International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 11 Issue 8 ǁ August 2023 ǁ PP. 105-121

Real-Time Control of Drones with Focus on NN-basedAttitude Control

Vemema Kangunde · Rodrigo Jamisola Jr. · Lucky Mohutsiwa
Botswana International University of Science and Technology Palapye, Botswana

Abstract *This paper develops a neural network (NN) attitude controller for real-time unmanned aerial ve- hicle (UAV) control implementation. UAV path plan- ning is achieved by position trajectory-tracking which is necessitated by the action of the lower-level attitude controller. Thus accurate attitude trajectory-tracking is indispensable for UAV path planning. Sensory data from UAV onboard sensors is continuously processed by the onboard UAV flight controller in order to up- date UAV position and orientation. IMU sensors may provide attitude information, while the GPS provides the UAV position. An onboard control algorithm utilise these sensory information for UAV guidance and con- trol. In this paper, focus is made on the development of the lower level attitude controller for UAV applications. The control of UAV attitude is key to all successful UAV applications. A PID and a neural network controller are developed for attitude control. The PID controller was used for bench marking as well as to facilitate the de- velopment of the neural network controller. The per- formances of the respective controllers are investigated on a UAV test platform. The main challenge for the NN controller development is training data collection and preparation for effective attitude control. Further- more the low computational power in most open sourcemicro-controller units (MCUs) poses a major limita- tion.*

Keywords *Multi-rotor drones · Real-time control · Neural networks · Inertial measurement unit · Kalmanfilter · Low-pass filter · Unmanned aerial vehicle · Feedback control*

Date of Submission: 04-08-2023 Date of acceptance: 17-08-2023

I. Introduction

Unmanned aerial vehicles (UAVs) are found in a wide range of applications. These include survey and map- ping, transportation, environmental monitoring, disas- ter relief, industrial monitoring, traffic monitoring, bor- der surveillance as well as military services such as air defence early warning, battlefield surveillance, target lo- calisation and tracking, and military strikes [1–3]. The rapid adoption of UAV systems has bee due to advance- ment in computing, sensing, and control [4, 4–6]. UAVs serves as a platform on which many, such as the afore- mentioned, applications can be implemented. It acts as a carrier vehicle for applications payloads. Various payload sensors are attached onboard UAVs for vari- ous application, object detection is discussed in [7, 8], onboard cameras are used for vision. Various cameras such as multispectral and hyperspectral cameras can be attached to UAV systems for precision agriculture. High quality images could be captured in order to ex- tract vegetation indices to enable farmers to monitor crop variability, stress conditions, as well as diseases and pests [9]. UAV applications in transportation is dis- cussed in [10], transportation of medical supplies, blood specimens, vaccines, as well as blood transfusions using UAVs was discussed. In all UAV applications the suc- cess of the mission relies on successful UAV control. Furthermore the attachment of payloads increase UAV dynamics leading to more demands on the UAV control.The control of UAV includes position and attitude con-

trol. Attitude control is the lower level control and the attitude controller is manipulated to enable UAV position control. Various control strategies are discussed in the literature, however most UAV control implementations are dominated by PID controllers. Neural network controllers promises better performance than classical PID controllers but they have not been widely adopted. Most previous studies apply neural networks in obstacle avoidance during UAV navigation [11–15], as well as in compemsator models for external disturbance and model errors [13,16]. Furthermore most of the works regarding NN-based UAV control are of simulation while others run models on host machines and communicate with the UAV system via links such as wireless LAN [14] without actually porting the model to an onboard embedded MCU. This is due to shortage of computational power in most open source MCUs, as well as implementation complexity and other technical challenges. Therefore, there is limited studies on the employment of NN models for attitude control.

This paper develops an NN attitude controller for real-time UAV control implementation. A test platform was built for testing the controller. A PID controller was first developed and implemented on the test platform. The NN controller was then developed based on data from the PID controller and controller performances were compared. The NN controller model was trained using ANNHub platform. This is a machine learning platform that optimises NN models for resource constrained MCUs, enabling models to be embedded on low computational power MCUs. The platform can export trained models to real-life applications.

2 UAV Control Strategies

UAV altitude, as well as velocity-and-heading control needs a robust and accurate controller [17]. The altitude controller is to achieve UAV desires altitude during flight, take off, and landing stages of the UAV. The velocity-and-heading controller on the other hand ensures UAV follows desired waypoints [18]. Various control strategies are discussed in the literature to achieve UAV control objectives, these includes Fuzzy Logic, Linear Quadratic Regulator (LQG), Sliding Mode Control (SMC), Proportional Integral Derivative (PID), Neural Networks (NN), e.t.c. Robust control have been considered to address parametric uncertainties as well as external disturbances. In multirotor UAVs, disturbance noise caused by propeller rotation, blade flapping, and variations in propeller rotational speeds needs a robust nonlinear controller [19]. In [19] robustness as well as Compensation for system nonlinearities was addressed by combining nonlinear sliding mode control (SMC), robust backstepping controller and a nonlinear disturbance observer (NDO). The backstepping controller stabilised translational movement while the SMC controlled the rotational movement of the quadrotor. The NDO provided all the estimates of disturbances ensuring robustness of the feedback controls. Sliding mode control (SMC), robust backstepping controller and a nonlinear disturbance observer (NDO) were cooperatively used to address system nonlinearities [19]. The backstepping controller was employed for translational movement stabilisation and the SMC controller was used for control of rotational movement. System disturbances were estimated by the NDO to ensure robustness in feedback estimates.

A PID controller and a direct inverse control neural network (DIC-ANN) were compered via simulation in [20]. Both controllers were excited with the same altitude reference input and their responses were plotted. Quadrotor flight was simulated in four stages comprising take-off and climb phase at $0 < t < 10\ s$, hovering phase at $10 < t < 20\ s$, climb in ramp phase at $20 < t < 22.5\ s$, and lastly the final altitude phase at $22.5 < t < 50\ s$. The DIC-ANN outperformed the PID controller in handling quadrotor altitude dynamics. At hovering conditions the DIC-ANN displayed less steady state error as compared to the PID controller. The transient oscillations damped faster with the DIC-ANN, this means it handles nonlinearities better than the PID controller. Most autopilot systems employs PID controllers due to their ease of implementation. PID controllers, however, have limitations when subjected to unpredictable and harsh environments. The performance and accuracy of a neural network controller was investigated in [21]. The neural network (NN) based controller is trained through reinforcement learning (RL) state of the art algorithms, the Deep Deterministic Policy Gradient (DDPG), Trust Region Pocy Optimisation (TRPO), and the Proximal Policy Optimisation (PPO). The NN controller was compared with a PID controller to establish the suitability of NN controller in high precision, time-critical flight control. Controller performances ware evaluated in simulation, using GYMFC environment. The results revealed that RL trained NN controller can trail accurate attitude controllers, also the controller trained with PPO outperformed a fully tuned PID controller on almost every criteria.

The linear quadratic regulator (LQR) optimal control algorithm controls a dynamic system by minimizing an appropriate cost function [22]. When the LQR is used in combination with a linear quadratic estimator (LQE), and Kalman filter, it is called the linear quadratic Gaussian (LQG). The LQG was employed in

[23] for quadrotor micro aerial vehicles (MAVs) altitude control. The linearised model for altitude control problem was obtained as (1), disregarding air resistance. The state space model is given by (2) , while the cost function is given by (3), referred to as the quadratic form criterion in [23]. The objective is to determine the control input $U(t)$ to minimise the cost function. [22].

$$\ddot{Z} = a = \frac{F - mg}{m} \tag{1}$$

$$\begin{aligned} \dot{x}_a(t) &= \frac{d}{dt}\begin{bmatrix} x(t) \\ x_r(t) \end{bmatrix} \\ &= \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}\begin{bmatrix} x(t) \\ x_r(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(t) \\ &= A^* x_a(t) + B^* u(t) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(t), \end{aligned} \tag{2}$$

where

$$A^* = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}, \quad B^* = \begin{bmatrix} B \\ 0 \end{bmatrix}.$$

$$J = \frac{1}{2}\int_0^\infty [x_a^T Q x_a(t) + u^T(t) R u(t)]\, dt. \tag{3}$$

The linear Quadratic Regulator (LQR), Linear Quadratic Gaussian (LQG), and a non-linear controller were considered in [24] to address external disturbances for fixed wing UAV control. The control techniques were applied to a linearised model of the UAV. Controller performances were assessed on simulation using Matlb/Simulink toolbox. The nonlinear controller showed faster response, good robustness and stability as compared to the LQR and LQG. The LQG preformed better than the LQR controller in the presence of disturbances. In some control implementations, sliding-mode control (SMC) strategy is considered. Sliding-mode control is a nonlinear control strategy that applies a discontinuous control signal to the system ensuring that it slides along a prescribed path or sliding surface [22]. An SMC based fault tolerant control design for underactuated UAVs was implemented on a quadrotor in [25]. system dynamics were separated into two sub-systems, a fully actuated and an under-actuated subsystem. A Nonsingular Fast Terminal Sliding Mode Controller (NFTSMC) was designed for the fully actuated subsystem, the Under-actuated Sliding Mode Controller (USSMC) was then derived for the under-actuated subsystem. The controller performance was evaluated on a quadrotor platform. The controller demonstrated excellent robustness to actuator faults and external disturbances. It had fast convergence and high tracking precision. Herrera et al. designed a sliding-mode controller and applied by simulation to a quadrotor. They considered a PD (proportional Derivative) sliding surface for vertical take-off and landing. A wide scope on control algorithms for quadrotors can be found in [22].

3 Feedback Control

State estimates, such as estimates for UAV translation, rotational movements, and position, are required for unmanned aerial vehicle (UAV) feedback control [26–28]. The flight controller generates UAV state estimates using onboard sensor measurements. The need for state estimation is due to uncertainties in sensor measurements arising from factors such as atmospheric disturbances, vibrations noise, inaccuracy of coordinate transformations, and missing measurements [26,29]. Furthermore signal obstructions for sensors such as global positioning system (GPS) affects the reliability of the sensor. Despite all setbacks timely response of UAV and robotic systems is critical [30–32]. Multiple sensors can be used together for augmentation purposes [28]. This compensates missing signals from affected sensors. The gyroscope, accelerometer, magnetometer, GPS, and pressure sensors, for example, may be combined for UAV localisation. A high frequency update on UAV states is required for effective UAV control. Low update frequency of some Sensors, such as the GPS, with a typical update frequency of 4Hz, can be negated for using Kalman filtering techniques. Kalman filtering techniques can also be employed for gyroscope drift correction. Gyroscope drift errors can also be addressed using model compensation where the gyroscope random drift is modelled, and the model is used to predict the drift. The gyroscope reading is then offset by the predicted drift [33, 33, 34].

The accelerometer and gyroscope are widely used in stabilisation platforms [35, 36]. Accelerometer measurements suffer from high-frequency noise while gyroscope readings are prone to drift error due to integration [35, 37]. In the literature, several compensation approaches for gyroscope drift are discussed. In [38] gyroscope drift was modeled in order to predict the drift error of a micro electro-mechanical systems (MEMS) gyroscope. The wavelet threshold denoising algorithm was used to separate drift error and white noise. Improved Elman neural network was used to promote gyroscope drift model accuracy [38]. The gyroscope reading was offset by the drift error predicted by the gyroscope drift

model. Adaptive H∞ Kalman filter was used for random gyroscope drift modelling in [39]. In [40], empirical mode decomposition (EMD) algorithm was used to decompose the gyroscope drift signal, threshold denoising and a clustering algorithm was then used to extract the non drifting gyroscope signal, and ensemble-ELM (extreme learning machine) predicted and compensated for the gyroscope drift error. Some implementations uses the zero velocity compensation (ZVC) method to overcome gyroscope drift errors [41–43]. The ZVC employs the gyroscope data static and moving mean mean. If the moving mean is less than zero, the velocity is added with the static mean and if the moving mean is higher than zero, the static mean is subtracted from the velocity. The selection of drift error compensation techniques is application dependent.

4 Real-time Control

In order to implement real-time control for UAVs, tasks have to be defined. An RTOS is required for tasks scheduling, inter-task communication, and management of available resources such memory, power consumption, etc. Each task is allocated a memory space, called a stack, in the microprocessor. This is enabled by the RTOS kernel's support for multi-threading. Real time implementation in UAV control is needed, especially, for management of multiple UAV tasks, to ensure timely reagement of multiple UAV tasks, to ensure timely response and context switching between the tasks. A real-time operating system (RTOS) is used for task scheduling, inter-task communication. It also manages available computing resources, power consumption, etc. Task scheduling and prioritisation, as well as sensor update frequency ensure timeliness of application tasks. In [44], an RTOS (RT-Thread) is used to address such needs as real-time response, heavy workload and difficulty in control of a quadrotor. Results showed that quadcopter control system implemented on RT-Thread displayed good response time and smooth flight with a PID controller. Tasks such as attitude information acquisition, attitude information fusion, and PID control were considered in this work. Application tasks were implemented on RT-Thread RTOS running on STM32F407VGT6 microprocessor. The processor has a high-performance ARM Cortex-M4 core with maximum system frequency of 168MHz, an FPU (floating-point unit), 1 Mbyte of flash, and 192 Kbytes of SRAM. It has peripherals such as ADC, SPI, USART, controller area network (CAN) bus, DMA, etc. High operating frequencies and high-speed memory provide high computational power to enable quadcopter complex calculations to be performed. Also additional peripherals reduce the need for external IC and reduce computational burden from the microprocessor. The implementation in [45] uses a dual processor configuration. One processor is used for telemetry while the other is used for quadcopter control. The telemetry processor performs software tasks such as communicating reconfiguration and monitoring data with the ground control station (GCS), sensor data collection, and wireless transmission of data to the GCS. The tasks are managed by μC/OS-II™, an RTOS. The control processor runs the PID controller algorithm for the quadcopter stabilization and navigation. This task was achieved through several tasks allocated to the control processor. Tasks include reading GPS, compass, IMU, and altitude sensor data received from the telemetry processor. Other tasks include implementation of the roll, pitch, yaw, and altitude PID control loops, and communicating reconfiguration and monitoring data with the telemetry processor via CAN bus. The literature on real-time implementation of drone control systems is relatively limited, and the number of reported studies on UAV scheduling has been minimal [26]. The leading characteristic of a real-time implementation is the employment of an RTOS, referred to as UAV operation system in some literature. An RTOS is mostly needed with increased number of UAV tasks and missions. It provides a real-time kernel for implementation of UAV tasks. The timing requirements of each task is achieved by the use of real-time scheduling algorithms [46, 47]. FreeRTOS

of real-time scheduling algorithms [46, 47]. FreeRTOS

is the widely used RTOS for UAV applications [47]. It has features such as multi-threading, task scheduling and prioritisation, inter-task communication etc.

5 Neural Networks

Artificial neural networks (ANN), or simply neural networks (NN), learns from past data and when given new data responds by making predictions or classifications based on past experience of input-output characteristics [48–50]. The advantage of ANN is the ability to transform system inputs to outputs without the knowledge of their corresponding physical relation [50–57]. NN models are used in a variety of applications from engineering, medicine, and finance, to name a few [58–61].

In terms of UAV applications, some examples can be highlighted; Jiang et al. [16] proposes sigma-pi neural networks (SPNN) augmented dynamic inversion controller for UAV flight control performance improvement.The UAV dynamics inversion model was used to linearise the UAV nonlinear system while a neural network was incorporated to eliminate errors caused by the dynamics model. Experimental and simulation results demonstrated that the NN-based control system outperformed the PID -based control system in terms of trajectory

tracking accuracy. Padhy et al . [14] facilitates a UAV with a monocular camera for navigation in previously unknown and GPS denied indoor environments. A video from UAV front camera is fed into a deep neural network model which determines the next course of maneuver. Authors used the Parrot AR Drone as their test platform. The NN model was run on a host machine which interfaced with the UAV via wireless LAN. The UAV was expected to take off and navigate along an indoor corridor before landing safely at the corridor end. Authors determined the performance of the UAV based on some ratios; the no collision ratio (NCR) and the full flight ratio (FFR). The NCR considers the ratio of the number of times the quadcopter successfully navigates the whole length of the corridor, without any collision with the walls, to the total number of trials. The FFR is the same as the NCR except that it tolerates less impactful sideways collision with the walls. The proposed model generated necessary control command to safely navigate the UAV in unknown corridor environments.

5.1 Neural Networks in Embedded Systems

The significant benefits of machine learning (ML) methods has been hindered on MCUs applications due to large memory demand and expansive computational throughput of ML inference models [62–64]. There has been mounting interest for NN inference applications on embedded MCUs. This offers the advantage that data can be processed locally, which averts challenges posed by data transmission latency, when transferring to ground processing unit (GPU), or to the cloud, such as in mobile applications [65,65,66]. Also data captured by sensors may be lost due to transmission costs, bandwidth limitations, as well as power constraints [67–70]. To enable ML on embedded MCUs, neural architecture search (NAS) algorithms have been developed and ML algorithms are in recent years being implemented on embedded MCUs . The paradigm of embedding ML algorithms on MCUs is referred to as TinyML. It is an active research field aiming to see efficient performance of ML models on embedded computers. NAS algorithms optimise neural network inference for deployment on resource constrained hardware, such as MCUs [71–73]. Various NAS algorithms are discussed in the literature. NAS targeted for embedded applications need not only optimisation of model performance but also have consideration of the resource constrained nature of embedded MCUs. Cassimon et.al [73] make improvements on a recent NAS, the Efficient Neural Architecture Search (ENAS), to take into account embedded resource constraints. Extra constraints such as MCU flash memory, cache memory, and NN model compression estimation were considered when designing the ENAS algorithm's reward function to raise the NN inference model's awareness of the resource constraints of the targeted embedded MCU. The improved algorithm was used to train an NN model to predict English words at they would be spoken. The model was ported to a Raspberry Pi 3B MCU having 1GB RAM, half of which was used by the model. The need to expand the focus of TinyML research from ML model size reduction and optimisation of existing models, which in most cases are static, is addressed in [65]. Authors highlight the fact that many of the data monitored by embedded Internet of Things (IoT) applications is constantly changing and in some cases evolves to an unprecedented format due to unforeseen disturbances. The constant evolution of the data render offline static ML models ineffective. Authors present a novel evolving algorithm, the Tiny Anomaly Compressor (TAC) for local and online data compression on embedded IoT devices.

5.2 Neural Networks Training

During training, ANN model learns the input-output relationship. NN learning can be categorised into supervised, unsupervised, and reinforcement learning. In supervised learning the NN model produces an inferred function to map inputs to outputs based on example input-output pairs. Unsupervised learning on the other hand only have input data with no corresponding output, and an appropriate learning algorithm is used to explore hidden data patterns to find similarities and differences for purposes of data clustering. In reinforcement learning a machine learning model or agent is trained to take an action, and it is given rewards or penalties for the action taken as a means to optimise its performance. Neural networks structure conform to the multilayer perceptron (MLP), widely used in most applications[74–76]. The MLP is a supervised learning paradigm that use feed forward and back propagation. Data flows forward from the input layer to the output layer which determines the output prediction. The structure of an MLP NN is shown in Fig. 1, where

x_1,x_2,x_3, and x_4 are generalised inputs and there is one output, y. There could be more than one output as well as more than one hidden layers. Each neuron in the NN layers has an activation function which determines the output level of the neuron, for example, the sigmoid function has an output range of $0-1$. Each neuron is connected to neurons in the next layer and has an associated weight, w. NN training happens in feed forward and back propagation where inputs: x_1,x_2,x_3, and x_4, fed into the input layer, are processed by the

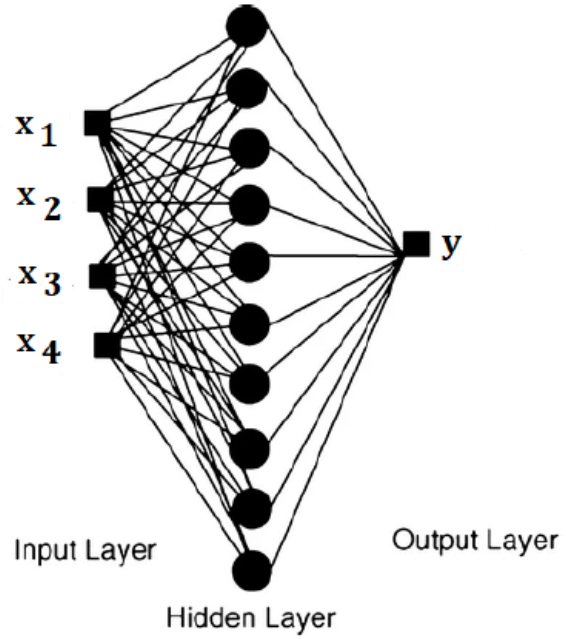

Fig. 1: Neural network

activation function and scaled by their corresponding weights as they pass through the input layer, the hidden layers, and the output layer. The output prediction of the neural network is determined by the output layer. Prediction values are compared to known outputs provided in the training data. The error between the prediction of the NN model, $\overline{y}$ and the known output y is used during back propagation to adjust the weights of the NN neurons from the output layer to the input layer to increase the precision of the NN model, see Fig. 2 .

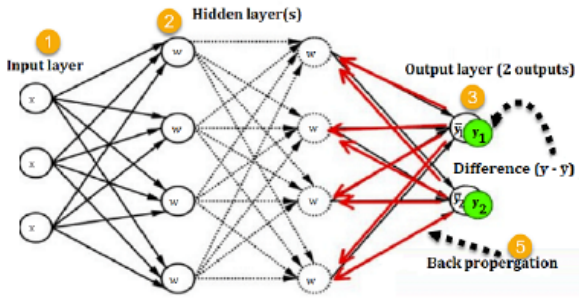

Fig. 2: NN training

6 Experiment Setting

Accelerometer and gyroscope sensors are used to provide feedback information on UAV attitude. However, the high sensitivity noise and gyroscope drift errors requires that sensor measurements be processed before they can be used for UAV control. The GY-85 MEMS inertial measurement unit (IMU) was used int this paper for feedback update. An arduino-UNO was used as the flight controller MCU. A single axis platform with two counter-rotating propellers was built to test the PID and NN attitude controllers. The single axis can represent either the roll-axis or pitch-axis of a symmetric quadrotor. In this paper the control of the roll axis was considered. Effects of motor vibrations on sensory data were considered. Accelerometer and gyroscope data was processed using a low-pass filter, and a complementary Kalman filter. On board the test platform is an arduino board, the GY-85 MEMS IMU, and a radio receiver. A PID controller was implemented to stabilise the test platform. Figures 4 and 5 depicts the platform used to tune the gains of the PID controller. PID controller inputs and outputs data was then used in the development of a neural network controller. Figure 3 depicts a free body diagram of the test platform [77]. A pitch about the y-axes results in a horizontal force (resultant thrust vector) to the left (right if tilted in the opposite direction). A roll about the x-axis results in a horizontal force (forward into the paper and backward out of the paper depending on the direction of the tilt). When $\theta = 0$ and the drone is level, a simultaneous increase in thrust to all motors such that the total thrust vector $T > mg$ will result in a vertical upward force. The hardware used is a custom built frame of the test platform, brushless DC motors, power distribution board, and electronic speed controllers. The hardware was assembled in the laboratory and interfaced with an arduino board for flight control. The software implementation was done in c++ using arduino integrated development environment (IDE). It took advantage of the basic linear algebra arduino library for the implementation of the Kalman filter algorithm. The library enables more compact matrix initialisation, as well as direct matrix computation without need for element-by-element operation as in c/c++ based programming environments. Figure 6 shows a block diagram of the UAV platform control system with unity feedback closed loop. The PID controller used in this work ,(denoted $C(s)$), can be expressed as in (4). The PID controller transfer function (TF) can be found by taking the Laplace transform of (4) giving (5). Figure 7 depicts a diagram of the planar model of the UAV platform, this has the corresponding equation (6).

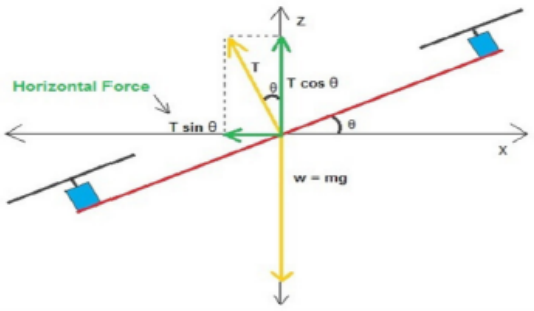

Fig. 3: Force components acting on a quadrotor, picture reprinted from [77].

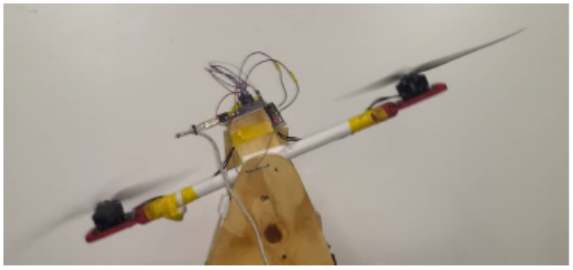

Fig. 4: Unstable system, no steady state achieved, video can be viewed at https://www.youtube.com/watch?v=suwssJHjQNE.

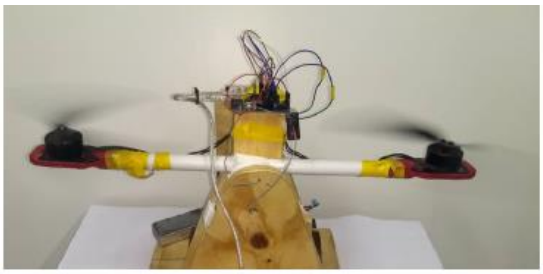

Fig. 5: Stable system, coming to steady state, video can be viewed at https://www.youtube.com/watch?v=IdfV9ztwVuo.

This model makes the assumptions that there is no translational movement in the e_x direction, and furthermore the UAV platform translation in the e_y and e_z directions is zero since the center of mass of the test platform is fixed at center pivot. The dynamics of the platform therefore simplifies only to the roll rate ($\ddot{\phi}$) about the $x-axis$. The roll rate is given by (7). The system transfer function can be calculated by taking the Laplace transform of (7), and is given by (8). From fig 6, and letting $H(s) = C(s)P(s)$, the closed loop transfer function is calculated as (9). Substituting (5) and (8) into (9) yields (10). The characteristic equation of this closed loop transfer function is given in (11). Table (1) is the Routh table for stability analysis of the UAV platform system. According to the Routh method, the stability of the system is guaranteed if all the elements of the first column in the Routh array are positive and there is no sign change in the first column. Clearly all the elements of the first column are positive and there is no sign change, however, this solely depends on the choice of parameters constituting the third element of the Routh array. The parameters should be chosen such that $K_dK_p - K_iI_{xx} > 0$. The terms K_p, K_i, and K_d can all be adjusted during PID tuning. The value of I_{xx} is positive even though not readily available. See Table (1) footnote for the meaning of parameters used in the equations. The system stability can therefore be guaranteed and is dependent upon PID parameter tuning, this is covered in Section 8.

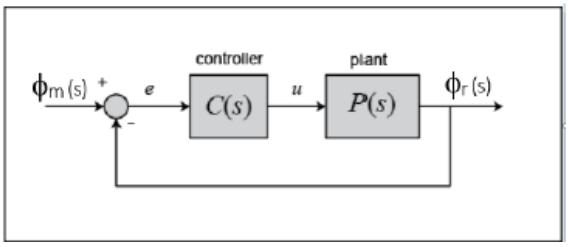

Fig. 6: Block diagram of the PID controlled UAV system

$$u(t) = K_p e(t) + K_i(t)dt + K_d\frac{de}{dt} \tag{4}$$

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \tag{5}$$

$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{-1}{m}sin(\phi) & 0 \\ \frac{1}{m}cos(\phi) & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{6}$$

$$\ddot{\phi} = \frac{u_2}{I_{xx}} \tag{7}$$

$$P(s) = \frac{\Phi(s)}{U(s)} = \frac{1}{I_{xx}s^2} \tag{8}$$

$$CTF = H(s) + \frac{1}{1 + H(s)} \tag{9}$$

Table 1: Routh table for the UAV test platform system [1]

s^3	1	$\frac{K_p}{I_{xx}}$
s^2	$\frac{K_d}{I_{xx}}$	$\frac{K_i}{I_{xx}}$
s^1	$\frac{K_d K_p - K_i I_{xx}}{I_{xx} K_d}$	
s^0	$\frac{K_i}{I_{xx}}$	

[1] K_p –proportional gain; K_d- derivative gain; K_i – Integral gain; I_{xx} – moment of Inertia for UAV platform;

$$CTF = \frac{K_d s^2 + K_p s + K_i}{I_{xx} s^3 + K_d s^2 + K_p s + K_i} \tag{10}$$

$$s^3 + \frac{K_d}{I_{xx}} s^2 + \frac{K_p}{I_{xx}} s + \frac{K_i}{I_{xx}} = 0 \tag{11}$$

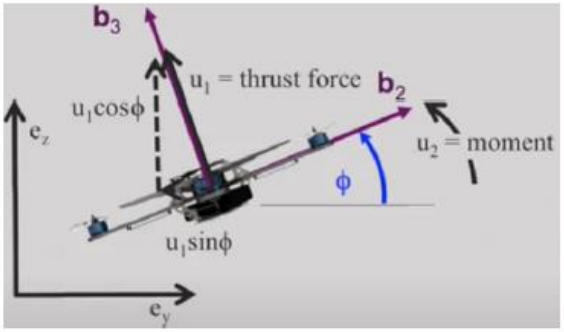

Fig. 7: Planar UAV model.

7 Sensor Data Processing

GY-85 MEMS IMU's ADXL345 accelerometer measures up to $\pm 16g$. In the case of attitude estimation, an accelerometer sensor continually measures the gravitational force of up to ±16g, at a resolution of 13-bit. The digital output data is formatted as 16-bit two's complement and was accessed via I^2C interface. When the GY-85 MEMS IMU is placed spirit level, the gravitational force is only in the positive z-axis (downwards). When the IMU rolls and pitches, the gravitational force component is projected in the x (denoted A_x) and y (denoted A_y) axis, then the roll and pitch angles can be calculated by trigonometry. When the sensor resolution is set to full scale, the sensitivity scale factor for all ranges is 256 lsb/g according to the ADXL345 datasheet. The raw accelerometer readings are therefore divided by 256 before they are used in calculations. The GY-85 MEMS IMU is equipped with an ITG-3200 gyroscope for measuring angular rates. The sensor was used in this paper to estimate roll and pitch angles. The output data from the gyroscope is in two's complement format for each x, y, and z axis. Output data for each axis comprises two bytes, for the least significant and most significant bytes respectively. The most significant byte is shifted eight positions to the left to avoid overlapping when adding the least significant and most significant bytes to compute an output for each axis. The gyroscope was set to a full-scale range. Readings were sampled at 100Hz and the digital low-pass filter cut off frequency was set to 42Hz.

7.1 Accelerometer and Gyroscope Setbacks

Accelerometer readings are very sensitive to external noise [78–80], while the gyroscope values is susceptible to drifting due to integration when calculating for gyroscope angular position. Disturbance forces acting on a quadcopter can therefore greatly affect the integrity of the data from the accelerometer. Rotor vibrations can not be neglected as they would be fused into accelerometer readings. It is, therefore, necessary to improve the quality of accelerometer readings through filtering to remove noise. In this paper, a low-pass filter is used to process accelerometer data.

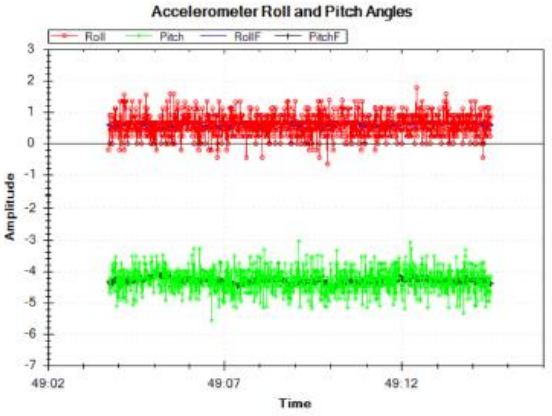

Fig. 8: Filtered and unfiltered angle estimate.

The gyroscope measures angular rate in degrees per second ($^\circ/s$). Therefore gyroscopic readings are in terms of angular velocity which when integrated can give information of the tilt angle. In this paper, the roll and pitch angles from the gyroscope are estimated through integration according to (12) and (13) respectively.

$$\phi = \int \omega_x dt \tag{12}$$

$$\phi = \int \omega_y dt \tag{13}$$

The gyroscope estimates of the tilt angles tend to drift from static conditions over time. This is indicated in Fig. 9. The roll and pitch angles drift from the starting values while the IMU has not been physically rotated. Gyroscope drift is a major setback in that over time the gyroscope angle estimates can no longer be trusted and hinders the performance of the considered application. In this paper, a Kalman filter is used to address the gyroscope drift. A process model (14) in state-space representation defines the evolution of process states from time $k-1$ to time k.

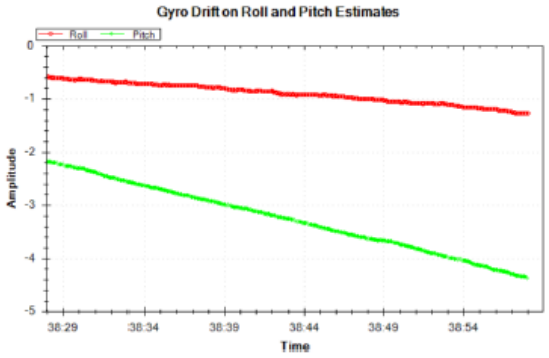

Fig. 9: gyroscope drift for roll and pitch angle estimates.

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{14}$$

A is the state transition matrix, B is the control input matrix, x_k is the state vector, u_{k-1} is the control vector, and w_{k-1} is the process noise vector. Assuming the process noise is a zero-mean Gaussian distribution, it has a related process noise covariance matrix Q. Associated with the process model is the measurement model (15) which relates the state x_k to the measurement at the current time step k.

$$z_k = Hx_k + v_k \tag{15}$$

where z_k is the measurement vector, H is the measurement matrix, and v_k is the measurement noise vector having an associated measurement noise covariance matrix R, assuming a zero-mean Gaussian distribution of the measurement noise. The process model can be written as (16) [81], where α is a generalisation for the roll and pitch states, β is a generalisation for the roll and pitch biases, T is the gyroscope sampling time, whereas u_k is the angular rates from the gyroscope. The matrix equation (17) shows the matrix form state-space representation of the process considered in this paper, the states are the roll (ϕ) and pitch (θ) angles as well as their biases β_ϕ and β_θ respectively. Comparing (14) and (17), the matrices A and B can be deduced while the control vector u_k is the angular rates from the gyroscope $[\omega_\phi, \omega_\theta]^T$. The measurement matrix H is obtained from the relationship between the current states and future states as (18).

$$\alpha_k = \alpha_{k-1} - \beta x T + u_k x T \tag{16}$$

$$\begin{bmatrix} \phi \\ \beta_x \\ \theta \\ \beta_y \end{bmatrix}_k = \begin{bmatrix} 1 & -T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi \\ \beta_x \\ \theta \\ \beta_y \end{bmatrix}_{k-1} \begin{bmatrix} T & 0 \\ 0 & 0 \\ 0 & T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} \tag{17}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{18}$$

Different approaches to determining measurement noise covariance matrix R and the process noise covariance matrix Q are documented in the literature [82], but in this paper, a method based on the IMU measurement error was considered. The IMU was placed level and the accelerometer readings were taken. The readings are supposed to be zero when the offset error is removed. Fifty readings were taken from the accelerometer for both roll and pitch to form a $2X50$ measurement noise vector. To obtain the process noise, the vibrations due to quadcopter propellers were modeled as the main contributor to process noise. A set of fifty readings were taken from the gyroscope roll and pitch with stationary quadrotor propellers, another set was taken with quadrotor propellers throttled to takeoff thrust. These sets of readings were then subtracted to form $2X50$ vector for process noise. The covariance matrices of these vectors were computed to form the measurement noise covariance matrix (19) and process noise (20) covariance matrix.

$$R = \begin{bmatrix} 0.0353 & -0.0062 \\ -0.0062 & 0.0393 \end{bmatrix} \tag{19}$$

$$Q = \begin{bmatrix} 190.8424 & 0 & -12.6294 & 0 \\ 0 & 0 & 0 & 0 \\ -12.6294 & 0 & 144.7351 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

The Kalman filter consists of two parts, the prediction of the state under consideration and the update of the predicted state. The prediction can be written as (21) and (22). The update part entails the computation of the measurement residual (23), the Kalman gain (24), the updated state estimate (25), and the updated error covariance (26). The prediction $\hat{x}_k^-$ predicts the current state using the state estimate from the previous time step and the current input u_{k-1}. The update part of the Kalman filter equation uses the measurement z_k to compute the measurement residual Y. The measurement refers to that the part of the system state that can be measured with certainty. Accelerometer roll and pitch angle estimates are used as the measurement in the Kalman filter equation.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{21}$$

$$P_k^- = A\hat{P}_{k-1}^+ A^T + Q \tag{22}$$

$$Y = z_k - H\hat{x}_k^- \tag{23}$$

$$K_k = P_k^- H^T (R + HP_k^- H^T)^{-1} \tag{24}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k Y \tag{25}$$

$$P_k^+ = (I - K_k H)P_k^- \tag{26}$$

Kalman filter roll and pitch angle estimates depicted in Fig. 10 show elimination of gyroscope drift by Kalman filtering. The Kalman filter uses the roll and pitch angle estimates from the accelerometer and the raw gyroscope outputs $[\omega_x, \omega_y]^T$.

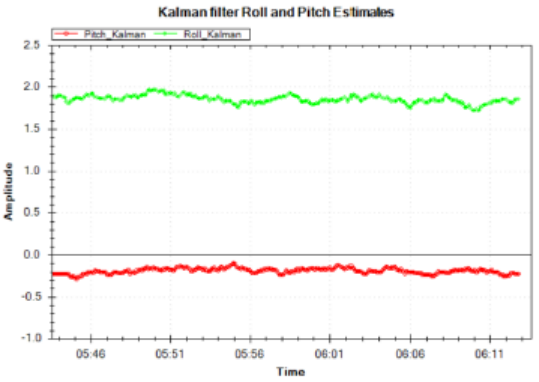

Fig. 10: Kalman Filter Roll and Pitch Angles.

8 Results and Discussions

8.1 PID Controller

A PID controller was tuned empirically for quadrotor control. The Kalman filter roll estimates were used to implement PID roll-axis control on the test platform. Figure 11 depicts the approach taken for system development. The PID controller was considered tuned when it could stabilise the test platform and also ensuring the platform system has a good settling time after disturbance. Figure 12 shows the performance of the PID controller when subjected to disturbance. The disturbance was introduced into the system by flicking underneath the propeller with the hand. The system demonstrated good performance, returning to its steady state in about 2 seconds.

Figure 13 shows the transient and steady-state response of the quadrotor system when the roll-reference angle signal was provided by the RC. The quadrotor system, at a steady-state, was given a reference signal of about 27° on the positive roll-axis. The reference-signal input is not a perfect step input, nevertheless, step response time analysis suffices to give insight into the performance of the system. Time response analysis provides detailed information in terms of the responsiveness of the system, and it was therefore considered for system performance evaluation in this work. Figure 14 shows an elaborated view of the system response. A small perturbation in the reference signal at $t = 39.41$ seconds causes a slight response of the system. This happens after the system has come to a steady-state value of 28.73°. The peak value of the system response is 32.21°. Table (2) gives the characteristic time responses of the system.

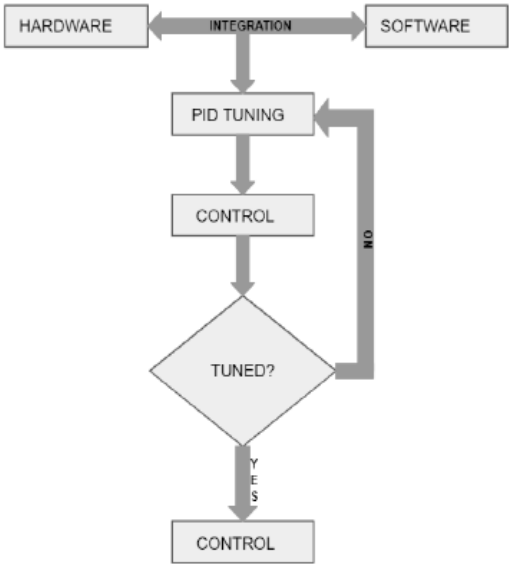

Fig. 11: Hardware-software integration and PID tuning for quadrotor control.

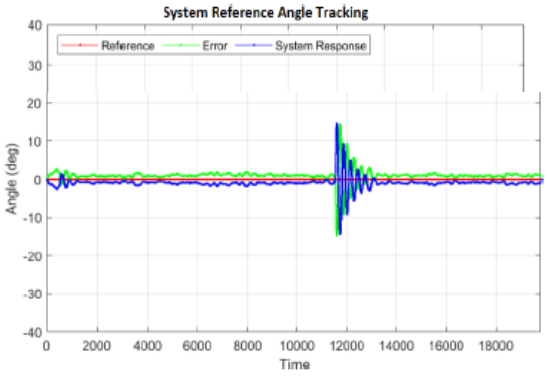

Fig. 12: PID controlled system response to disturbance with reference angle set to 0°.

8.2 NN Controller Development

8.2.1 Data for NN Model Training

The NN model was developed using data from the PID controlled UAV system. The IMU roll angle, the control error, the gyroscope rate about x-axis, and the PID output value for corrective action were considered for training data. This constituted a four column data set depicted in Fig 15, the PID corrective output value is the output to be predicted by the NN model, while the first three columns are input data. NN model training was

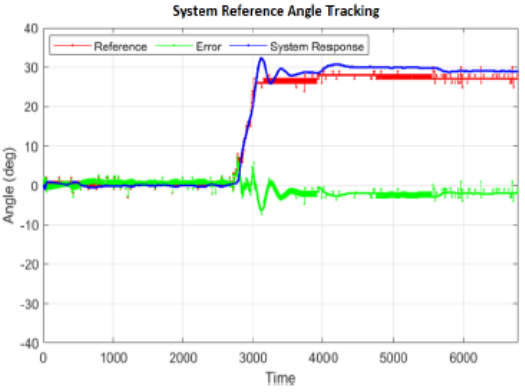

Fig. 13: PID system transient and steady state response.

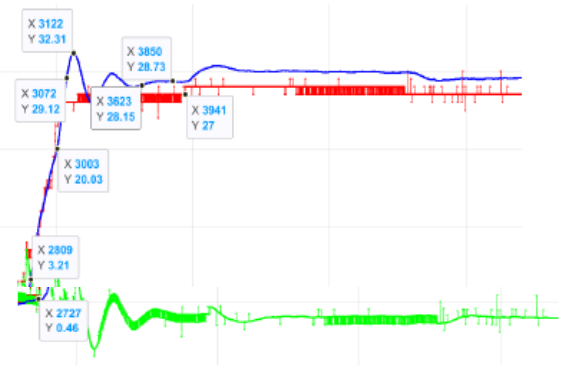

Fig. 14: PID system transient and steady state response elaboration.

Table 2: System Time Response Characteristics [2]

Parameter	Description	Magnitude
T_{step}	0% - 63.2 % of peak value	2.76s
T_{rise}	10% - 90 % of peak value	2.63s
dead time	T_{step} - T_{rise}	0.13s
settling time	response error within 2% of steady state value	8.96s
SSE	R_s−SS	1.73°

[1] T_{step} –step response time; T_{rise}- rise time; SSE – steady state error; R_s – reference signal input (27° in this case); SS – steady state value (28.73° in this case)

therefore a supervised learning approach which aims to approximate the PID corrective output. The success of the NN model depends on the choice of the training data parameters. A combination of input parameters are needed in the training data set that best estimate the output parameter. A strong correlation between the input and the output has to exist in the training data set. Therefore a careful selection of the parameters has to be made for the best prediction of the output parameters. Input parameter combination of thrust, roll angle, and reference angle from the RC controller have demonstrated poor prediction of PID corrective action. Critical analysis on the choice of training data parameters considered the fact that the PID controller, widely adopted in UAV control, uses control error. This informs on the proportion of the corrective action to be taken. It was observed that for better performance the NN model needs knowledge of the platform attitude in terms of the roll angle in addition to the control error. Furthermore, inclusion of the gyroscope rate ensured that the NN model acquires knowledge about disturbances. A total of 155000 training and 5947 testing data sets were captured. The data sets were taken from the PID controlled platform, all possible movements of the platform as well as disturbances were simulated. During training, the training data set was split, 60% for training and 40% for validation and testing.

IMU Roll	Control error	Gyro rate	Predicted Output
1.02	-1.02	556	104.06
0.86	-0.86	87	11.41
0.76	-0.76	-254	-56.13
0.7	-0.7	-543	-113.51
0.58	-0.58	-634	-130.89
0.61	-0.61	-697	-143.65
0.65	-0.65	-695	-143.58
0.72	-0.72	-417	-88.41
0.75	-0.75	-150	-35.26
0.84	-0.84	55	5.14
0.86	-0.86	197	33.41
0.82	-0.82	362	66.65
0.77	-0.77	442	83.02
0.7	-0.7	437	82.49
0.64	-0.64	473	90.11
0.56	-0.56	341	64.25
0.5	-0.5	202	36.92
0.48	-0.48	81	12.86

Fig. 15: Sample training data for NN model.

8.2.2 NN Model Development and Training

The NN model was developed using ANNHub platform. ANNHub is a machine learning platform that enables ML model design, training, and validation. It facilitates the user to develop ML models and export them to various applications to solve real-life problems. ANNHub optimises models to be able to fit into resource constrained MCUs such as the arduino-uno used in this implementation. It supports advanced training algorithms such as Scaled Conjugate Gradient, Levenberg Marquardt, Quasi-Newton, and Bayesian regularisation. The Bayesian regularisation showed good performance. Model over-fitting problem is handled by early stopping through selection of few but adequate training epochs or by the Bayesian regularisation automatically. Training data is first uploaded to ANNHub before configuring the model. Figure 16 shows the NN model configuration. The mean square error (MSE) was used as a cost function. The model was then trained, as shown in Fig 17. The model has an MSE of about 48, this implies a root mean squared error (RMSE) of about 6.9. In this case the quantity being predicted is the pulse width modulated (PWM) value written to the electronic speed controllers (ESCs), and hence no specific units for MSE and RMSE. The RMSE of 6.9 suffices given the output data range of 800. Furthermore lower RMSE implies, in most cases, an over-fitted model that lacks the generalisation ability.

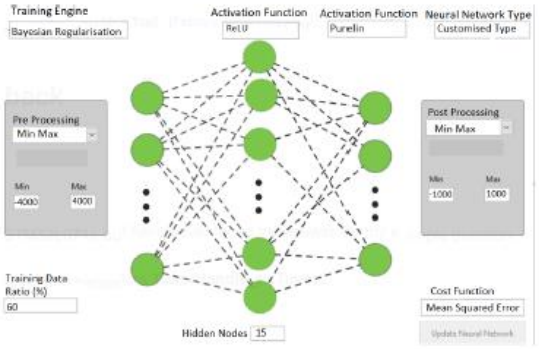

Fig. 16: NN model configuration.

8.2.3 NN Model Testing and Evaluation

The trained model was tested using an unseen data set to the NN inference. Figure 18 depicts the data set used as well as the NN inference prediction of the corrective output. The NN model prediction is satisfactory and al-

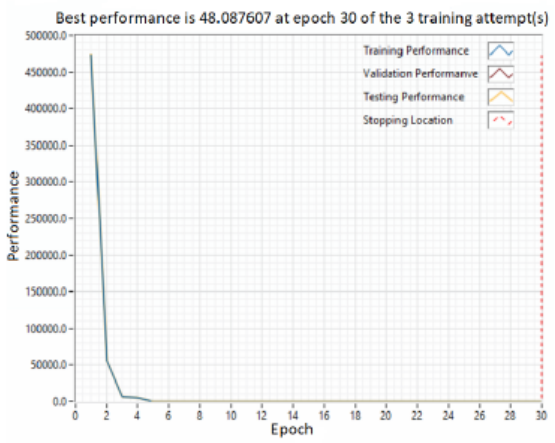

Fig. 17: NN model training.

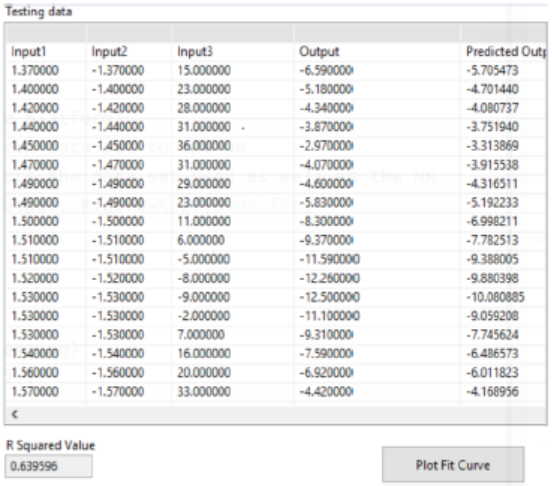

Fig. 18: Testing data and ANNHub NN model prediction.

most within the range of the actual output. The prediction has an R^2 value of about 0.64. R is the correlation coefficient and R^2 is the coefficient of determination. These quantities are widely used to evaluate the goodness of NN models. The value of R has a range of $-1-1$ while R^2 has a range of $0-1$. A value close to 1 indicate a strong positive correlation between the predicted and actual output while a value close to -1 indicates a strong negative correlation between the two variables. Figure 19 shows the explanations of R (as well as R^2) for various R (R^2) ranges, where R^2 takes the positive half of the range $-1-1$. It is indicative, therefore, that the R^2 value of this NN inference model falls in the strong category and has positive correlation. The value of 0.64 for R may seen low but it suffices given that the range of the output data is relatively high, about 800. The corresponding R value is about 0.8. Figure 20 shows a plot fit of the NN prediction to the actual output. It shows a quite good fit with much resemblance between the predicted and actual output.

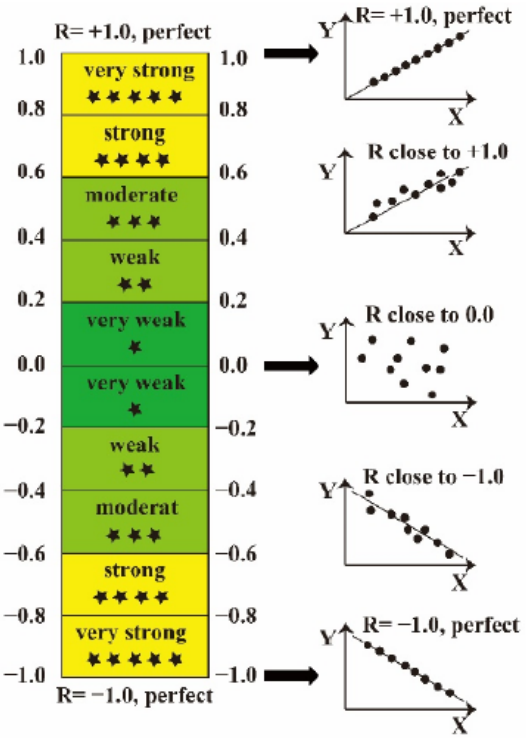

Fig. 19: Explanation of R (R^2) values, reprinted from [83]

8.3 NN Model Deployment and Comparison with the PID Controller

So far the ANN inference has been tested only within the ANNHub platform. ANNHub has provision for exporting the trained model for use in real world application. The model was exported for use on arduino IDE. The generated arduino file for the model was integrated with the rest of the arduino code for the test platform. The code was then uploaded to the onboard arduino flight controller. The performance of the model on the platform was evaluated. The main purpose of the model is to estimate the corrective action to the motors for

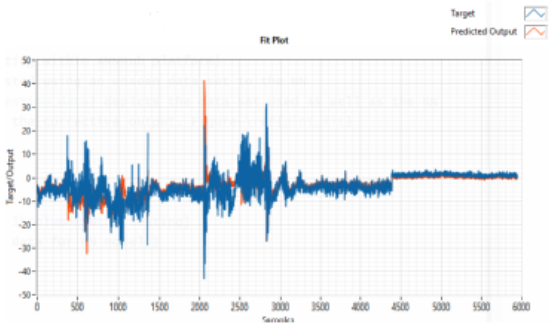

Fig. 20: ANNHub fit-plot depicting how the NN model approximate the true output.

platform stabilisation. Figure 21 shows the performance of the NN controller when the test platform was subjected to disturbance. The disturbance was introduced by flicking underneath the propeller with the hand. Response shows fewer oscillations as compared to the PID controller, see Fig 12 for comparison. This indicates that the NN model handles system non-linear dynamics better than the PID controller. The system also returns to a stable state faster as compared to the PID controller. Figure 22 shows the transient and steady state response of the test platform subject to NN controller action. Figure 23 shows an expanded view of the platform system response. The time response characteristics of the system are shown in Table (3), the table also includes the PID controller time response characteristics for comparison. The NN controller shows overall good performance, it has a lower settling time, as well as a lesser overshoot. See Fig 24, with a link for the performance of the NN model.

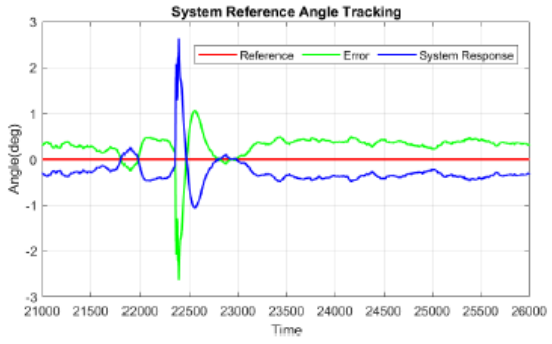

Fig. 21: NN system response to disturbance with reference angle set to 0°.

Although the NN model was developed using data from the PID controller, it offers significant advantages in this implementation approach. First of all, NN model considers all PID controller performance and uses the performance data to relate input to output. Few occasions of PID controller under-performance are not strongly captured during model training as the NN model will only capture the predominant data features. This is shown in less oscillations with the NN controller. Furthermore the platform algorithm is shortened with the use of an NN model leading to faster computations and high motor update frequency. This is due to parallel processing as the NN model takes in input variables simultaneously as they are generated while the PID algorithm needs the error to be calculated first. The generalisation ability of the NN modeled ensured good performance even with the roll angle estimate from the accelerometer. This further shortened the control algorithm and reduced computational burden from the flight controller MCU due the heavy dependence of the kalman filter algorithm on matrix calculations. The platform algorithm has a loop time of 10ms with the PID algorithm compared to 6ms with the NN algorithm, this corresponds to motor update frequencies of $100Hz$ and $167Hz$ respectively. This can be attributed to the smaller rise time with the NN model depicted in Table (3).

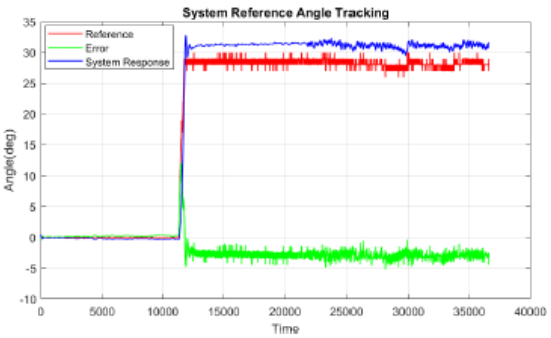

Fig. 22: NN system transient and steady state response.

9 Conclusion

The work of this paper furthers the adoption of NN controllers for UAV control. UAV control systems are dominated by PID controllers due to their ease of implementation despite their difficulty in tuning. The NN controller developed in this paper demonstrated better performance in attitude control of a single axis plat-

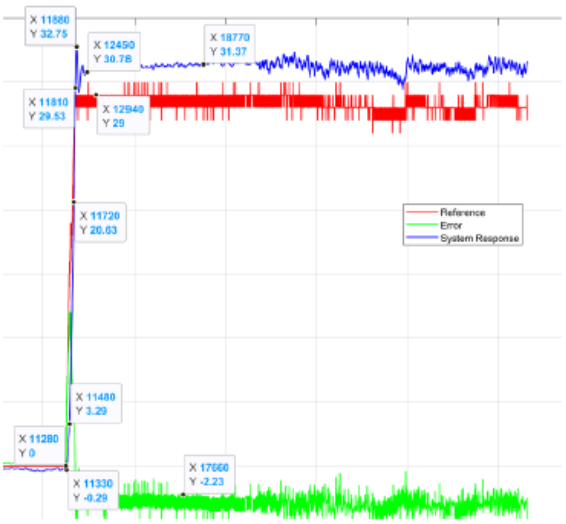

Fig. 23: NN system transient and steady state response elaboration.

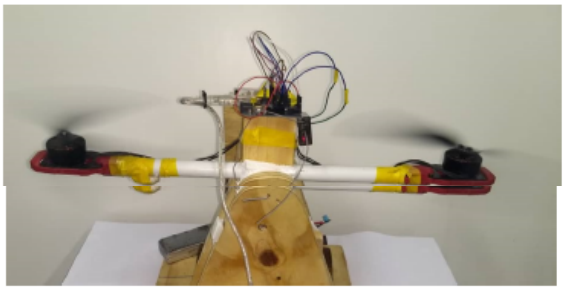

Fig. 24: Neural Networks controller used for attitude control, video can be viewed at https://www.youtube.com/watch?v=tWPMAHZIBCs.

form. This was enabled by advancement in TinyML where NN models are optimised for implementation on resource constrained embedded computers. This work, therefore, is a motivation for improvement and further works on UAV NN attitude controllers. The developed NN model cannot be effective in real life applications on full UAV dynamics control due to limited training data used. The future direction of this research would require more training data that will cover all possible UAV maneuvers. This will ensure that the built NN model will be extensively exposed to diverse UAV operational environments. In addition, the model can be more extensive if it includes all possible environmental disturbances. However this requirement can not be met by performing several experiments. Thus, external disturbance should be incorporated into NN learning to ensure that such environmental disturbances and UAV maneuvers are included in the NN model.

Table 3: System Time Response Characteristics [3]

Parameter	Description	PID controller	ANN controller
T_{step}	0% - 63.2 % of peak value	2.76s	2.34s
T_{rise}	10% - 90 % of peak value	2.63s	1.98s
dead time	T_{step} - T_{rise}	0.13s	0.3s
settling time	response error within 2% of steady state value	8.96s	6.72s
SSE	R_s−SS	1.73°	2.3°
overshoot	peak value - R_s	4.16°	3.75°

[1] T_{step} –step response time; T_{rise}- rise time; SSE – steady state error; R_s – reference signal input SS – steady state value

Acknowledgment

The authors would like to acknowledge the funding support on this work from the Botswana International University of Science and Technology (BIUST) Drones Project with project number P00015.

10 Declarations

10.1 Funding

Funding was provided by the Botswana International University of Science and Technology (BIUST) Drones Project, with project number P00015. BIUST provided research facilities (laboratory), the design of the study, collection, analysis, interpretation of data, and the writing of the manuscript was purely the work of the authors.

10.2 Conflicts of Interest

The authors declare that they have no competing interests.

10.3 Code or data availability

Not applicable.

10.4 Author's contributions

VK is the main author and corresponding author. He ensures that the research direction is properly understood and well-planned based on the discussions with the supervisor. He is responsible for writing the manuscript, experimental work, data analysis and writing of code. RJ is the supervisor of this work. He oversees the research contribution, practical implementation, and steers the direction of research. L.M participated in experimental works and writing code.

10.5 Ethics approval

Not applicable.

10.6 Consent to participate

Not applicable.

10.7 Consent for publication

Not applicable.

References

1. G. Skorobogatov, C. Barrado, E. Salamí, Unmanned Systems **8**(02), 149 (2020)
2. F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, Y. Ren, IEEE Transactions on Cloud Computing **7**(3), 866 (2017)
3. W. Mmereki, R.S. Jamisola, D. Mpoeleng, T. Petso, in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)* (IEEE, 2021), pp. 241–246
4. R. Shakeri, M.A. Al-Garadi, A. Badawy, A. Mohamed, T. Khattab, A.K. Al-Ali, K.A. Harras, M. Guizani, IEEE Communications Surveys & Tutorials **21**(4), 3340 (2019)
5. P. Fraga-Lamas, L. Ramos, V. Mondéjar-Guerra, T.M. Fernández-Caramés, Remote Sensing **11**(18), 2144 (2019)
6. B. Makgantai, N. Subaschandar, R.S. Jamisola, in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)* (2021), pp. 1459–1465. DOI 10.1109/ICUAS51884.2021.9476801
7. M. Liu, X. Wang, A. Zhou, X. Fu, Y. Ma, C. Piao, Sensors **20**(8), 2238 (2020)
8. T. Petso, R.S. Jamisola Jr, D. Mpoeleng, E. Bennitt, W. Mmereki, Ecological Informatics **66**, 101485 (2021)
9. P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, I. Moscholios, Computer Networks **172**, 107148 (2020)
10. K. Yakushiji, H. Fujita, M. Murata, N. Hiroi, Y. Hamabe, F. Yakushiji, Drones **4**(4), 68 (2020)
11. J. Kim, S. Lee, S. Lee, Y. Kim, C. Song, IEEE Access (2021)
12. J.F. Horn, E.M. Schmidt, B.R. Geiger, M.P. DeAngelo, Journal of Guidance, Control, and Dynamics **35**(2), 548 (2012)
13. H. Razmi, S. Afshinfar, Aerospace Science and Technology **91**, 12 (2019)
14. R.P. Padhy, S. Verma, S. Ahmad, S.K. Choudhury, P.K. Sa, Procedia computer science **133**, 643 (2018)
15. S. Back, G. Cho, J. Oh, X.T. Tran, H. Oh, Journal of Intelligent & Robotic Systems **100**(3), 1195 (2020)
16. F. Jiang, F. Pourpanah, Q. Hao, IEEE Transactions on Industrial Electronics **67**(3), 2076 (2019)
17. H. Li, C. Li, H. Li, Y. Li, Z. Xing, IEEE Access **5**, 10941 (2017)
18. H. Chao, A. M., Y. Han, Y. Chen, M. Mckee, Advances in Geoscience and Remote Sensing (2009). DOI 10.5772/8333
19. M. Labbadi, M. Cherkaoui, Aerospace Science and Technology **93**, 105306 (2019)
20. J. Muliadi, B. Kusumoputro, Journal of Advanced Transportation **2018** (2018)
21. W. Koch, R. Mancuso, R. West, A. Bestavros, ACM Transactions on Cyber-Physical Systems **3**(2), 1 (2019)
22. A. Zulu, S. John, arXiv preprint arXiv:1602.02622 (2016)
23. W. Wang, H. Ma, M. Xia, L. Weng, X. Ye, Mathematical Problems in Engineering **2013** (2013)
24. A. Ingabire, A.A. Sklyarov, in *E3S Web of Conferences*, vol. 104 (EDP Sciences, 2019), vol. 104, p. 02001
25. S. Mallavalli, A. Fekih, in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)* (IEEE, 2018), pp. 152–155
26. V. Kangunde, R.S. Jamisola, E.K. Theophilus, International Journal of Dynamics and Control pp. 1–15 (2021)
27. M. Noor-A-Rahim, M.O. Khyam, G.M.N. Ali, Z. Liu, D. Pesch, P.H. Chong, IEEE Transactions on Reliability **68**(3), 1061 (2019)
28. in *IOP Conference Series: Materials Science and Engineering*, vol. 260 (IOP Publishing, 2017), vol. 260, p. 012013
29. M.A. Al-Mashhadani, Measurement and Control p. 0020294019866860 (2019)
30. R.S. Jamisola, D.N. Oetomo, M.H. Ang, O. Khatib, T.M. Lim, S.Y. Lim, Advanced Robotics **19**(5), 613 (2005)
31. R. Jamisola, M.H. Ang, D. Oetomo, O. Khatib, T.M. Lim, S.Y. Lim, in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1 (IEEE, 2002), vol. 1, pp. 400–405
32. A. Rohan, M. Rabah, S.H. Kim, IEEE access **7**, 69575 (2019)
33. Y.L. Hsu, J.S. Wang, IEEE Access **7**, 17551 (2019)
34. Y.H. Tu, C.C. Peng, IEEE Sensors Journal **21**(3), 2712 (2020)
35. S. Li, Y. Gao, G. Meng, G. Wang, L. Guan, Electronics **8**(5), 594 (2019)
36. P. Beňo, M. Kubiš, Transportation Research Procedia **40**, 150 (2019)
37. X. Zhao, Y. Kang, in *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)* (IEEE, 2019), pp. 102–106
38. Z. Ruoyu, G. Shuang, C. Xiaowen, in *2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)* (IEEE, 2019), pp. 1754–1761
39. M. Wang, X. Dong, C. Qin, J. Liu, Measurement **167**, 108170 (2021)
40. H. Gu, X. Liu, B. Zhao, H. Zhou, IEEE Sensors Journal **19**(13), 5070 (2019)

41. Y.L. Hsu, J.S. Wang, C.W. Chang, IEEE Sensors Journal **17**(10), 3193 (2017)
42. T.Y. Pan, C.H. Kuo, H.T. Liu, M.C. Hu, IEEE Transactions on Emerging Topics in Computational Intelligence **3**(3), 261 (2018)
43. S. Xu, Y. Xue, X. Zhang, L. Jin, Pattern Recognition **116**, 107939 (2021)
44. Q. Huang, L. Li, H. Zhang, A. Gao, W. Yan, in *4th International Conference on Information Systems and Computing Technology* (Atlantis Press, 2016)
45. B. Sababha, H.C. Yang, O. Rawashdeh, in *AIAA Infotech@ Aerospace 2010* (2010), p. 3414
46. Y. Khosiawan, Y. Park, I. Moon, J.M. Nilakantan, I. Nielsen, Neural Computing and Applications **31**(9), 5431 (2019)
47. Z. Zheng, X. Guanping, Chinese Journal of Aeronautics **32**(1), 176 (2019)
48. M. Fatehnia, G. Amirinia, International Journal of Geo-Engineering **9**(1), 1 (2018)
49. D. Lee, K. Kim, Energies **12**(2), 215 (2019)
50. T. Akter, S. Desai, Materials & Design **160**, 836 (2018)
51. F. Rodríguez, A. Fleetwood, A. Galarza, L. Fontán, Renewable Energy **126**, 855 (2018)
52. I.M. Nasser, S.S. Abu-Naser, International Journal of Engineering and Information Systems (IJEAIS) **3**(3), 17 (2019)
53. A.P. Marugán, F.P.G. Márquez, J.M.P. Perez, D. Ruiz-Hernández, Applied energy **228**, 1822 (2018)
54. M. Salah, K. Altalla, A. Salah, S.S. Abu-Naser, International Journal of Engineering and Information Systems (IJEAIS) **2**(20), 11 (2018)
55. S.P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, S. Jain, in *2017 international conference on computer, communications and electronics (comptelix)* (IEEE, 2017), pp. 162–167
56. A. Addeh, A. Khormali, N.A. Golilarz, ISA transactions **79**, 202 (2018)
57. D.G. Barrett, A.S. Morcos, J.H. Macke, Current opinion in neurobiology **55**, 55 (2019)
58. G.C. Calafiore, S. Gaubert, C. Possieri, IEEE transactions on neural networks and learning systems **31**(12), 5603 (2020)
59. P. Verpoort, P. MacDonald, G.J. Conduit, Computational Materials Science **147**, 176 (2018)
60. D. Selvamuthu, V. Kumar, A. Mishra, Financial Innovation **5**(1), 1 (2019)
61. Y. Li, W. Jiang, L. Yang, T. Wu, Neurocomputing **275**, 1150 (2018)
62. M. de Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, N. Pazos, Sensors **21**(4), 1339 (2021)
63. R. Udendhran, M. Balamurugan, A. Suresh, R. Varatharajan, Microprocessors and Microsystems **76**, 103094 (2020)
64. S. Ramadurgam, D.G. Perera, Electronics **10**(11), 1323 (2021)
65. G. Signoretti, M. Silva, P. Andrade, I. Silva, E. Sisinni, P. Ferrari, Sensors **21**(12), 4153 (2021)
66. I. Martinez-Alpiste, P. Casaseca-de-la Higuera, J.M. Alcaraz-Calero, C. Grecos, Q. Wang, Journal of Field Robotics **37**(3), 404 (2020)
67. C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. Janapa Reddi, M. Mattina, P. Whatmough, Proceedings of Machine Learning and Systems **3**, 1 (2021)
68. R. Sanchez-Iborra, A.F. Skarmeta, IEEE Circuits and Systems Magazine **20**(3), 4 (2020)
69. A.H. Jebril, A. Sali, A. Ismail, M.F.A. Rasid, Sensors **18**(10), 3257 (2018)
70. M. Jia, Z. Yin, D. Li, Q. Guo, X. Gu, IEEE Internet of Things Journal **6**(3), 4252 (2018)
71. P. Achararit, M.A. Hanif, R.V.W. Putra, M. Shafique, Y. Hara-Azumi, IEEE Access **8**, 165319 (2020)
72. M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, M. Sjödin, Microprocessors and Microsystems **73**, 102989 (2020)
73. T. Cassimon, S. Vanneste, S. Bosmans, S. Mercelis, P. Hellinckx, Internet of Things **12**, 100234 (2020)
74. Z. Alameer, M. Abd Elaziz, A.A. Ewees, H. Ye, Z. Jianhua, Resources Policy **61**, 250 (2019)
75. M. Anemangely, A. Ramezanzadeh, B. Tokhmechi, A. Molaghab, A. Mohammadian, Journal of Geophysics and Engineering **15**(4), 1146 (2018)
76. D. Li, F. Huang, L. Yan, Z. Cao, J. Chen, Z. Ye, Applied Sciences **9**(18), 3664 (2019)
77. Z. Tahir, M. Jamil, S.A. Liaqat, L. Mubarak, W. Tahir, S.O. Gilani, Indian Journal of Science and Technology **9**, 27 (2016)
78. A. Noordin, M.A.M. Basri, Z. Mohamed, International Journal of Electrical and Electronic Engineering and Telecommunications **7**(4) (2018)
79. Á. Odry, R. Fuller, I.J. Rudas, P. Odry, Mechanical systems and signal processing **110**, 569 (2018)
80. V. Kangunde, L.O. Mohutsiwa, R.S. Jamisola, in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)* (2021), pp. 188–194. DOI 10.1109/ICUAS51884.2021.9476809
81. H. Ferdinando, H. Khoswanto, D. Purwanto, in *2012 International Symposium on Innovations in Intelligent Systems and Applications* (IEEE, 2012), pp. 1–5
82. F. Cui, D. Gao, J. Zheng, in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)* (IEEE, 2018), pp. 1–6
83. A. Jierula, S. Wang, T.M. Oh, P. Wang, Applied Sciences **11**(5), 2314 (2021)