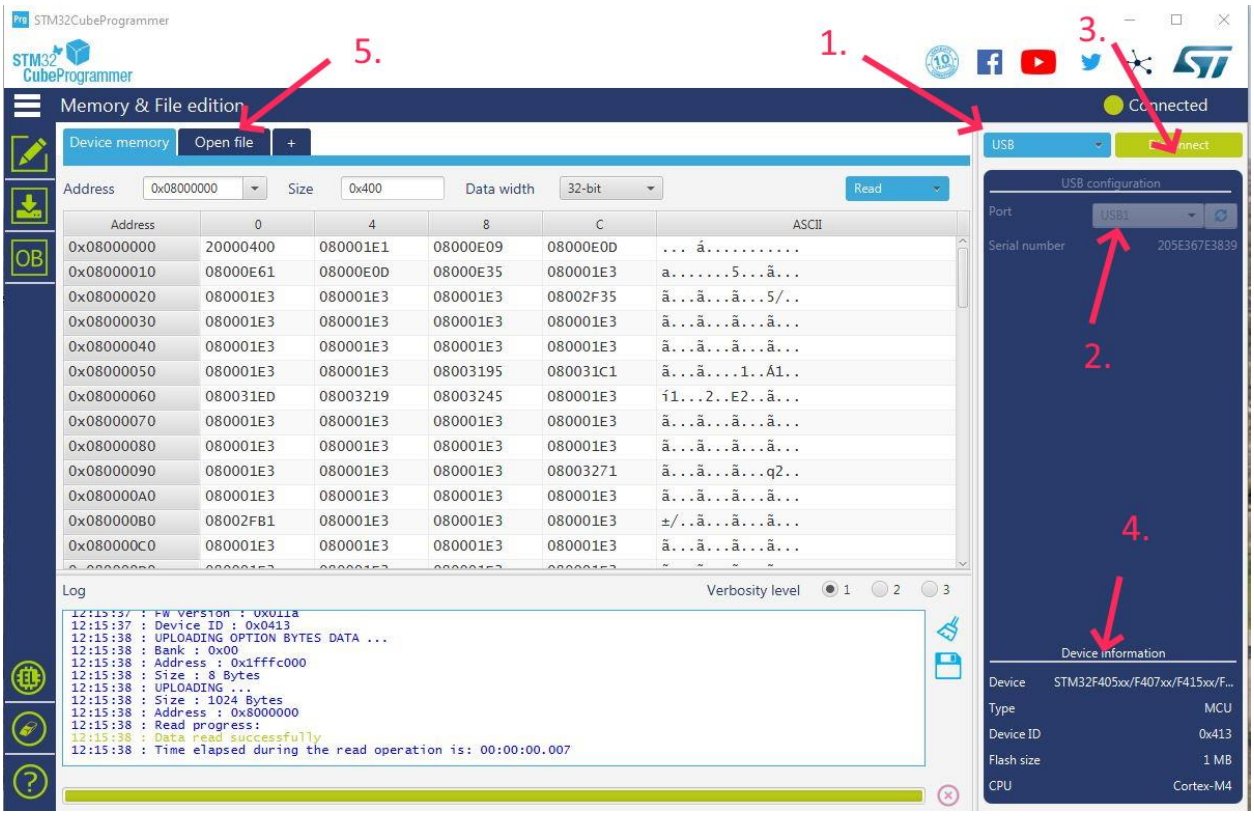
- Download the ArduPilot firmware for your board from firmware.ardupilot.org. You can normally find the appropriate firmware by doing the following:
    - open firmware.ardupilot.org
    - select click on the link for your vehicle type (i.e. Plane, Copter, Rover, Sub or Antenna Tracker)
    - select “beta” or “stable”
    - look for the directory with the name that most closely matches the autopilot
    - download the “arduXXX_with_bl.hex” file clicking on it. It will usually be saved in your Downloads folder.

Upload the firmware to autopilot:

- Hold down the board’s DFU button or temporarily bridge its “BOOT” pins, and plug in a USB cable (attached to your PC). Release button or unbridge once powered.
- Open the windows device manager and look under “Universal Serial Bus devices” for “STM32 BOOTLOADER” to confirm that the board is in Dmode.



Start the STM32CubeProgrammer



1. Select the connection method: USB
2. Make sure a USB port shows…that means the board is detected in DFU mode.
3. Press “Connect”
4. Then the boards cpu specifics will appear here.
5. Press “Open file” to select the “arduXXX_with_bl.hex” file you downloaded.
6. The file name will appear in the tab.

1.4.4.2.6 Setting up the connection:

To establish a connection you must first choose the communication method/channel you want to use, and then set up the physical hardware and Windows device drivers. You can connect the PC and autopilot using USB cables, Telemetry Radios, Bluetooth, IP connections etc.

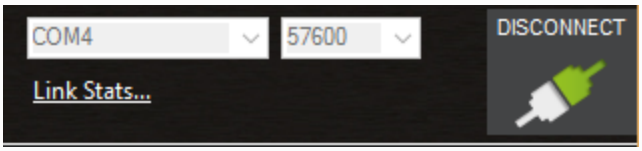
1.4.4.2.7 Connection using SiK Radio

On *Mission Planner*, the connection and data rate are set up using the drop down boxes in the upper right portion of the screen.

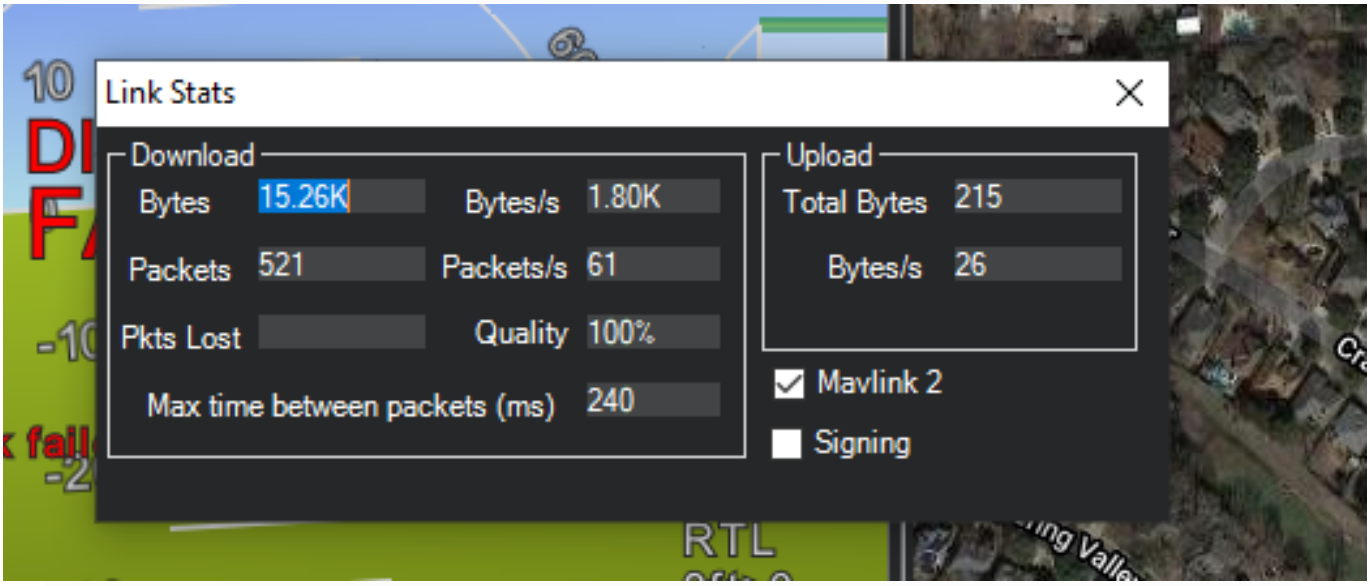
Once you've attached the USB or Telemetry Radio, Windows will automatically assign your autopilot a COM port number, and that will show in the drop-down menu (the actual number does not matter). The appropriate data rate for the connection is also set

(typically the USB connection data rate is 115200 and the radio connection rate is 57600).

Select the desired port and data rate and then press the **Connect** button to connect to the autopilot. After connecting **Mission Planner** will download parameters from the autopilot and the button will change to **Disconnect** as shown:

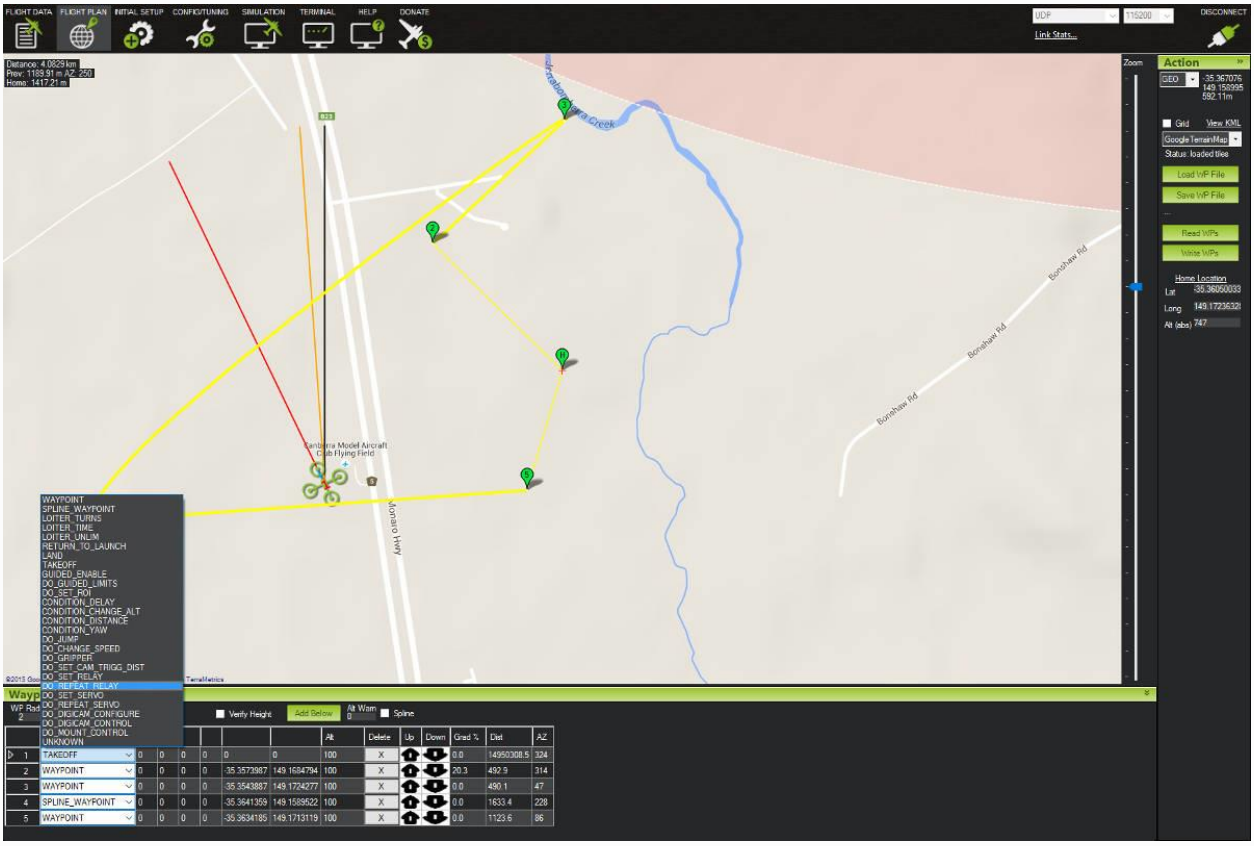


The "Stats…" hotlink beneath the port selection box, if clicked, will give information about the connection, such as if Signing security is active, link stats, etc. Sometimes this window pops up beneath the current screen and will have to be brought to the front to be seen.



1.4.4.2.8 Mission Planning

This section contains articles about creating missions that will run when the vehicle switched to AUTO mode.



1.4.4.3 Planning a Mission with Waypoints and Events:

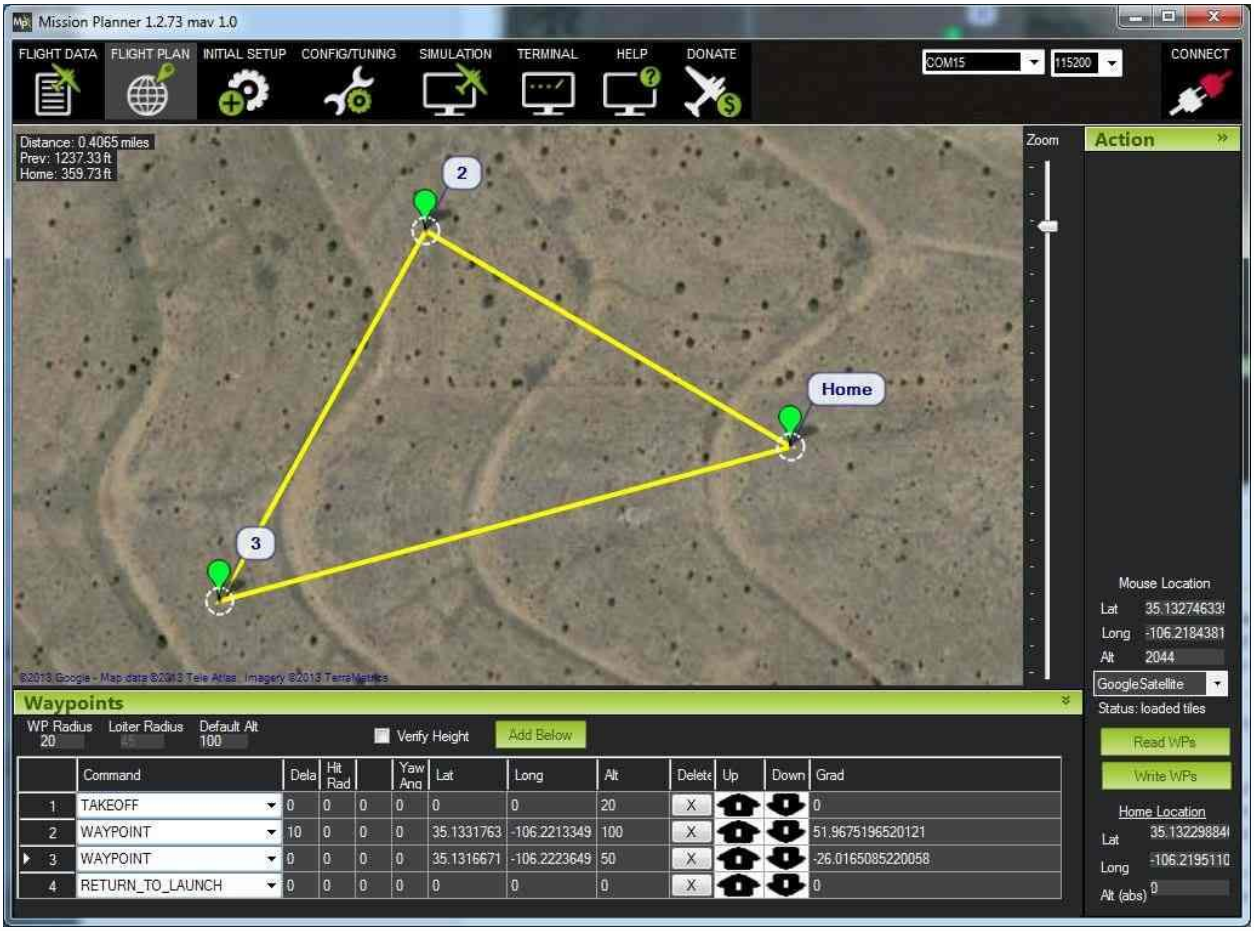
1.4.4.3.1 Setting the Home Position

For Copter, Plane and Rover the home position is set as the location where the vehicle was armed. This means if you execute an RTL, it will return to the location where it was armed, so arm your vehicle in the location you want it to return to, or use a rally point to setup an alternative return point.

Instructions

In the screenshot below, a Copter mission starts with an auto takeoff to 20 meters altitude; then goes to WP 2 rising to 100 meters altitude on the way, then waits 10 seconds; then the craft will proceed to WP 3 (descending to 50 meters altitude on the

way), then returns to launch. After reaching the launch position, the craft will land. The mission assumes that the launch position is set at the home position.



You can enter waypoints and other commands (see the Mission commands section below for more information). In the dropdown menus on each row, select the command you want. The column heading will change to show you what data that command requires. Lat and Lon can be entered by clicking on the map. Altitude is relative to your launch altitude/home position, so if you set 100m, for example, it will fly 100m above you.

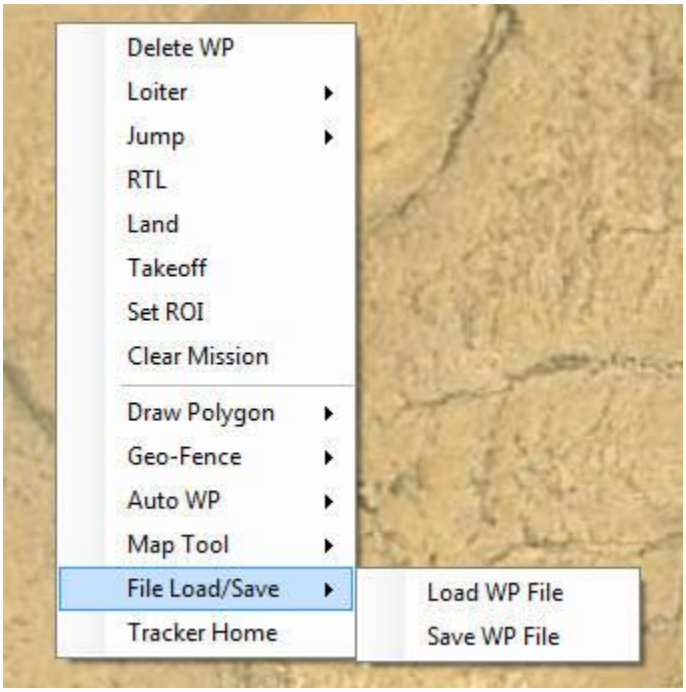
Default Alt is the default altitude when entering new waypoints. See Understanding Altitude in ArduPilot for altitude definitions.

Verify height means that the Mission Planner will use Google Earth topology data to adjust your desired altitude at each waypoint to reflect the height of the ground beneath.

So if your waypoint is on a hill, if this option is selected the *Mission Planner* will increase your ALT setting by the height of the hill. This is a good way to make sure you don't crash into mountains!

Once you are done with your mission, select **Write** and it will be sent to APM and saved in EEPROM. You can confirm that it's as you wanted by selecting **Read**.

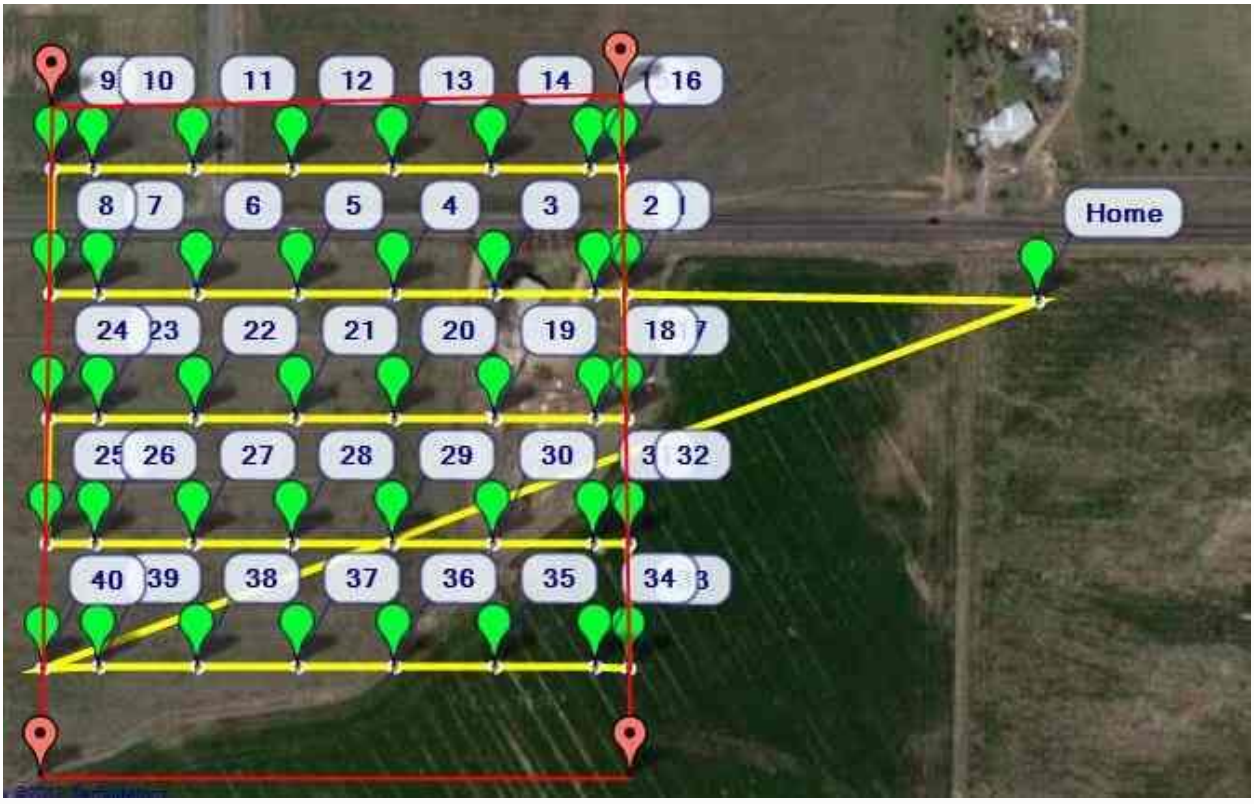
You can save multiple mission files to your local hard drive by selecting **Save WP File** or read in files with **Load WP File** in the right-click menu:



Auto grid

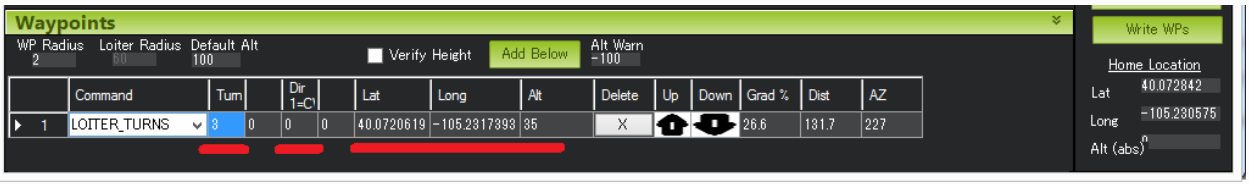
You can also have the *Mission Planner* create a mission for you, which is useful for function like mapping missions, where the aircraft should just go back and forth in a "lawnmower" pattern over an area to collect photographs.

To do this, in the right-click menu select Polygon and draw a box around the area you want to map. Then select Auto WP, Grid. Follow the dialog box process to select altitude and spacing. The *Mission Planner* will then generate a mission that looks something like this:



1.4.4.3.2 Mission commands

Mission Planner provides a filtered list of the commands appropriate for the current vehicle type, and adds column headings for the parameters that need user-supplied values. These include navigation commands to travel to waypoints and loiter in the vicinity, DO commands to execute specific actions (for example taking pictures), and condition commands that can control when DO commands are able to run.



Example: LOITER_TURNS command withheadings for number of turns, direction, and location to loiteraround.

The full set of mission commands supported by all ArduPilot platforms are listed in MAVLink Mission Command Messages (MAV_CMD). This includes the full name of

each command (as defined in the protocol definition), information about which parameters are supported, and also the corresponding *Mission Planner* column headings.

1.4.4.3.3 Mission Reset

One can set an RCx_OPTION switch function ("24") to reset the mission item pointer to the beginning of the mission list at any time.

1.4.4.3.4 Mission Re-Wind

The behavior of returning to a mission sequence when interrupted by a mode change is described in the Mission Rewind on Resume section.

1.4.4.3.5 Camera Control in Auto Missions:

Planning a camera mission is almost exactly the same as planning any other mission with waypoints and events. The only difference is that in a camera mission you specify commands to trigger the camera shutter at waypoints or at regular intervals as the vehicle moves. If the camera is mounted on a gimbal, you can also set the gimbal orientation, or make it track a particular point of interest.

For simple missions you can manually specify the required waypoints and camera commands. For more complex paths and grid surveys Mission Planner makes things easy by providing tools to automatically generating the required mission for arbitrary regions.

1.4.4.3.6 Camera commands

- DO_SET_CAM_TRIGG_DIST — Trigger the camera shutter at regular intervals. This is most commonly used for supporting area surveys.
- DO_DIGICAM_CONTROL — Trigger the camera shutter once every time this command is called.

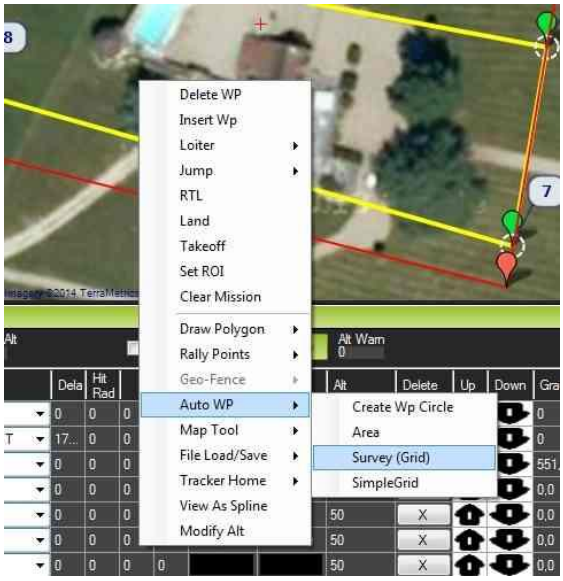
1.4.4.3.7 Auto-mission types

Mission Planner supports the following *Auto Waypoint* options. To access these open the *Flight Plan* screen, right-click on the map and select the option from under the *Auto WP* menu:

- **Create WP Circle** — Create a circle of waypoints.
- **Area** — Displays the area of the current polygon (if defined).
- **Create Spline Circle** — A circle where the altitude of waypoints follows a rising spline (relevant to flying vehicles).
- **Survey (Grid)** — Automatically create waypoints and camera control commands to survey a specified polygon.
- **Survey (Gridv2)** — *Under construction!* This is a simpler grid control for creating a rectangular survey area.
- **SimpleGrid** — A simple auto-created survey grid. No camera control is defined, so this must be added separately.

1.4.4.3.8 Survey (Grid) Example

Mission Planner's *Survey (Grid)* option automatically defines the waypoints required to cover an arbitrary polygon, and sets DO_SET_CAM_TRIGG_DIST on relevant waypoints to ensure that pictures are captured at regular intervals in the vehicle's path.



- Open the *Flight Plan* tab
- **Right click** on the map, and select **Draw Polygon | Add PolyGon Point**. Create points surrounding the area to be photographed.
- **Right click** on the map and select **Auto WP | Survey(Grid)**:

1.4.5 Battery

1.4.5.1 Size

Battery size is the first point you need to consider because the size is often related to capacity. Usually, the larger the battery, the more capacity stored and the longer it can fly. However, a battery that is too large will add to the overall weight of the drone, thus reducing flight time and affecting the flight experience.

Therefore, you need to choose the capacity while keeping the maximum takeoff weight of the drone.

1.4.5.2Voltage

Battery voltage is also known as the cell count: it is proportional to the power produced by the motor. It will help your drone motors in having more power, but, at the same time, the weight will be heavier with the increased number of cells.

The nominal voltage of a normal LiPo battery is 3.7V. The voltage of the battery pack can be increased by connecting the batteries in series. The number of cells used in a LiPo pack is shown by a number followed by the letter 'S'. For example, a 2S battery has 2 cells wired together in series to create a 7.4v battery.

To find the best voltage for your drone, check the motor thrust data. You can determine the number of batteries required based on the data. Before making a decision, check with the manufacturer whether the motor supports a specific battery number and voltage range.

1.4.5.3 Connector

The connector is also an important part of the drone battery. Not only do they allow you to easily replace the battery, but they can also help when you decide to make a new drone. There are many types of connectors, so make sure that the one you select is fully compatible with your drone.

1.4.5.4 Discharge Rate

The discharge rate is the C-rating of the battery: this rating helps you in identifying the maximum current that your battery can safely discharge. A higher C rating means that you will use less throttle input to hover and more current intensity to the motor at full throttle, making your drone faster and powerful.

To ensure your drone can maximize its functions, check the motor's specification and assess the maximum current received using the following formula:

Maximum Continuous Amp Draw = Battery Capacity * Discharge rate.

For example, a 10C – 2,000mAh battery will safely discharge at 20,000mAh

*10 * 2000 = 20,000 mA*

1.4.5.5 Brand

Choosing a reliable brand can save you a lot of unnecessary hassle. There are many brands of batteries available in the market, but they vary in quality and performance. Poor quality batteries will reduce the overall performance of your drone.

Grepow, one of the top UAV battery manufacturers, is a great place for you to look. Its Tattu batteries have passed the UN 38.3 battery test, the highest quality standard test for lithium polymer batteries, and their reliability and reputation can provide you with a wonderful flying experience.

1.4.5.6 5000Mah Lipo Battery:

Battery parameter:ZOP Power 11.1V 5000mAh 3S 30C Capacity:5000mAh Size:25.5*48*155mm Plug Style:XT60 Plug Weight:385g Colors:Standard Colors Note: due to the difference between the measurement, the battery dimension and weight may have little error. Caution: Don't over-charge, or over-discharge batteries. Don't put it beside the high temperature condition. Don't throw it into fire. Don't throw it into water. Package Included: 1 x ZOP Power 11.1V 5000mAh 3S 30C Lipo Battery XT60 Plug

1.4.6 GPS module :

GPS drones are equipped with a GPS module that allows them to know their location relative to a network of orbiting satellites. Connecting to signals from these satellites allows the drone to perform functions such as position hold, autonomous flight, return to home, and waypoint navigation.

GPS is the same system that we are all familiar with in road navigation systems. A global network of orbiting satellites sends signals that a GPS module picks up with a radio receiver. These signals allow the module to determine its position, speed, and time. GPS uses the concept of triangulation to determine relative position and speed, typically using three or four satellite signals, although some drone GPS modules will lock on to up to seven or eight separate satellite signals for optimal performance

There are several different worldwide GPS systems. Within the US, the GPS network of satellites is distributed so as to ensure that at least four satellites are theoretically visible from any given point on the land surface. Russia operates the GLONASS GPS system of satellites, and Europe is in the process of developing the Galileo GPS system. China is also developing its own GPS network, the BeiDou, while India is developing the IRNSS GPS system.

Some GPS modules on drones are capable of receiving signals from any of these sources, though most are limited to a single system. Having access to a greater number of signals and systems can increase accuracy of positioning, and thus all the GPS dependent features of your drone. Especially if you plan to travel with your drone, you may want to look specifically for a GPS drone that can receive signals from other systems. Your standard GPS drone will have position location accuracy of a meter or so, while more advanced GPS drones can have accuracy of up to a centimeter.

GPS technology has improved enough over the past few years to make it both affordable and lightweight enough to be more or less standard in your average consumer drone. Even some of the drones that would fall in the toy category come with GPS functionality. Having GPS on your drone makes a big difference in how it performs. It plays an important role in many of the features that pilots have come to rely on. Check out the following ways GPS helps out drone pilots.

1.4.6.1 Position Hold

When the drone can lock on to a GPS signal, it is able to identify and maintain its position at a fixed location. You could also call this the ability to achieve a stable hover. If you are not feeding the drone any controls at all, with GPS it will be able to stay in place even under a moderate breeze. If it identifies that it has drifted away from its original location, it will automatically correct and return to the same spot mid-air.

The same principle applies to achieving more stable flight. A drone without GPS is much more blown off course by breezes, and the pilot has to work much harder to maintain a straight, smooth flight path. The GPS, however, helps the drone to stay on course much more smoothly and steadily by identifying where it is.

1.4.6.2 Altitude Hold

Some drones also use GPS to hold a steady altitude while in flight. This is especially important given the FAA regulation that drone aircraft must operate below 400ft Above Ground Level (AGL). Many GPS drones will have this altitude limit pre-programmed in the flight controls. However, many drones actually use onboard barometric sensors to determine altitude, or imaging sensors to determine proximity or distance from objects directly below, so altitude hold is not necessarily dependent on GPS. It depends on the drone. Either way, holding a steady height in manual flight (without GPS or other sensors) can be quite a challenge.

1.4.6.3 Return to Home

The return to home feature is one of the best friends of a beginner pilot (and more advanced as well!). The drone will remember and return to its exact take-off location, to within a foot or two. This is obviously only possible with GPS to tell the drone where it is at any given time, and where it started from.

Return to home becomes important if your battery power runs low, you're reaching the outer range of connection to your controller or you've gotten disoriented at the controls and don't know how to get your drone going back in the right direction. Some drones will have an automatic return to home function kick in if the battery gets too low or your

controller signal is interrupted. Almost all GPS drones button that will automatically send the drone back to its take-off point.

1.4.6.4 Reporting

Many GPS drones create a flight log of each flight – how long you flew, where you flew, etc. This can be helpful just in terms of record keeping, but it is also helpful in locating and recovering a crashed drone. You can access the flight log through your controller or controller app, and determine the last known GPS location. Chances are your drone will be there, or somewhere not too far off.

1.4.6.5 Waypoint Navigation

If you would like to preplan a flight path or mission for your drone, this can be done by directing it to navigate to specific GPS coordinates along the path, called waypoints. A GPS drone can use autopilot function to travel along the path that you have predetermined. Autonomous flight instructions at the waypoints can also direct the drone to hover for a set amount of time at each location.

You might do this if you want to get pictures of a specific area. For an aerial photographer, if the drone is set on a predetermined course and flies it without any need for input from the controller, the pilot can focus on working the camera for the shot you want. The autonomous flight is likely going to be smoother than any manual control can achieve, and will therefore also deliver smoother, clearer camera images. Many commercial sectors heavily rely upon waypoint navigation in drone technology. GPS is absolutely essential for the use of drones in mapping, inspection, construction, agriculture, and more.

1.4.6.6 Mapping

This is not a flight assist feature, but as a side note, if you are hoping to use images with any 3D mapping or photogrammetry software, GPS on drones is essential for the geo-tagging of images to create the maps.

1.4.6.7 Battery Drain

GPS in a drone drains the battery faster. There's no way around it. If you have a small drone with a small battery, the extra weight and extra technology needed to operate the GPS module will bring down the battery life. However, for all intents and purposes, drones these days that are not equipped with GPS are going to have a relatively shorter battery life compared to a standard consumer drone, just because they are cheaper and have smaller batteries anyway.

1.4.6.8 Higher Cost

Drones with GPS are going to cost a bit more than drones without it. Partly it's the GPS module, but mostly it's all the other technology that has to go in to make use of the satellite signals – the hovering stability, the return to home, all the other intelligent features need programming, and that equals higher cost.

1.4.6.9 Potential Complications

The GPS dependent functions on the drone rely on the signals they are able to receive. Sometimes there can be a delay in the signal, or the GPS module can switch between signals mid-flight and confuse the drone about its actual location, possibly leading to a crash or flyaway. However rare these cases might be, if it happens to you, it's sure to be extremely frustrating.

Another complication is that many GPS drones will look for simultaneous connection to three satellites, or may not even report as being ready to fly until seven signals are detected. While this is presumably to achieve optimal flight, you just might be in a location that can't pick up that many satellites, and you're left grounded. So would it be better to fly without any GPS, or to not get off the ground at all because you can't get enough signals? A frustrating choice.

1.4.6.10 GPS-denied Scenarios

There are certain situations where GPS is simply not available, but drones are needed that will have high levels of location precision and accuracy. Most of these scenarios involve commercial or professional applications of drones, and they can get around the lack of GPS with other sophisticated sensors and programming. Some of these situations include:

Indoor inspections – If a drone is flying inside a warehouse, or inside an asset such as oil tanks or industrial boilers, chances are small that it will be able to find GPS signals with any degree of reliability.

Mining – GPS signals can't get underground, so again, drones are not going to be able to rely on GPS for location.

Bridge or Building Inspections – Large metal objects can obstruct the GPS signal, causing problems for the drone's ability to operate reliably.

Critical Infrastructure or High Security Locations – There is some concern for the security of GPS signals, and for that reason some government agencies are hesitant to utilize GPS for drone operations near places such as military bases or power plants.

In these types of cases, high-end professional drones utilize a variety of methods to achieve stable flight, hover in place, and "know" its relative location in space. The most common method is obstacle avoidance sensors that provide the drone with reference points, giving it the ability to hold itself steady in relation to them. Another approach is a system called SLAM (Simultaneous Location and Mapping) by which drones make use of sensors to create real time maps of their surroundings and navigate by them.

1.4.6.11 NEO-7M GPS Module

Neo 7M GPS module that includes an HMC5883L digital compass. The new NEO 7 series is a high sensitivity, low-power GPS module that has 56 channels and outputs precise position updates at 10Hz. This GPS module also comes with a molded plastic case which keeps the module protected against the elements making it ideal for use on your aircraft or quadcopter.

This Neo 7M GPS module uses an active circuitry ceramic patch antenna to provide an excellent GPS signal which outperforms the older Neo 6 series modules. This Neo 7 module also includes a rechargeable backup battery to allow for HOT starts and also includes an I²C EEPROM to store the configuration settings. Out of the box this GPS module is configured to run at 38400 Baud and is configured to run with APM/Pixhawk systems. This GPS module includes two cables, a 6pin connector for the GPS module, and a 4 pin connector for the i2c compass.

1.4.6.12 Features :

1. Locate performance
2. These are Pre-configured, Flashed with the correct settings, and tested. To make them Plug and Play.
3. Super Bright LED
4. Backplane with Standard Mk style mounting holes 45mm X 45mm
5. 38400 bps (Default) Changed to 115200bps!
6. Output GGA, GSA, and RMC frames
7. 1Hz (Default) Changed to 5Hz!
8. Permanent Configuration Retention
9. compass on board
10. 6 pin connector for EZ connect to MEGA BLACK
11. 4 pin connector for only GPS use
12. 4 pin connector for compass only use
13. Can use both 4 pins at once.

1.4.7 Transmitter:

A radio control system is made up of two elements, the transmitter you hold in your hands and the receiver you put inside your drone. Dramatically simplifying things here, your drone transmitter will read your stick inputs and send them through the air to your receiver in near real time. Once the receiver has this information it passes it on to your drones flight controller which makes the drone move accordingly. A radio will have four separate channels for each direction on the sticks along with some extra ones for any auxiliary switches it may have.

1.4.7.1 FREQUENCY AND CHANNELS

Thankfully frequency and channel wise radio controls are a lot smarter than their FPV counter parts and are much easier to manage. Video transmitters and receivers for example both require setting to the correct channel along with diligent channel management every time you fly. A Radio Controller however simply needs to bind or pair with a receiver when it's first setup.

From then on it will always link and hop over various frequencies in the 2.4Ghz band to ensure a solid link with theoretically hundreds of pilots operating at the same time.

1.4.7.2 RANGE TECHNOLOGY

The limit of range is normally where the receiver can no longer clearly hear what the transmitter telling it and typically falls in the 1km range in normal conditions. Imagine trying to talk to someone across a field The range of your radio link will be dependent on a few factors:

The output power of your transmitter - Many run just below the legal maximum to be compliant with international standards.

The sensitivity of the Receiver - A more sensitive receiver is like having better hearing, the signal will travel further however it may pickup more noise in certain conditions

The quality of your antennas at both ends - Antennas could be an entire article on their own but basically a larger antenna will send and receive a better signal. Often optimising your antenna placement will make a huge difference to the performance to the system. Although typical radio systems use the 2.4Ghz band, specialist long range systems such as the TBS Crossfire can run on much lower frequencies which are able to travel much further at the same power

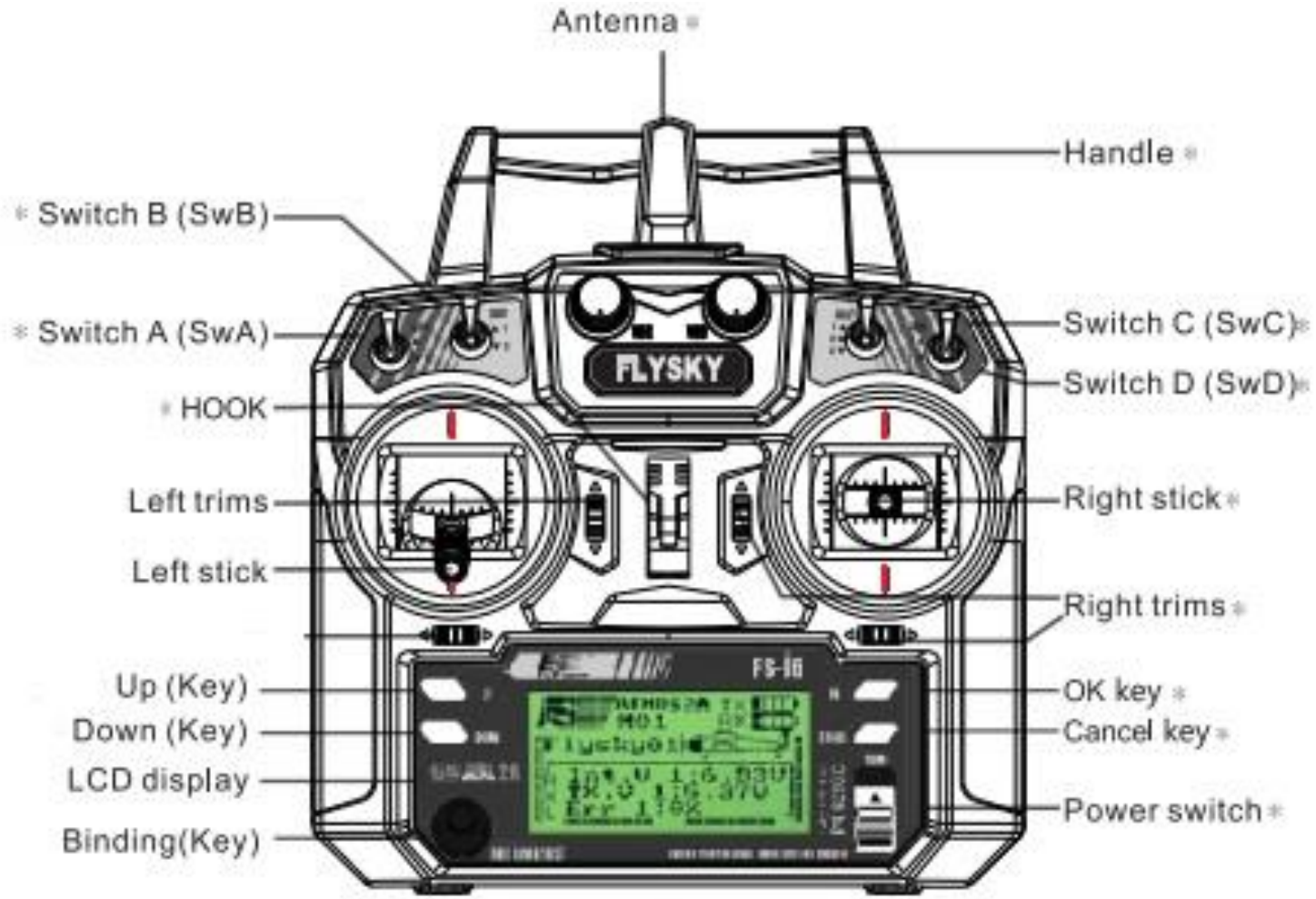
1.4.7.3 FlySky FS-i6

This is one of the cheapest remote currently available and for the price it is surprisingly solid. It can use a fast iBus protocol, features four configurable switches and is simple to operate. The range is reasonable but it lacks telemetry and defaults with only six channels. This remote requires 4xAA batteries to run which could get expensive. A lot of this remotes shortcomings can be easily fixed by some simple mods. Overall you can't fault this radio for the price point however if you stick with the hobby over a year you will most likely of reached it's limit be looking for a next level upgrade**.** OpenTX is a highly configurable system offering plenty of options for all types of RC models. It is created by FPV pilots with pilots in mind.

Wi-Fi and Bluetooth have made it possible for drone manufacturers to create device controls that run on smartphones or tablets. Not just a smartphone version of your RC transmitter, but high-tech controls**.**

1.4.7.4 Features:

- Works in the frequency range of 2.405 to 2.475GHz.This band has been divided into 142 independent channels ,each radio system uses 16 different channels and 160 different types of hopping algorithm.
- This radio system uses a high gain and high quality multi directional antenna, it covers the whole frequency band. Associated with a high sensitivity receiver, this radio system guarantees a jamming free long range radio transmission
- Each transmitter has a unique ID, when binding with a receiver the receiver saves that unique ID and can accepts only data from the unique transmitter. this avoids picking another transmitter signal and dramatically increase interference immunity and safety.
- This radio system uses low power electronic components and sensitive receiver chip. The RF modulation uses intermittent signal thus reducing even more power consumption.
- AFHDS2A system has the automatic identification function, which can switch automatically current mode between single-way communication mode and two-way communication mode according to the customer needs.


Antenna
Handle
Switch B (SwB)
Switch A (SwA)
Switch C (SwC)
Switch D (SwD)
HOOK
Left trims
Right stick
Left stick
Right trims
Up (Key)
Down (Key)
LCD display
Binding(Key)
OK key
Cancel key
Power switch
FLYSKY
FS-i6

1.4.8 Receiver:6 channel fsi6

A Radio Receiver is the device capable of receiving commands from the Radio Transmitter, interpreting the signal via the flight controller where those commands are converted into specific actions controlling the aircraft.

A Receiver must be compatible with the Radio Transmitter which in most cases means that the same brand of Rx and Tx needs to be purchased in order to establish a communication.

Frequencies must also be the same on both Rx and Tx. For instance; a 2.4GHz Transmitter can only work with 2.4GHz Radio Receiver.

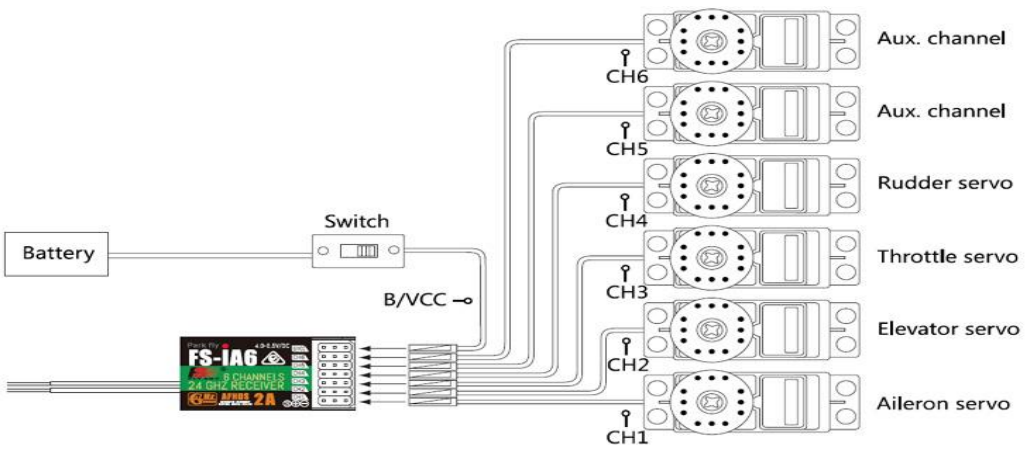
Thus, while selecting your Drone Transmitter and Receiver, it is important that they are compatible with each other in terms of frequency and other parameters. Also, it is necessary that both the components have the size and features according to the specifics needed. It is recommended to purchase a high-quality Radio transmitter with Receiver when starting out to fly your own drone as it is one of the components that will last long enough

CH1 to CH6: These channels are used to connect the ESCs (Electronic speed controller), Vcc, or other parts.

B/VCC: It is used to connect the binding cable for the binding receiver and transmitter.

1.4.8.1 FS-IA6B/FS-iA6 Receiver Connections

We connect motors with the receiver through ESCs (electronic speed controllers). The connection of the receiver is shown in the figure given below.



1.4.9 Propellers:

Propellers are devices that transform rotary motion into linear thrust. Drone propellers provide lift for the aircraft by spinning and creating an airflow, which results in a pressure difference between the top and bottom surfaces of the propeller. This accelerates a mass of air in one direction, providing lift which counteracts the force of gravity.

Propellers for multirotor drones such as hexacopter, octocopter and quadcopter propellers, are arranged in pairs, spinning either clockwise or anti-clockwise to create a balance. Varying the speed of these propellers allows the drone to hover, ascend, descend, or affect its yaw, pitch and roll.

Propeller speeds are varied by changing the voltage supplied to the propeller's motor, a process that is handled by an Electronic Speed Controller (ESC). The correct signal is fed to the ESC by the drone's flight controller, which relies on inputs from either the human pilot's controller or an autopilot, and may also take into account information from an IMU (Inertial Measurement System), GPS and other sensors.

1.4.9.1 UAV propeller dimensions and tradeoffs

Hybrid VTOL UAVs combine VTOL capability with the standard forward propulsion of a fixed wing UAV. In many hybrid VTOL UAVs, rotary lift propellers are typically incorporated into the aircraft's wings, which then transition for forward flight.

Drone propeller manufacturers usually specify two main measurements, quoted in the form A x B. The first number is the total length of the propeller from end to end. The second is the pitch, which is related to the angle of the propeller and is defined as how far the propeller will move forward under ideal conditions for every rotation. This can be thought of in a similar way to how far a screw will sink into a surface for every rotation of the screwdriver.

Shorter propellers require less energy to get up to a particular speed, and due to reduced inertia are easier to control and quicker to change speed. Longer propellers generate more lift for a particular RPM and create greater stability when hovering, but require more motor power.

Propellers with higher pitch will provide more lift than a flatter blade and allow a drone to fly faster for a particular RPM, but will drain the drone battery faster due to requiring more power from the motor.

Heavy-lift drones will typically require longer propellers with smaller pitch, as they require stability rather than speed, and will be able to carry larger batteries or power sources such as hydrogen fuel cells in order to offset the increased requirements.

1.4.9.2 Drone propeller construction

Drone propellers can be constructed with two, three, or four blades. Propellers with more blades provide greater lift due to more surface area moving through the air per rotation, but are more inefficient due to increased drag. Smaller drones with limited battery life are best suited to propellers with fewer blades.

Drone propeller blades are most commonly constructed from plastic or carbon fiber. Plastic propellers are cheaper and more flexible, allowing them to absorb impact better. The increased stiffness of carbon fiber propellers, although providing less durability, decreases vibration thus improving the flight performance of the drone and making it quieter. Carbon fiber is also lighter than plastic, allowing weight savings.

1.4.9.3 Description

The propeller 1045/1045R is a set of counter rotating RC aircraft propellers perfect for use in twin and multiple motor RC airplanes or coaxial RC helicopters. You receive 2 propellers: one prop rotates clockwise cw and the other prop rotates counter clockwise ccw (1 left hand & 1 right hand rotation). These matched rotor blades are a hard to find matched counter rotating pair. The length is 10 inches and the pitch is 4.5 inches per revolution.

- Diameter: 10in Pitch: 4.5in Propeller diameter: 25.4cm
- The adapters allow the propeller to be used with motors of different shaft diameters Weight: about 14g/pair Color: black, red, yellow, green, orange. Usage: for quadcopter and multirotor
- This propeller can be used with our A2212 1000KV, 1400KV, 1800KV and 2200kV motors and our 30A ESC

Chapter 2

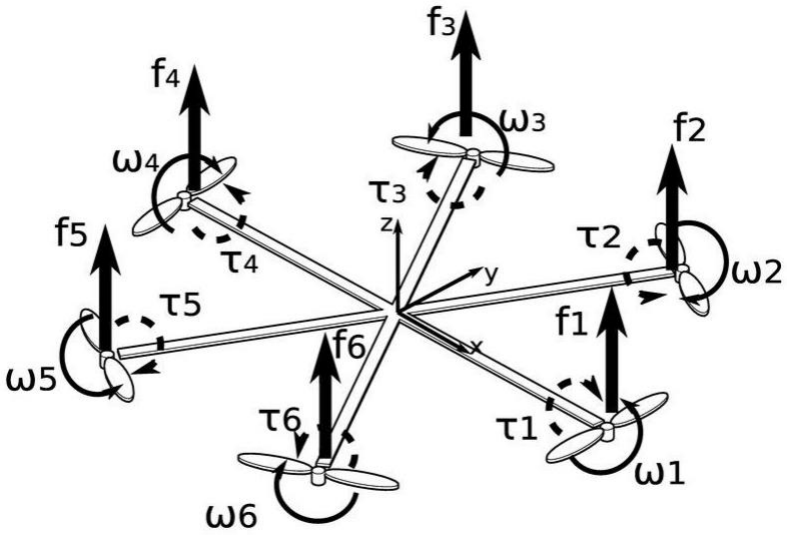
2.1 Methodology

2.1.1 Hexacopter construction:

2.1.1.1 SYSTEM MODELING

This section will deal with the methods used to build the mathematical model for the hexacopter. It deals with the coordinate frames chosen and the reference frame chosen to describe the dynamics of the hexacopter. The angular orientation of the aircraft is described by the three angles (Roll, Pitch, and Yaw) that govern the flight of the hexacopter, these are called the Euler Angles. These Euler angles represent an ordered set of sequential equations between the reference frame and the body frame.

2.1.1.2 Euler Angles

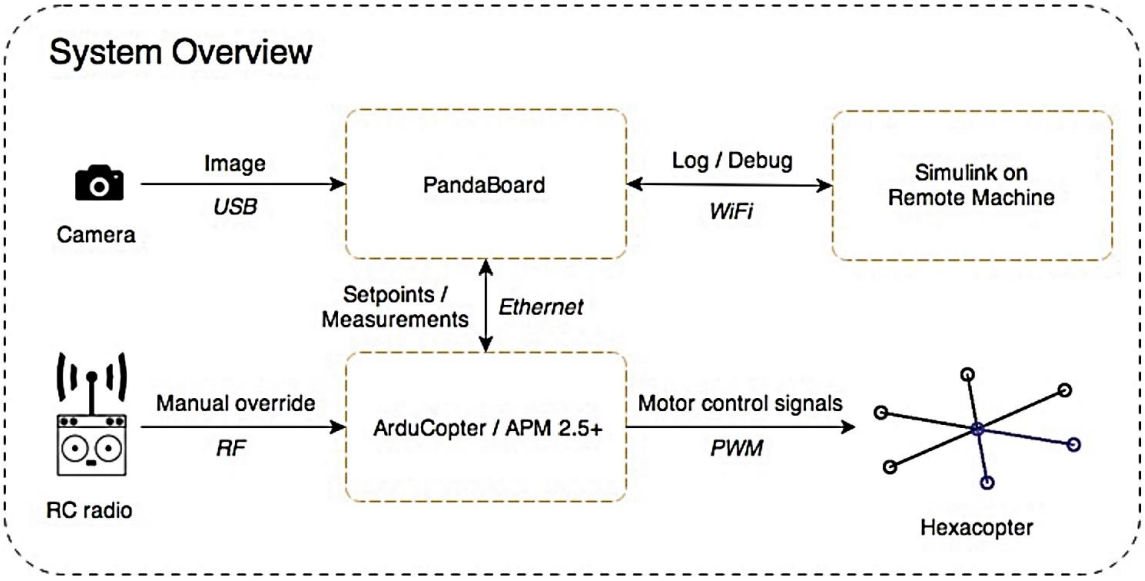


2.1.1.3 Schematic of a Hexacopter

Leonhard Euler described the Euler angles. These angles describe the angular orientation of a fixed body with respect to a reference frame. The Euler angles for the aircraft are Yaw (ψ), Pitch (ɵ), and Roll (ϕ). Yaw angle originates due to the center of gravity and is perpendicular to the wings. Pitch angle originates due to the center of gravity and goes parallel to the wingtip. Roll angle originates due to the center of gravity and goes parallel to the wings of the aircraft. The schematic of the hexacopter is presented below [1]. Two frames describe the motion of the hexacopter which are Inertial or Fixed frame and the Body frame. The motion of an aircraft is planned by geographical coordinates, so it is useful to define an earth-fixed frame tangent to the earth surface. The Euler angles Yaw (ψ), Pitch (ɵ), and Roll (ϕ) define the angular position of the Body frame with respect to the Inertial frame.

2.1.1.4 Hexacopter Construction

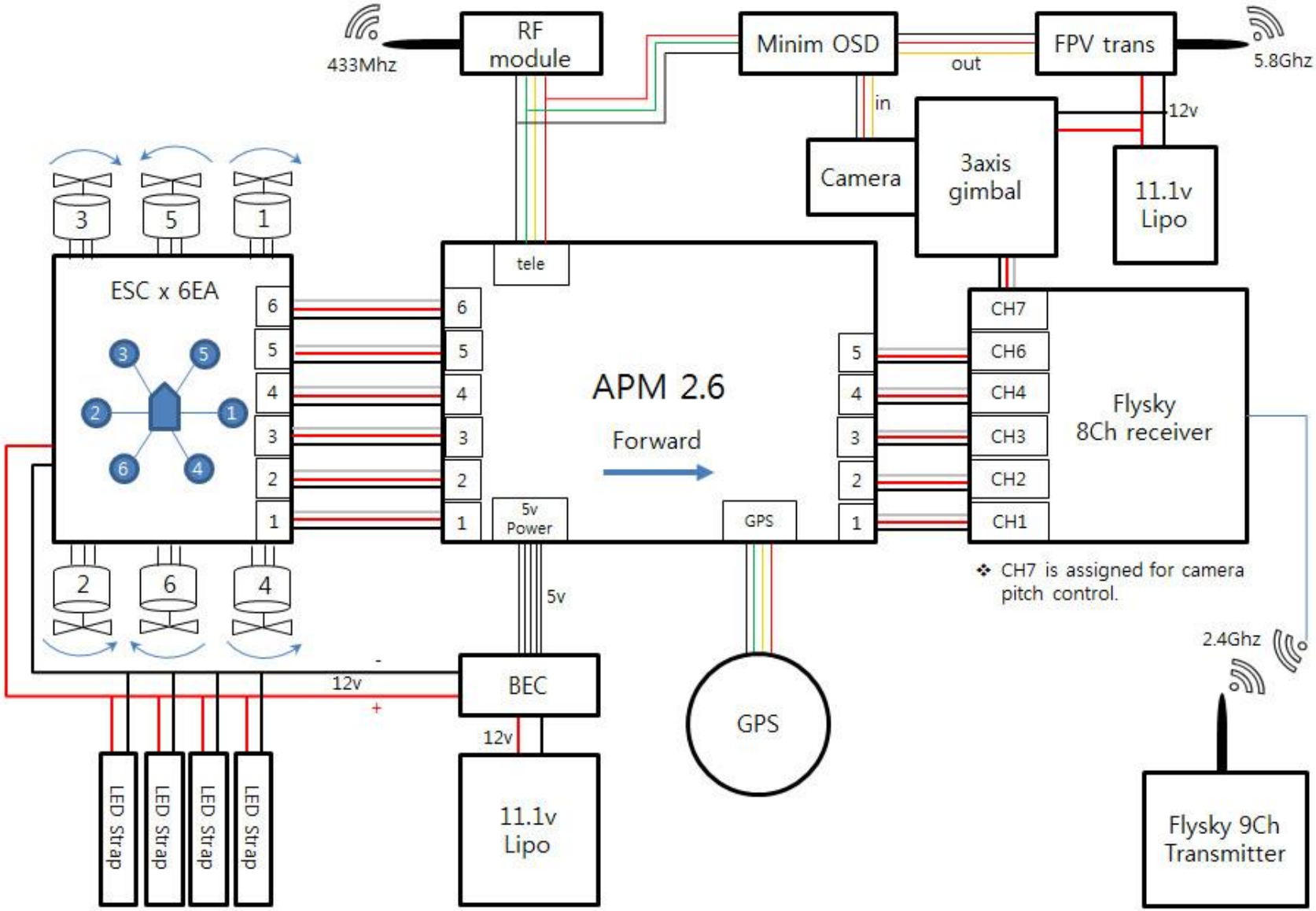
An overview of how the hardware is connected can be seen in Fig. 2.2.1, which is directly taken from [1]. The high level estimation and control is developed in Matlab Simulink and runs on the PandaBoard after automatic code generation into C-code. By using a WiFi-connection it is possible to download the model as well as interact with the running model by the External mode in Simulink. The PandaBoard is supplied with images from the camera via one of the USB ports and it also receives measurements and status data from the ArduCopter via Ethernet. The higher level position controllers on the PandaBoard communicate with the ArduCopter using the same Ethernet connection. The protocol used for the Ethernet connection is UDP. The PandaBoard is mounted on top of the hexacopter and has its own power distribution thanks to the work done by .The control system overview is presented in Fig. 2.2.2. The ArduCopter autopilot will be used for low level control like controlling the angular rotations and setting



motor outputs and also for providing sensor data used by the higher level estimation and control on the PandaBoard. The IMU, sonar and vision data is fused by using a nonlinear filter (unscented Kalman filter) and the estimates are supplied to the controllers that control the vertical and horizontal position of the hexacopter by sending attitude and throttle setpoints back to the ArduCopter autopilot.

The hexacopter position is controlled by sending attitude and thrust setpoints to the ArduCopter autopilot. These setpoints are the output of cascaded PID controllers.

Hexacopter components connections



Chapter 3

3.1 UAV communication

3.1.1 ZERO HOUR UAV COMMUNICATION

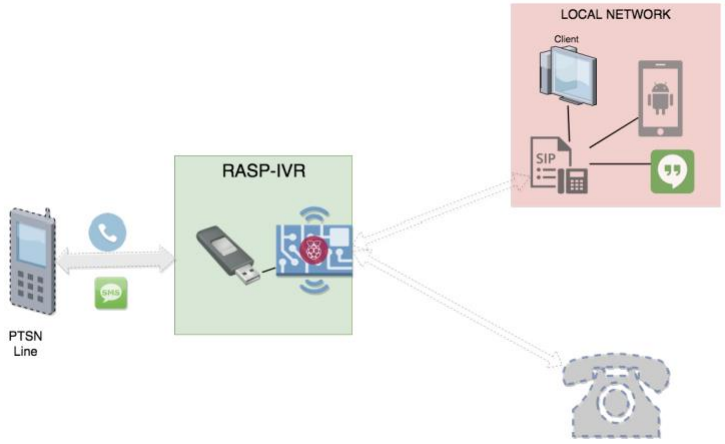
- Here Once the UAV reaches the specified location or Accident spot or the given latitude and longitude co-ordinates, the IVR code in Raspberry Pi starts to run, Here the latitude and longitude co-ordinates are fed in the raspberry Pi as well as the hospital list also will be fed in the raspberry Pi.
- Once the raspberry Pi co-ordinates and the Co-ordinate of the drone matches each other ,Now the Interactive voice response process is triggered .
- Now this IVR makes a call to the nearby Hospital line by line. If one hospital attends the call of IVR ,Then a sms will be sent to that hospital by using GSM sim card. Here In that SMS the latitude and longitude co-ordinate of the accident spot along with the local host link will be sent.
- Once the Doctor enters the Google Meet link then he/she can see the accident spot through the live streaming via the web cam available in the Raspberry Pi. Then the doctor can guide/Interact with the Public and suggest the steps that need to be taken as First aid.

3.1.2 RASP-IVR DESIGN

RASP-IVR was from our preliminary interviews with CBOs across the world interested in IVR technology. Firstly, our partners were concerned with the setup cost and were interested in a prototype before an investment in IVRs. Secondly, existing cloud systems allowed researchers to prototype applications but the power dynamic [27] was skewed towards researchers and away from participants. We reflected on these tensions [19] and saw an opportunity to build low-cost IVR systems. Lastly, following the vision of Brewer et al. [17] to promote open source in the development domain and from prior successes (such as Open Data Kit [18]

and Commcare) we have open-sourced our application. The “RASP-IVR” name was chosen because it was built on the Raspberry Pi, a low-cost, Linux-based, single board computer. The OS is based on RASPBX with a few customizations and libraries to better support IVR interactions. The system consists of a Raspberry Pi model 3 and a GSM mobile telephone modem. The GSM modem interfaces with an Asterisk server. The Asterisk server can be managed using FreePBX web interface to create and customize the IVR tree or voice over IP (VoIP) parameters. We have added scripts using open source libraries to extend RASPBX to support Voice messages, SMS, and local SIP transfers. Figure shows the working of RASP-IVR. The GSM modem interfaces with voice calls and text messages to connect it to an Asterisk server. The Asterisk server can connect the GSM call to the local or global VoIP network based on the requirements. The VoIP calls can be routed to various extension based on the clients touch tone input or answered through SIP clients like Yate, Zoiper among others using a desktop or a mobile device through a local SIP account. Additionally, the VoIP call can be transferred to external services like Twilio to use their cloud-based services e.g. Speech to text engine or cloud storage if needed. However, RASP-IVR is self-sufficient for participatory design or for small-scale deployments. RASP-IVR is a very low-cost system and it consumes very little power. Our system can be set up for $50, i.e. $35 for a raspberry PI and $15 for the GSM Modem. The system’s

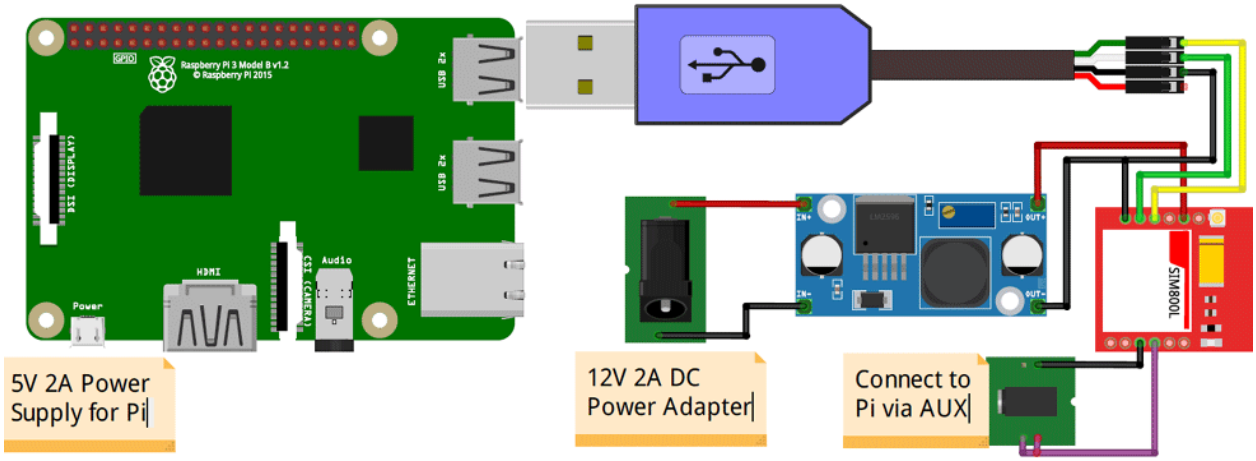
capability can be enhanced by connecting two or more modems allowing for more concurrent calls and call routing. Adding multiple modems may cause the modems to restart during high power consumption actions like calls. This limitation can be overcome using a low-cost USB hub which costs $10. The basic system costs $50 but the system can be augmented with low-cost tools to boost its capability based on context. For example, a 5V DC uninterruptible power supply costing $20 can be added to handle power outages. Developing countries are not well supported by cloudbased services and even when services are present, they fall



beyond the budget of local communities. RASP-IVR circumvents this issue by using local SIMs which allows the use of airtime bundles and existing SIM cards familiar to users. Table 1 shows the comparison of 1000 minutes of Twilio's cloud-based usage fees versus 1000 minutes of the default local calling costs to the RASP-IVR for a few countries in East Africa and we show the percentage savings. Using a plan or airtime bundle considerably lowers the calling costs further (e.g. see Tanzania), this can be capitalized to further increase savings. We profiled a RASP-IVR instance using automated calls using sipP. sipP [6] is a command line tool to simulate VoIP calls on so-called SIP servers. We set up a SIP channel on a RASP-IVR instance (on an MTN E153 modem and Raspberry PI 3) and tested the system using a sipP script. We ran a simulation for 10 calls per second, 20 concurrent calls, and 100 total calls and measured the response rates. A test call

dialed into a SIP channel and listened to a recorded message, We found that RASP-IVR successfully handled all 100 of the calls with no failure in 2 mins. The system handled 0.828 calls/second. As the experiment progressed, the response times of calls converged towards 0.015 seconds. Figure 3 shows the plot of response times over the duration of the

3.1.3 Circuit Diagram



3.1.4 Components

- Raspberry Pi (2 or 3 or 4) running latest version of Buster
- SIM800L GSM Module
- 2G SIM Card (Airtel)
- Aux Cable
- LM2596 Buck Converter Module
- USB to TTL Converter
- 12V 2A Adapter
- Perf Board
- Berg Sticks
- Connecting wires
- Soldering Kit

Powering the SIM800L Module: The most important thing to note here is how you power the SIM800L module. The SIM800L module operated between 3.7V to 4.2V (designed for Li-po batteries). The ideal operating voltage is around 4V. We have used LM2596 Buck converter to convert the 12V 2A input from the adapter to 4V as required by the SIM800L module. Care should be taken that the wires connecting the buck module with SIM800L are thick and short to carry high current easily. If your power connections are not made adequate, the module will reset itself when powered and will throw garbage valued on serial communication. So, make sure the SIM800L module is powered with a proper 12V 2A adapter and the wires are thick enough to carry high current without providing much resistance. If you face any problem, directly connect a fully charged Li-po battery to the Vcc and Gnd pins of SIM800L module, this will solve the power issues for sure.

- **Serial Communication between SIM800L and Raspberry Pi:** "AT Commands" to the SIM800L module. AT commands can be used to make/receive calls, send/read messages, detect a keypress, and more. In this project, we will be using Python from Raspberry Pi to send these AT commands to the GSM SIM800L module. To do that, we have used a USB to TTL converter to connect the Rx and TX pin of the SIM800L module to the USB port of the Raspberry Pi.

- **Audio input to SIM800L module from Raspberry Pi:** The SIM800L module supports microphone input through the MIC+ and MIC- pins on the module. Once the call is made, any audio input given to these pins will be played on the call receiver's phone. However, these pins are actually meant to connect a microphone and, in our case, we need to play a pre-recorded voice from the Raspberry. The audio output from the Pi through the 3.5mm jack is called line-out audio and we need to convert it to mic level audio to be able to receive it on the GSM module. Now, you can build a line to mic converter circuit to get this done professionally, but I just connected them directly and reduced the volume level on the Pi to just 2 points. I got it working this way without any problem.

- I built the complete circuit on a perf board and made sure the power connections had enough lead to provide low resistance connectivity. My board when soldered looks like this.

- **Important:** LM2596 is a variable buck converter so before you use it, make sure you set the output voltage to 4V using the on-bord trimpot. Anything more than 4.2V can kill the SIM800L module permanently.

- Once you reach here, just power the board and insert the SIM card. Again make sure you have inserted the SIM card in the right orientation and the antenna is properly installed. If everything is done right you should notice the onboard led on

the SIM800L board blinking once every 3 seconds. This means that the SIM800L module is able to set-up a network with our SIM card.

- This ensures that our powering circuit is working properly. Now, we can connect the board without Raspberry Pi and start writing our Python script.

3.1.5 Raspberry Pi Python Code to Make Calls and Send SMS using SIM800L Module:

We start the program by importing the required header files. We have used the serial package to enable serial communication between Pi and SIM800L, the pygame package is used to play music, in our case, play the audio files. Apart from that, we have the time package to create delays. All three packages are pre-installed in the Buster OS, so you need not install anything new.

```
import serial #for serial communication with GSM SIM800L
import time
import pygame #to play music
```

3.1.5.1 def SIM800 (Command): This function is used to send AT command from PI to SIM800L and get a response for that AT command. All the AT commands sent to SIM800L should end with "\r\n" and should be encoded in ASCII. This function appends all our AT commands with "\r\n" and encodes them into ASCII form before sending them to SIM800L. Then it also reads the response and decodes the ASCII values so that we can use them in our program.

```
#Speak with SIM800 -> gets AT command return as response
def SIM800(command):
    AT_command = command + "\r\n"
    ser.write(str(AT_command).encode('ascii'))
    time.sleep(1)
    if ser.inWaiting() > 0:
        echo = ser.readline() #waste the echo
        response_byte = ser.readline()
        response_str = response_byte.decode('ascii')
        return (response_str)
    else:
        return ("ERROR")
```

3.1.5.2 def wait_for_SIM800(): This function is also very similar to the above function, but it does not send any value to SIM800, it just waits for the SIM800L to respond with something. When it gets a reply it returns it as a result.

```
#checks if SIM800L is speaking and returns it as response
def wait_for_SIM800():
    echo = ser.readline()  # waste the echo
    response_byte = ser.readline()
    response_str = response_byte.decode('ascii')
    return (response_str)
```

3.1.5.3 def Init_GSM(): The initialize GSM function prepares the GSM module for IVR operations. It first checks for the module by sending an "AT" and waiting for an "OK" and then sends some specific AT commands to put the GSM module in messaging mode and DTMF receiver mode. Apart from that, it also disables all notifications so that we won't be disturbed by text message notifications during a call.

```
#Checks SIM800L status and connects with ShopifyAPI
def Init_GSM():
    if "OK" in SIM800("AT"):
        if ("OK" in (SIM800("AT+CLCC=1"))) and ("OK" in (SIM800("AT+DDET=1"))) and
("OK" in (SIM800("AT+CNMI =0,0,0,0,0"))) and ("OK" in (SIM800("AT+CMGF=1"))) and
("OK" in (SIM800("AT+CSMP=17,167,0,0"))):  # enble DTMF / disable notifications
            print("SIM800 Module -> Active and Ready")
    else:
        print("------->ERROR -> SIM800 Module not found")
```

3.1.5.4 def play_wav (file_name): The next function is used to play the *wav files* once the call has been answered. We have stored pre-recorded *wav files* like, "*intro.wav*", "*confirm.wav*", "*cancel.wav*" etc. We have to play each of these files based on the keypad response from the user. This play_wav function can be used to play any *wav file* of our choice. Make sure these *wav files* are saved in the same directory of your Python code.

```
#plays the given wav file #8000Mhz mono audio WAV works best on SIM800L
def play_wav(file_name):
    pygame.mixer.init(8000)
    pygame.mixer.music.load(file_name)
    pygame.mixer.music.play()
```

```
    #while pygame.mixer.music.get_busy() == True:
        #continue
```

3.1.5.5 def Call_response_for (phone_number): This is the most important function in the program. It gets the phone_number to which a call has to be made and provides the response for that call. The response can be anything like *NOT_REACHABLE*, *CALL_REJECTED*, *CONFIRMED*, *CANCELED*, etc. The function uses AT commands to make the call to a given phone number and plays the recorded voice. Then it checks for the DTMF response from the caller and based on the response it plays the relative recorded voice and finally tells us what the caller has selected. If the caller rejected the call or was not reachable, the function will also provide that as a response.

```
# Makes a call to given number and returns NONE, NOT_REACHABLE, CALL_REJECTED, REJECTED_AFTER_ANSWERING,  REQ_CALLBACK,CANCELED, CONFIRMED
def Call_response_for (phone_number):
    AT_call = "ATD" + phone_number + ";"
    response = "NONE"
    time.sleep(1)
    ser.flushInput() #clear serial data in buffer if any
    if ("OK" in (SIM800(AT_call))) and (",2," in (wait_for_SIM800())) and (",3," in (wait_for_SIM800())):
        print("RINGING...->", phone_number)
        call_status = wait_for_SIM800()
        if "1,0,0,0,0" in call_status:
            print("**ANSWERED**")
            ser.flushInput()
            play_wav("intro.wav")
            time.sleep(0.5)
            dtmf_response = "start_over"
            while dtmf_response == "start_over":
                play_wav("press_request.wav")
                time.sleep(1)
                dtmf_response = wait_for_SIM800()
                if "+DTMF: 1" in dtmf_response:
                    play_wav("confirmed.wav")
                    response = "CONFIRMED"
                    hang = SIM800("ATH")
```

```
                    break
                if "+DTMF: 2" in dtmf_response:
                    play_wav("canceled.wav")
                    response = "CANCELED"
                    hang = SIM800("ATH")
                    break
                if "+DTMF: 9" in dtmf_response:
                    play_wav("callback_response.wav")
                    response = "REQ_CALLBACK"
                    hang = SIM800("ATH")
                    break
                if "+DTMF: 0" in dtmf_response:
                    dtmf_response = "start_over"
                    continue
                if "+DTMF: " in dtmf_response:
                    play_wav("invalid_input.wav")
                    dtmf_response = "start_over"
                    continue
                else:
                    response = "REJECTED_AFTER_ANSWERING"
                    break
        else:
            #print("REJECTED")
            response = "CALL_REJECTED"
            hang = SIM800("ATH")
            time.sleep(1)
            #ser.flushInput()
    else:
        #print("NOT_REACHABLE")
        response = "NOT_REACHABLE"
        hang = SIM800("ATH")
        time.sleep(1)
        #ser.flushInput()
    ser.flushInput()
```

```
    return (response)
```

3.1.5.6 def send_message (message, recipient): Apart from making calls and getting the response, this program also allows us to send a message, the send_message function is used to do exactly that. It gets the message and recipient's phone number and sends the message.

```
#Receives the message and phone number and send that message to that phone number
def send_message(message, recipient):
    ser.write(b'AT+CMGS="' + recipient.encode() + b'"\r')
    time.sleep(0.5)
    ser.write(message.encode() + b"\r")
    time.sleep(0.5)
    ser.write(bytes([26]))
    time.sleep(0.5)
    print ("Message sent to customer")
    time.sleep(2)
    ser.flushInput()  # clear serial data in buffer if any
```

3.1.5.7 def incoming_call(): The last function I made for this project is to detect the caller number of an incoming call. Since this SIM card is going to make calls to new people, some might try to call back to this number. In that case, this function can be used to check from which number we are receiving an incoming call and then later send a message or call them back as required.

```
def incoming_call():
    while ser.in_waiting: #if I have something in the serial monitor
        print ("%%Wait got something in the buffer")
        ser.flushInput()
        response = SIM800("ATH") #cut the incoming call
        if "+CLCC" in response:
            cus_phone = response[21:31]
            print("%%Incoming Phone call detect from ->", cus_phone)
            return (cus_phone)
        else:
            print("Nope its something else")
```

```
            return "0"
    return "0"
```

Now that all the functions are defined, it's time to write the main code in which we will use all these functions to do something beautiful. Now for demonstration purpose, I am just going to hard code the customer name and customer phone number, but you can get it from an Shopify API call or read from a spreadsheet as required. I am entering the customer name as “AISHA” and number as “9877XXXXXX” for testing

```
cus_name = "Aisha"
cus_phone = "968837XXXX"
```

Inside the main infinite while loop, we will be starting a serial communication at 9600 baud rates with 15 seconds time out. Now, some SIM800L modules might work in different baud rates, make sure you enter the right COM directory and baud rate here.

```
# COM defanition for windows -> Should change for Pi
ser = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=15)   # timeout affects call duration and waiting for response currently 30sec
print("Established communication with", ser.name)
```

Next, we will make the phone call and get the required response from the customer, then based on the response, we will send a message to the customer. For example, if the response is confirmed, we will send a message regarding that, similarly, we can change the message for a different response from the customer.

3.1.6 Code

```
import serial #for serial communication with GSM SIM800L
import time
import pygame #to play music
# ____________________________________________________________________________#
# Intro text
print("Setting up Raspberry PI IVR")
#Speak with SIM800 -> gets AT command return as response
def SIM800(command):
    AT_command = command + "\r\n"
    ser.write(str(AT_command).encode('ascii'))
```

```
    time.sleep(1)
    if ser.inWaiting() > 0:
        echo = ser.readline() #waste the echo
        response_byte = ser.readline()
        response_str = response_byte.decode('ascii')
        return (response_str)
    else:
        return ("ERROR")
#checks if SIM800L is speaking and returns it as response
def wait_for_SIM800():
    echo = ser.readline()  # waste the echo
    response_byte = ser.readline()
    response_str = response_byte.decode('ascii')
    return (response_str)
#Checks SIM800L status and connects with ShopifyAPI
def Init_GSM():
    if "OK" in SIM800("AT"):
        if ("OK" in (SIM800("AT+CLCC=1"))) and ("OK" in (SIM800("AT+DDET=1"))) and
("OK" in (SIM800("AT+CNMI =0,0,0,0,0"))) and ("OK" in (SIM800("AT+CMGF=1"))) and
("OK" in (SIM800("AT+CSMP=17,167,0,0"))):  # enble DTMF / disable notifications
            print("SIM800 Module -> Active and Ready")
    else:
        print("------->ERROR -> SIM800 Module not found")
#plays the given wav file #8000Mhz mono audio WAV works best on SIM800L
def play_wav(file_name):
    pygame.mixer.init(8000)
    pygame.mixer.music.load(file_name)
    pygame.mixer.music.play()
    #while pygame.mixer.music.get_busy() == True:
        #continue
# Makes a call to given number and returns NONE, NOT_REACHABLE, CALL_REJECTED,
REJECTED_AFTER_ANSWERING,  REQ_CALLBACK,CANCELED, CONFIRMED
def Call_response_for (phone_number):
    AT_call = "ATD" + phone_number + ";"
    response = "NONE"
```

```
    time.sleep(1)
    ser.flushInput() #clear serial data in buffer if any
    if ("OK" in (SIM800(AT_call))) and (",2," in (wait_for_SIM800())) and (",3," in
(wait_for_SIM800())):
        print("RINGING...->", phone_number)
        call_status = wait_for_SIM800()
        if "1,0,0,0,0" in call_status:
            print("**ANSWERED**")
            ser.flushInput()
            play_wav("intro.wav")
            time.sleep(0.5)
            dtmf_response = "start_over"
            while dtmf_response == "start_over":
                play_wav("press_request.wav")
                time.sleep(1)
                dtmf_response = wait_for_SIM800()
                if "+DTMF: 1" in dtmf_response:
                    play_wav("confirmed.wav")
                    response = "CONFIRMED"
                    hang = SIM800("ATH")
                    break
                if "+DTMF: 2" in dtmf_response:
                    play_wav("canceled.wav")
                    response = "CANCELED"
                    hang = SIM800("ATH")
                    break
                if "+DTMF: 9" in dtmf_response:
                    play_wav("callback_response.wav")
                    response = "REQ_CALLBACK"
                    hang = SIM800("ATH")
                    break
                if "+DTMF: 0" in dtmf_response:
                    dtmf_response = "start_over"
                    continue
```

```
                if "+DTMF: " in dtmf_response:
                    play_wav("invalid_input.wav")
                    dtmf_response = "start_over"
                    continue
                else:
                    response = "REJECTED_AFTER_ANSWERING"
                    break
        else:
            #print("REJECTED")
            response = "CALL_REJECTED"
            hang = SIM800("ATH")
            time.sleep(1)
            #ser.flushInput()
    else:
        #print("NOT_REACHABLE")
        response = "NOT_REACHABLE"
        hang = SIM800("ATH")
        time.sleep(1)
        #ser.flushInput()
    ser.flushInput()
    return (response)
#Receives the message and phone number and send that message to that phone number
def send_message(message, recipient):
    ser.write(b'AT+CMGS="' + recipient.encode() + b'"\r')
    time.sleep(0.5)
    ser.write(message.encode() + b"\r")
    time.sleep(0.5)
    ser.write(bytes([26]))
    time.sleep(0.5)
    print ("Message sent to customer")
    time.sleep(2)
    ser.flushInput()  # clear serial data in buffer if any
def incoming_call():
    while ser.in_waiting: #if I have something in the serial monitor
```

```
        print ("%%Wait got something in the buffer")
        ser.flushInput()
        response = SIM800("ATH") #cut the incoming call
        if "+CLCC" in response:
            cus_phone = response[21:31]
            print("%%Incoming Phone call detect from ->", cus_phone)
            return (cus_phone)
        else:
            print("Nope its something else")
            return "0"
    return "0"
cus_name = "Aisha"
cus_phone = "96883XXXXX"
while (1): #Infinite loop
    # COM defanition for windows -> Should change for Pi
    ser = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=15)  # timeout affects
call duration and waiting for response currently 30sec
    print("Established communication with", ser.name)
    Init_GSM() #check if GSM is connected and initialize it
    print("_____________________IVR START_____________________")
    response = Call_response_for(cus_phone) #place a call and get response from
customer
    print ("Response from customer => ", response)
    if response == "CONFIRMED":
        text_message = "Hi " + cus_name + ". Your booking has been confirmed. Thank
you!!. -Circuitdigest"
        send_message(text_message, cus_phone)
    if response == "CANCELED":  # if the response was to cancel
        text_message = "Hi " + cus_name + ". Sorry that you have decided to cancel
your booking. If you cancled by mistake, kindly contact us through phone. -
Circuitdigest"
        send_message(text_message, cus_phone)
    if ((response == "CALL_REJECTED") or (response == "REJECTED_AFTER_ANSWERING")):
# if the response was rejected
        text_message = "Hi " + cus_name + ". We from circuitdigest.com have been
trying to reach you, to confirm your booking. You will receive another call within
few minutes, we kindly request you to answer it. Thank you"
```

```
        send_message(text_message, cus_phone)
    print("_____IVR END_____")
    ser.close()
    time.sleep (5)
```

Picture of Zero Hour UAV

Chapter 4

Conclusion

This project is helpful in providing necessary medicines in areas where normal traffic transportations services are not & also in regions where the geographical terrain is not fit for traditional transportation methods. Secondly, crucial applications comes in emergency situations like floods earthquake etc. where the resident and doctors need vital medicines which can be delivered easily via our medicine drone delivery system. Thirdly, this Drone comes handy in cities also. The rising population and tremendous increase in private vehicles on city roads have increased traffic congestions making it difficult for the traditional delivery systems to function effectively. Hence also this UAV provides real time support to patient and the doctors as well, This drone provide live steaming where the doctor can interact with patient and the patient can also interact with doctor through the web cam in the drone. And also monitor the patient health through the sensors that are present in the UAV.

REFERENCES

[1] V. Artale, C. Milazzo and A. Ricciardello, "Mathematical modeling of hexacopter", Applied

Mathematical Sciences, vol. 7, pp. 4805-4811, 2013.

[2] J. Ligthart, P. Poksawat, L. Wang and H. Nijmeijer, "Experimentally Validated Model

Predictive Controller for a Hexacopter. The authors gratefully acknowledge the partial

sponsorship by DSTG, Australia.", IFAC-PapersOnLine, vol. 50, no. 1, pp. 4076-4081, 2017.

[3] M. Moussid, A. Sayouti and H. Medromi, "Dynamic Modeling and Control of a HexaRotor

using Linear and Nonlinear Methods", International Journal of Applied Information Systems,

vol. 9, no. 5, pp. 9-17, 2015.

[4] A. Alaimo, V. Artale, C. Milazzo, and A. Ricciardello, "PID Controller Applied to

Hexacopter Flight", Journal of Intelligent & Robotic Systems, vol. 73, no. 1-4, pp. 261-270,

2013.

[5] Morales, Camilo, Diana Ovalle, and Alain Gauthier. "Hexacopter maneuverability

capability: An optimal control approach." Modeling, Simulation, and Applied Optimization

(ICMSAO), 2017 7th International Conference on. IEEE, 2017.

[6] Falconí, Guillermo P., Christian D. Heise, and Florian Holzapfel. "Fault-tolerant position

tracking of a hexacopter using an Extended State Observer." Automation, Robotics, and

Applications (ICARA), 2015 6th International Conference on. IEEE, 2015.

[7] Ahmed, O. A., et al. "Stabilization and control of autonomous hexacopter via visual servoing and cascaded-proportional and derivative (PD) controllers." Automation, Robotics,

and Applications (ICARA), 2015 6th International Conference on. IEEE, 2015.

[8] Baránek, Radek, and František Šolc. "Modeling and control of a hexacopter." Carpathian

Control Conference (ICCC), 2012 13th International. IEEE, 2012.

55

[9] Leishman, Robert, et al. "Relative navigation and control of a hexacopter." Robotics and

Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012.

[10] E. Bekir, Introduction to modern navigation systems. Singapore: World Scientific, 2007.

[11] D. Küchemann, Progress in aerospace sciences, vol. 12. Pergamon Pr., 1972.

[12] D. Wang, "Quadcopter Parts: What are they and what do they do? – Quadcopter

Academy", Quadcopter Academy, 2017. [Online]. Available:

http://www.quadcopteracademy.com/quadcopter-parts-what-are-they-and-what-do-they-do/.

[Accessed: 24- Nov- 2017].

[13] "Components of a Multirotor - Drones and Multirotors: Types, Components &

Uses", Sites.google.com, 2017. [Online]. Available:

https://sites.google.com/a/ewg.k12.ri.us/drones-types-components-and-uses/home/components of-a-multirotor. [Accessed: 24- Nov- 2017].

[14] Ae01.alicdn.com, 2017. [Online]. Available:

https://ae01.alicdn.com/kf/HTB1YRgDKpXXXXaPXXXXq6xXFXXXu/Hexacopter-Drone Brushless-Motor-450-Rc-F450-Quadcopter-Motor-Motors-Diy-Quadcopter-Drone-Parts-Set Part.jpg_640x640.jpg. [Accessed: 26- Nov- 2017].

[15] I2.wp.com, 2017. [Online]. Available:

https://i2.wp.com/www.americandronesonline.com/wp

content/uploads/2016/07/182435a99cb1.jpg?fit=500%2C343 [Accessed: 26- Nov- 2017].

[16] Pisces.bbystatic.com, 2017. [Online]. Available:

https://pisces.bbystatic.com/image2/BestBuy_US/images/products/4872/4872301_sd.jpg;maxH

eight=640;maxWidth=550. [Accessed: 26- Nov- 2017].

[17] P. Michal Mazur, "Six Ways Drones Are Revolutionizing Agriculture", MIT Technology