Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

PI Drone using Python

Prasant Chettri, Rajni Giri, Zerong Lepcha, Arvind Lal
Department of Computer Science and Technology
Centre for Computer and Communication Technology (CCCT), Chisopani, Sikkim, India

Abstract **- The drone is commonly known as Unmanned Aerial Vehicles (UAVs). In today's world drone is extensively used in every field, some of the common application of drone is now being used for the Precision Agriculture, Search and Rescue, Wildlife Monitoring, Entertainment and others fields. As the whole world was facing a fatal pandemic situation, drone surveillance becomes an accommodating technology for mankind. The drone brings a technology revolution to the global market. Drones are becoming profuse evolve technology in today's aeronautics, as like robotics. Every drone marketing company is focusing on AI (Artificial Intelligent) or autonomous flights system, but still, it requires human interactions. It was mainly controlled by remotes sensing technology, even with only hardware it is not possible to fly (UAVs) drones, we must necessitate its software and programming. The mathematical and physics law application makes it possible for us to fly drones, but it is required much accuracy for stabilized hovering. Here we are using different platforms such as ROS, Linux, Gazebo (modelling or simulating), Python, C++, to develop obstacle avoidance and keyboard control features in UAVs.**

Keywords- UAVs, Obstacle Avoidance, Lidar, SITL, EKF 2 ROS, GPS

1. INTRODUCTION

UAVs is broadly appropriating in search and rescue operations, military surveillance and civilian professions because of their prominent advantages, such as flexibility, light mass, stable mobility, and good concealment. Nowadays, the growth and applications of UAV technologies only not change the evolving area of many industries but also brings market and economic benefits. The autonomy level of UAVs varies according to the tasks at hand or the degree to which the vehicle can make decisions without being explicitly guided by a remote operator. Usually, drones use various classification on-board sensors that can manage situational consciousness and autonomous decision-making at run-time. Obstacle avoidance leads to the approach of moulding the robot's pathway to overcome unforeseen obstacles. The resulting movement depends on the drones' existent position and the sensor interpretations. There is a level of algorithms for restriction avoidance from basic re-planning to reactive rotation in the controller strategy. Advanced techniques vary on the execution of sensor data and on the motion control approaches to overcome obstacles. The most generous difficulty of autonomous drones is capable to react accurately and safely to the circumstances during flight. Consequently, these vehicles must be equipped with sensors proficient in respondents very instantly and processing the system intelligently interpreting all this data in real-time. This paper comprised III sections Hardware integration, Software integration and Modelling & Avoidance Analyses. The hardware platform used in this work is Raspberry pi & Navio2. In this, advanced environment ROS Gazebo is a dynamic 3D simulated environment for autonomous vehicles that are especially suitable for examination (OA) systems. Gazebo used with Software in Loop(SIL) and Hardware in Loop(HIL) design. These works are based on real-time detection using an RP Lidar sensor and performing modelled Lidar in a ROS gazebo simulation environment. Section C involves several embedded applications to process the detection sensor to command the navigation, adorned in pragmatic conditions.

A. System Hardware Architecture

1) The Raspberry pi and Navio2 UAVs System

An autonomous drone requires at least two levels of control to operate Inner Loop and Outer Loop. The Inner Loop stabilizes the vehicle at a desired angle or body motion. A board that controls the Inner Loop is a flight controller Navio2 that allows pilots to communicate the requested vehicle state, and output should stabilize commands to a motor to achieve that state. The Outer Loop generates an angle or rate of instruction to get the drone from point A to B. Raspberry Pi flight-controller is used to conducting all other outer loop control and request vehicle state to the flight controller. It is a decision making brain of a drone that request, as performed by various on-board sensor data, combines and specific task it was carrying out. GPS data tells the flight computer where it is in space and about the waypoint mission. The Navio2 eradicates several needs to become various controllers on board as everything is compressed within one (including the Raspberry Pi), consequently enhancing the robustness of our project and expediting the community. With the Navio2, we can control our flying robots such as multirotor and planes. The Navio2 is adorned with a High-resolution barometer double IMU and GNSS receiver with GPS, GLONASS, Beidou, Galileo and SBAS satellites for exact positioning and adjustment.
The Navio2 is executing a data processing technique for navigating to estimate the drone position and visual estimation. The method we are using for path planning and obstacle avoidance is SLAM (simultaneous localization and mapping) is a procedure used for an autonomous drone that allows us to build a map and localize our drone to map in a real-time process. SLAM algorithm allowing the UAVs to map out unrevealed environments. There exist different SLAM methods but, we are working with the LiDAR SLAM system. Usually, LIDAR SLAM is used with a laser sensor to produce a 3D map of its environment. LiDAR (Light Detection and Ranging) estimates the range of an object by locating the nearby object using an active laser 'pulse'. It is working with a swift and accurate classification. So, it can work in a comprehensive range of environments and conditions.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

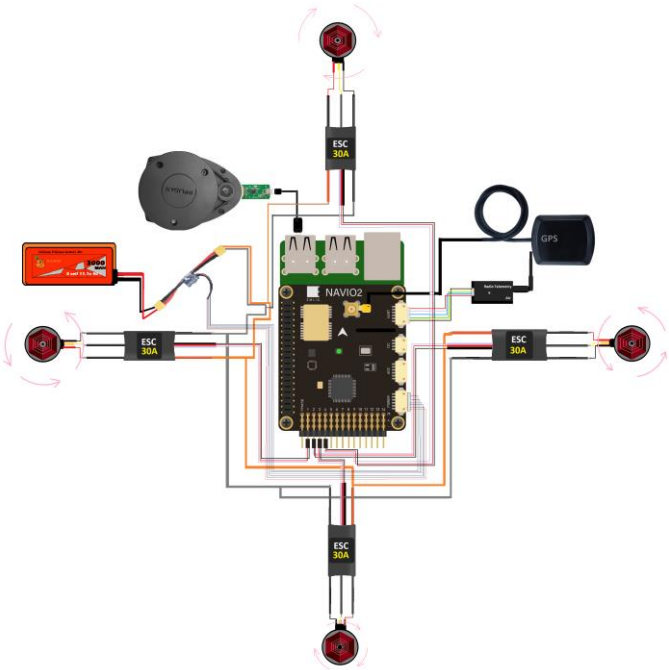


Fig 1. Hardware integration

(a) Attitude and Heading Reference System

AHRS works as a stabilization and motion control. It computes a central processing unit (CPU) to the IMU, the AHRS features state algorithms for a wide range of dynamic motions of UAVs. It comprises IMU 3-axis gyroscopes, accelerometers, and magnetometers that support wing, heading, pitch and roll data, also providing high-frequency real-time UAVs rotation data and open-ended gyroscope fixings. These sensors are attached to an external GNSS receiver to enhance its execution. It implants the Extended Kalman Filter that produces position and heading information.

(b) Inertial Navigation System (INS)

INS is a computer system that takes input from a global navigation satellite system such as GPS or Inertial Measurement Unit (IMU), which measures the Payload Stabilization & Orientation of the UAVs. The IMU outputs gyroscopes, accelerometers, and magnetometers raw data, and the Inertial Navigation System (INS) additionally implant a GNSS receiver for a clarified attitude in real-time. The inertial navigation system is a self-esteem process through which a system may track its attitude, orientation and velocity (once provided initial values for these parameters) without the necessity of such outer implications. If the acceleration of an object is classified, it is probable to use numerical integration to estimate the velocity.

(c) Extended Kalman Filter(EKF)

EFK is the nonlinear version of the Kalman filter that linearizes the estimation of the current mean and covariance. This algorithm evaluates vehicle position, velocity and angular orientation based on rate accelerometer, gyroscope, GPS, airspeed and barometric pressure estimations. This nonlinear state-space model illustrates the relationship state to sensors and the relationship states to state time derivatives.

$$X_{k_i} = F_k(x_{k_i}, u_k) + \epsilon \quad \rightarrow eq(1)$$

$$Y_{k_i} = H_k(x_{k_i}, u_k) + \eta \quad \rightarrow eq(2)$$

x_{k_i} is a vector of a state that is estimated by the EKF, y_k is a vector of the measurement and u_k is the vector of predetermined inputs to EFK. η measurement noise vector and ϵ state noise vector.

The variance of error state using covariance matrix is follow as M_k as shown in eq(3).

$$M_k = i\left[(x_k - x_{k_i})(x_k - x_{k_i})^T\right] \quad \rightarrow eq(3)$$

x_k represent state vector which is the output of EKF. x_{k_i} this represents the expected relationship to the measurement provided in equation (1) & equation (2), $M_k$is the covariance matrix state in EKF method

(d) Measurement Update

The process of updating the measurement vector y_k is called measurement update. This can be performed using the Kalman gain matrix. Here, t is a current update

$$H_{k_t} = H_k(x_{k_{t-1}}, u_{k_t})$$

$$G_{k_t} = M_{k_{t-1}} H_{k_t}^T (H_{k_t} M_{k_{t-1}} H_{k_t}^T + S_k)^{-1} \quad \rightarrow eq(4)$$

s_k is the matrix of expected sensor noise of measurement and the $H_k$is the matrix that maps the states and input of an EKF to the measurement of the EKF methods, t-1 on tensor indicate using a value from previous (KF) update of that tensor. The KGM G_{k_t} for this update is used to compute the update state estimate $x_{k_t}$and update M_{k_t} as shown in equations (5)&(6).

$$X_{k_t} = X_{k_{t-1}} + G_{k_t}\left[y_k - \frac{\delta H_k}{\delta X_k} x_{k_{t-1}}\right] \rightarrow eq(5)$$

$$M_{k_t} = (I - G_k \frac{\delta H_{k_t}}{\delta x_{k_t}}) M_{k_t}) M_{k_{t-1}} \quad \rightarrow eq(6)$$

(e) Time Update

The state prediction update performed applying the relationship in equation (1) estimated for the time update. Equation (1) evaluating continuous, the time interval chosen to balance the processor usage and estimator performance. In several statuses of discrete or continuous inputs, the time update is executed many times between each measurement update. At each of those time updates, state estimation uses to predicted from the measurement relationship.

$$x_{k_t} = x_{k_{t-1}} + \frac{T_s}{N} F_k(x_{t-1}, u_t \quad \rightarrow eq(7))$$

The integration to estimate the state vector x_{k_t} appear multiple times to reduce errors due to integration errors from nonlinearities in the state propagation. The variable N in equation (7) is the total aggregate of redundancies T_s is the time passed after the last measurement update. Essentially,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

there is position time between processor consumption and execution of the time update.

$$x_{k_t} = \Phi_{k_{t,t-1}} x_{k_{t-1}} \quad \rightarrow eq(8)$$

$$\Phi_{k_{(t,t-1)}} = \frac{\delta F_k(x_{k_t}, u_{k_t})}{\delta x_k} \Phi_{k_{(t,t-1)}} \quad \rightarrow eq(9)$$

where $\Phi_{k_{(t-1,\ t-1)}} = I \quad \rightarrow eq(10)$

Φ_k state transition matrix in state vectorx_k maps $t-1$ the value of one state to the next state t in the time update.

$$M_{k_t} = \Phi_{k_{(t,t-1)}} M_{k_{t-1}} \Phi^T_{k_{(t,t-1)}} + Q_k \quad \rightarrow eq(11)$$

Q_k is the matrix that adjusts parameters that express the predicted covariance of the state determined by equation (1)

$$M_{k_t} = M_{k_{(t-1)}} + \frac{T_s}{N}\left(\left[\frac{\delta F_k(x_{k_t}, u_{k_t})}{\delta x_k}\right] M_{k_{t-1}} \left[\frac{\delta F_k(x_{k_t}, u_{k_t}}{\delta x_k}\right]^T\right) + Q_k \quad \rightarrow eq(12)$$

Equations (11) and (12) both are a form of updating covariance matrix. Updating the covariance matrix can be iterated to improve the estimation.

(f) Second Order Extended Kalman Filter (EKF2)

The measurement of inertial velocities to 'smooth' the GPS measurements to resolve high-frequency estimates for the states.

$$x_{k2} = [p_n p_e v_g\ x\ w_n w_e\ \psi]^T \rightarrow eq(13)$$

$p_n p_e$ is the Inertial Earth position towards the North and East origin at an initial position. v_gThe velocity of the airframe to ground, x is zero wind sideslip if yaw angle is in same course angle. $w_n w_w$ is a component of wind velocity V_w in the direction from the north and east. ψ is Yaw angle in degrees, First Euler angle.

$$u_{k2} = \left[\hat{\phi}_k \hat{\theta}_k p_{LPF} q_{LPF} r_{LPF} v_{LPF}\right]^T \quad \rightarrow eq(14)$$

EKF2 use only the GPS measurement, shown in equation (15).

$$y_{k2} = \begin{bmatrix} y_{p_{n,GPS}} \\ y_{p_{e,GPS}} \\ y_{v_{g,GPS}} \\ y_{x_{gps}} \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow eq(15)$$

Two Zero placed here corresponds to two 'pseudo measurements' that this classification uses to represent the measured wind in the north and east directions. The flight path angle of the UAVs is zero that the wind has no vertical component pseudo-measurements are associated parts of the EKF(k) states x and truth inputs u as shown.

$$y_{5_{k2}} = 0 = V_a \cos\psi + w_n - V_g \cos x \rightarrow eq(16)$$

$$y_{6_{k2}} = 0 = V_a \sin\psi + w_e - V_g \sin x \quad \rightarrow eq(17)$$

The measurement to state relationship H_{k2} in equation (2) is written here as equation (18).

$$H_{k2}(x_{k2}, u_{k2}) = \begin{bmatrix} p_n \\ p_e \\ v_g \\ x \\ v_a \cos\psi + w_n - v_g \cos x \\ v_a \sin\psi + w_n - v_g \sin x \end{bmatrix} \rightarrow eq(18)$$

Jacobin measurement update is in equation (19)

$$\frac{\delta H_{k2}(x_{k2}, u_{k2})}{\delta x_{k2}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\cos x & v_g \sin x & 1 & 0 & -v_{a,LPF} \sin\psi \\ 0 & 0 & -\sin x & -v_g \cos x & 0 & 1 & v_{a,LPF} \cos\psi \end{bmatrix} \rightarrow eq(19)$$

a. flight path angle is zero in equation (20) & (21)

$$\dot{p}_n = v_g \cos x \quad \rightarrow eq(20)$$

$$\dot{p}_e = v_g \cos x \quad \rightarrow eq(21)$$

b. Relationship that wind and airspeed are constant equation (22) & (23)

$$\dot{v}_g = \frac{(v_a \cos\psi + w_n)(-v_a \dot{\psi} \sin\psi) + v_a \sin\psi + w_e)(v_a \dot{\psi} \cos\psi)}{v_g} \rightarrow eq(22)$$

$$\psi = q \frac{\sin\dot{\phi}}{\cos\theta} + r \frac{\cos\phi}{\cos\theta}$$

c. UAVs coordinated turns

$$\dot{x} = \frac{g}{v_g} \tan\phi \cos(x - \psi) \quad \rightarrow eq(23)$$

Taking matrix to relationship with equation (1), generated equation will be shown in equation (24)

$$F_{k2}(x_{k2}, u_{k2}) = \begin{bmatrix} v_g \cos x \\ v_g \sin x \\ \frac{(\hat{v}_{a,LPF} \cos\psi + w_n)(\widehat{-v_{a,LPF}} \psi \sin\psi) + (\hat{v}_{a,LPF} \sin\psi + w_e)(\hat{v}_{a,LPF}}{v_g} \\ \frac{g}{v_g} \tan\phi \cos(x - \psi) \\ 0 \\ 0 \\ \hat{q} \frac{\sin\hat{\phi}}{\cos\hat{\theta}} + \hat{r} \frac{\cos\hat{\phi}}{\cos\hat{\theta}} \end{bmatrix} \rightarrow eq(24)$$

Some of more equation that we use while using EKF2 method.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

$$\frac{\delta v_g}{\delta \psi} = -\frac{\psi v_{a,LPF}(w_n \cos\psi + w_e \sin\psi)}{v_g} \rightarrow eq(25)$$

$$\frac{\delta \dot{x}}{\delta v_g} = -\frac{g}{v_g^2}\tan\hat{\phi}\cos(x-\psi) \rightarrow eq(26)$$

$$\frac{\delta \dot{x}}{\delta x} = -\frac{g}{v_g}\tan\hat{\phi}\sin(x-\psi) \rightarrow eq(27)$$

$$\frac{\delta \dot{x}}{\delta \psi} = \frac{g}{v_g}\tan\hat{\phi}\sin(x-\psi) \rightarrow eq(28)$$

B. System Software Architecture

1) Ardupilot & DroneKit

ArduPilot facilitates the production and adoption of advanced, autonomous UAVs in real-time for passive advantages communication between GCS, including GPS positioning, battery status and other live erudition. It provides control algorithms for vehicles with robust sensor recompense algorithms, filtering and tuning inclinations. Enabled command modes to implement in every standard vehicle: Guided, Stabilize, RTL, Land, Flip, Poshold etc. ArduPilot led to Advanced Configuration allows the composition of more high-level innovations of the firmware and hardware peripherals. It implements a comprehensive suite of tools suitable for almost every medium of vehicle and application. DroneKit-Python API enables us to build courseware that runs on an onboard companion computer and interface with the ArduPilot flight controller adopting a low-latency interconnection. The API interacts with vehicles over MAVLink protocol, API primarily intended for use in onboard companion computers (to maintain high-level performance predicaments including computer vision, path planning, 3D modelling and more). It can similarly be adopted for the GCS, interacting with vehicles over a tremendous latency RF-link.

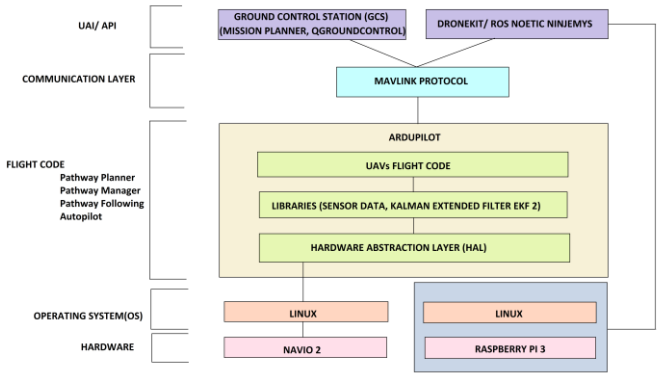


Fig 2. Integrating heterogeneity platform with Ardupilot

2) MAVLink & Mavros

The Micro Air Vehicles Link(MAVLink) is the communication protocol used to communicate between a Ground Control Station (GCS) and an Autopilot. This protocol mainly operates in ArduPilot Firmware and provides robust innovations like controlling, monitoring, integrating within the Internet. It provides systems for recognizing packet drops and allotting packet authentication. The Mavros ROS package facilitates MAVLink to accommodate flexible connection between companion computers with ROS and MAVLink enabled autopilots and GCS. The UDP broadcast used the process stage and switched to the GCS address. MAVROS supports ROS topics that can send commands, publishes telemetry and provides several services. SET_POSITION_TARGET_LOCAL_NED communication. Concedes frame target position/target speed and target yaw/angular yaw velocity, SET_ATTITUDE_TARGET communication. Concedes frame the target attitude /angular velocity and throttle level, SET_POSITION_TARGET_GLOBAL_INT. Concedes frame the target attitude in global coordinates (latitude, longitude, altitude) and flight speed.

3) ROS (Robot Operating System) & Ground Control Station (GCS)

Robot Operating System (ROS) is the framework that produces different packages, libraries and tools for us to develop and reiterate code within robotics applications. It provides conventional OS assistance like hardware abstraction, device operators, libraries, ingenuity, execution of commonly-used serviceability (communication within processes), package management etc. It includes various packages for computing trajectory, conduct SLAM algorithms or implements remote control. It can interact between Python and C++ nodes and build with a cross-collaboration concept. The project-based on ROS (1) Linux Noetic Ninjemys framework with heterogeneity Ubuntu Focal (OS) for UAVs multiple abstraction level development. The GCS software implements an autonomous operation and high-level Flight Disposition Compiler that assists the operator to compose complicated missions in a simple strategy. It predominantly works on a ground-based computer that is applied toward planning and operating a mission. UAV ground control software will provide a real-time image of the vehicle's state and providing the capability to improve our mission. The GCS mission planners are building around a 2D or 3D mapping, which in enrichment to altitude and topography may render separate erudition such as no-fly zones and temporary restraints. A telemetry player feature is possible that enables administrators to replay the mission for moreover insights and interpretation.

C. Integrated Software and Simulation

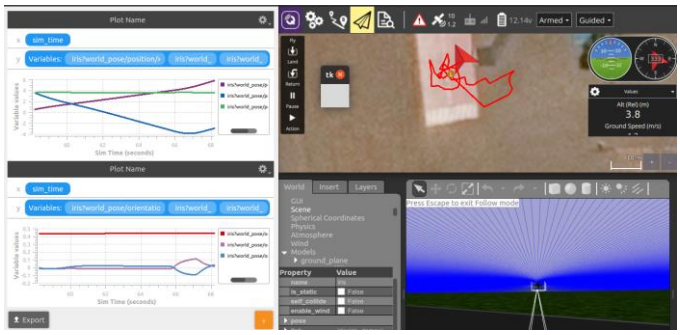
1) SITL (Software in The Loop) & ROS-Gazebo

SITL enables us to operate ArduPilot on our computers directly, externally any exceptional hardware. ArduPilot proceeding SITL gives us access to the extensive extent of community tools available for desktop C++ development, typically interactive debuggers, potential analyzers and dynamic analysis tools. Here we are operating with a heterogeneous interface of ArduPilot and Dronekit-Python. The simulator model we are using is Gazebo, a 3D dynamic simulator that has the extended capability to precisely and efficiently simulate communities of robots in complex indoor and outdoor conditions. To obtain ROS integration

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

among stand-alone Gazebo, an assortment of ROS packages specified gazebo_ros_pkgs implements wrappers approaching the stand-alone Gazebo. Gazebo plugins provide URDF models with more comprehensive functionality and join in ROS communications and assistance requests for sensor output and motor input. Building with Catkin workspace combines CMake macros and Python scripts to implement some functionality on top of CMake's workflow, decreases code redundancy with Gazebo. The demonstration of heterogeneity SITL, Simulator and GCS are shown in Figures (3) a & b.

Fig 3. a) ROS-Gazebo Simulation with Mission Planner



b) ROS-Gazebo Simulation with QGroundControl

2) The Obstacle Avoidance Repulsive Potential Field Methods

The repulsive potential sustains UAVs away from the obstacles, both those a priori apprehend or those recognized by the UAV onboard sensors. The Repulsive Potential is higher during the UAV is closer to interference and produces a decreasing magnetism when the UAV is far distant. The Linear nature from the repulsive potential sustains UAVs away from the intricacy, the repulsive potential proceeds from the sum of the Repulsive influence of all specific obstacles is

$$U_{rep}(X) = \sum_{i} U_{rep_i}(X) \rightarrow eq(29)$$

An obstacle quite far from the UAV is not possible to deflect. Furthermore, the magnitude of repulsive potential should expand when the UAV approaches a nearer obstacle. To estimate for this consequence and to the space surrounded influence, a feasible repulsion potential generated by an interference i is

$$U_{rep_i}(X) = \begin{cases} \frac{1}{2} k_{obst_i} \left(\frac{1}{d_{obst_i}(X, X_0)} - \frac{1}{d_0} \right)^2, & \text{if } d_{obst_i}(X, X_0) \le d_0, \\ 0, & \text{if } d_{obst_i}(X, X_0) > d_0, \end{cases} \rightarrow eq(30)$$

Where $d_{obst_i}(X, X_0)$ is the insignificant distance from (X) to obstacle i, K_{obst_i} is the repulsive potential field constant. d_0 is the influence range of the repulsive potential field.

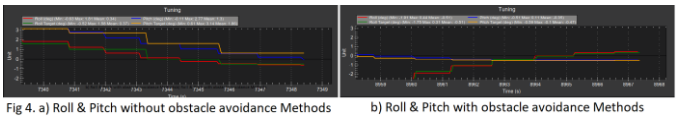


Fig 4. a) Roll & Pitch without obstacle avoidance Methods b) Roll & Pitch with obstacle avoidance Methods

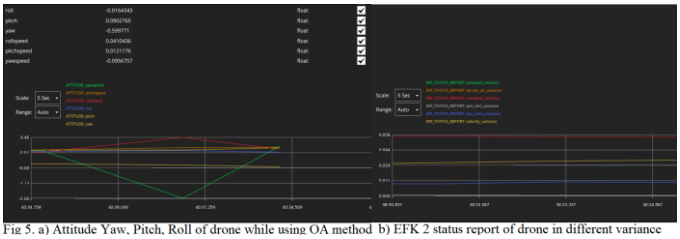


Fig 5. a) Attitude Yaw, Pitch, Roll of drone while using OA method b) EFK 2 status report of drone in different variance

3) LIDAR

Obstacle avoidance obtains by appealing LIDAR data to implement a two-dimensional Cartesian map. The proposition uses to create the map of the conditions preceding the pathway intended based on the map to enable the drone or robot to navigate in the petitioned obstacle-free area.

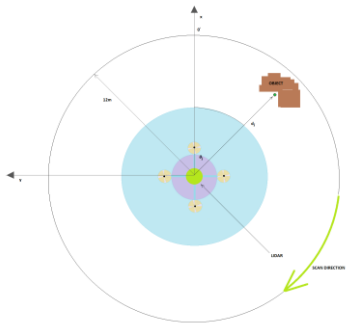


Fig 4: Scanning method of RPLIDAR Sensor

2. CONCLUSION

The Lidar adopted measures 360 distance points all of the raw laser points embodied in the polar coordinate system as $\{(d_i, \theta_i); 0 \le i \le 359\}$, anywhere d_i is the distance measured from the center of the observer UAV to the object and θ_i the relative angle of the measurement. The acquired Lidar erudition gathered in vector (d_i, θ_i), the stored information converting the infinity scan values that means that there is no obstacle corresponding the ray to the maximum extent power that could measure by the Lidar (d_{max}). An object located (d_{max}) the observer UAV will be ignored. In real-time, Lidar insistence instantly transmits the max range value to object outside their operating range. Furthermore, it is also conceivable to utilize the standards filtering to eliminate noise from the Lidar data.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

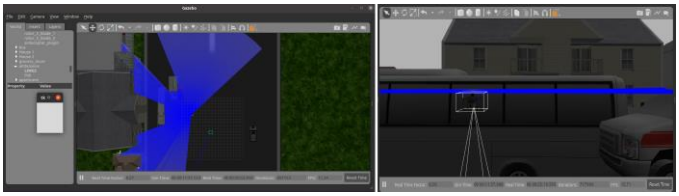


Fig 5. Lidar obstacle avoidance view in ROS-Gazebo Simulator

This paper presents a Platform including (Raspberry pi & Navio 2) procedure to experiment with the genuine simulated conditions for UAVs autonomous navigation to establish approaching data enactment and classification of the algorithm satisfied to hold the benefit of convenient reference. Therefore, the work is extended to improve the autopilot system using a distinct heterogeneity combination of Software and Hardware. Simulation is implementing using the ROS Noetic Gazebo (3D visualization and physic modelling) plugins with various sensors combinations of URDF files. Working with AHRS EKF 2 derivation, combine DroneKit-python and Ardupilot SITL for UAV attitude and orientation. The obstacle Avoidance or an Autonomous flight is obtain using LiDAR, where the algorithm is build using the Repulsive Potential Field Method. Mission Planner & QGroundControl GCS has been applied here for achieving real-time navigation and trajectory planning in this phase. We have additionally implemented python-TK for commanding UAVs using keyboard direction arrows.

3. REFERENCES

[1] Arning, Richard & Langmeier, Andreas & Stenzel, Erwin. (2007). UAV/UCAV NAVIGATION SYSTEMS - PRESENT AND POTENTIAL FUTURE.

[2] Ribeiro, Maria Isabel. "Obstacle avoidance." Instituto de Sistemas e Robótica, Instituto Superio Técnico (2005): 1.

[3] García, Jesús, and Jose M. Molina. "Simulation in real conditions of navigation and obstacle avoidance with PX4/Gazebo platform." Personal and Ubiquitous Computing (2020): 1-21.

[4] Meyer, Johannes & Sendobry, Alexander & Kohlbrecher, Stefan & Klingauf, Uwe & Von Stryk, Oskar. (2012). Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo. 7628. 400-411. 10.1007/978-3-642-34327-8_36.

[5] Yang, Shishan & Baum, Marcus. (2016). Second-Order Extended Kalman Filter for Extended Object and Group Tracking.

[6] Kallapur, Abhijit & Salman, Shaaban & Anavatti, S.G. (2007). Application of Extended Kalman Filter Towards UAV Identification. 10.1007/978-3-540-73424-6_23.

[7] Valiev, Mukhammad, and Husan Kosimov. "International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878." Locomotive Diesel Engine Excess Air Ratio Control Device 8.

[8] Ravankar, Ankit A., Abhijeet Ravankar, Yukinori Kobayashi, and Takanori Emaru. "Autonomous mapping and exploration with unmanned aerial vehicles using low cost sensors." In Multidisciplinary Digital Publishing Institute Proceedings, vol. 4, no. 1, p. 44. 2018.

[9] Khan, Zubair Ahmed. "Obstacle Avoidance Methods in UAVs." PhD diss., 2019.

[10] Chen, Shengyang, Han Chen, Weifeng Zhou, C-Y. Wen, and Boyang Li. "End-to-end UAV simulation for visual SLAM and navigation." arXiv preprint arXiv:2012.00298 (2020).

[11] Alborzi, Y., B. Safari Jalal, and E. Najafi. "ROS-based SLAM and Navigation for a Gazebo-Simulated Autonomous Quadrotor." In 2020 21st International Conference on Research and Education in Mechatronics (REM), pp. 1-5. IEEE, 2020.

[12] Park, Jongho, and Namhoon Cho. "Collision avoidance of hexacopter UAV based on LiDAR data in dynamic environment." Remote Sensing 12, no. 6 (2020): 975.