The Aeronautical Journal (2026), 1–25
doi:10.1017/aer.2026.10143

RESEARCH ARTICLE

Versatile UAV hardware platform for accelerating indoor aerial navigation research

Kyriakos M. Deliparaschos[1], Xuefei Huang[1], Michalis Neofytou[2,3], Savvas G. Loizou[4] and Argyrios Zolotas[5]

[1]Electrical and Computer Engineering and Informatics Department, Cyprus University of Technology, Limassol, Cyprus
[2]NMN Systems LTD, Limassol, Cyprus
[3]Cyprus University of Technology: Technologiko Panepistemio Kyprou, Cyprus
[4]Department of Mechanical Engineering and Materials Science and Engineering, Cyprus University of Technology, Limassol, Cyprus
[5]Faculty of Engineering and Applied Sciences, Cranfield, United Kingdom
Corresponding author: Kyriakos M. Deliparaschos; Email: k.deliparaschos@cut.ac.cy

Received: 5 January 2025; **Revised:** 22 January 2026; **Accepted:** 22 January 2026

Keywords: UAS testbeds; navigation; FPGA SoC; high-level synthesis; Delaunay triangulation; ROS 2

Abstract

This study presents a bespoke hardware platform for indoor navigation, featuring a quadrotor equipped with an FZ3 card incorporating the AMD (formerly Xilinx) Zynq UltraScale+ ZU3EG MPSoC as the onboard computer. A core component of this platform is a field programmable gate array (FPGA) module specifically designed to efficiently compute Delaunay triangulations, enabling enhanced spatial awareness and real-time surface reconstruction. The onboard computer communicates with the flight controller, inertial measurement unit (IMU), ultra-wideband (UWB) localisation system, stereo camera, light detection and ranging (LiDAR) and ultrasonic sensors via robotic operating system (ROS) 2. The primary objective is to develop a cost-effective, modular unmanned aerial vehicle (UAV) system that can be adapted for a range of indoor navigation applications. The modular design supports different onboard computer platforms and sensor configurations, allowing researchers to easily customise the system for various experiments. By providing a practical framework for precise indoor navigation, this platform addresses the limitations of simulated, simplified laboratory setups, accelerating prototyping and supporting the deployment of UAVs in complex real-world environments. This work explores the UAV's hardware architecture, the implementation of the Delaunay triangulation core on the FPGA system-on-chip (SoC), the ROS 2-based communication system and includes a detailed mass analysis and power estimation.

Nomenclature

DDS	data distribution service
DT	Delaunay triangulation
ENU	East-North-Up
ES	electronic speed controller
FCU	flight controller unit
FLU	forward-left-up
FPG	field programmable gate array
HDL	hardware description language
HLS	high-level synthesis
IMU	inertial measurement unit
IP	intellectual property
KF	Kalman filter
LiDAR	light detection and ranging

© The Author(s), 2026. Published by Cambridge University Press on behalf of Royal Aeronautical Society. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.
https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

MCC	maximum correntropy criterion
PCD	point cloud dataset
PDB	power distribution board
RFID	radio frequency identification
ROS	robotic operating system
RTL	register-transfer level
SDK	software development kit
SoC	system-on-chip
ToF	time-of-flight
UAV	unmanned aerial vehicle
UWB	ultra-wideband

Other nomenclature and symbols are defined as they appear.

1.0 Introduction

Unmanned aerial vehicles (UAVs) have significantly expanded research horizons, particularly in aerial navigation [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Despite extensive research efforts focused on deploying UAV systems with various navigation methods, especially for the challenging task of aerial indoor navigation, many proposed methodologies predominantly rely on simulations [11, 12]. These simulation-based approaches often fall short in meeting real-world deployment requirements.

This work presents a low-cost modular UAV hardware platform specifically designed to facilitate aerial indoor navigation research and system validation. Emphasising modularity, the platform provides a flexible and adaptable foundation for experiments, allowing researchers to customise and scale their setups within reasonable payload constraints to meet the specific demands of indoor environments. The work strongly contributes to the important area of open and explainable unmanned vehicle architectures for accelerating prototyping development and system deployment. One of the core modules is a field programmable gate array (FPGA) unit capable of efficiently computing Delaunay triangulations [13]. Initially, triangulation calculations are completed using an incremental algorithm, with the C++ code subsequently converted into hardware description language (HDL) through high-level synthesis (HLS). After optimisation in simulation, the module is deployed onto the FPGA. The Delaunay triangulation core plays a crucial role in the UAV's data processing, enabling efficient real-time surface reconstruction for spatial awareness. Delaunay triangulation being of particular interest in aerospace engineering and navigation application studies [14, 15]. Experimental results are presented in this paper [16]. While this approach strongly accelerates prototyping and development, it also enables detailed investigation into UAV navigation precision and reliability in indoor spaces. Through rigorous investigation of its architecture, capabilities and potential applications, we demonstrate how the proposed platform advances UAV indoor navigation research development [17, 18].

The paper is structured as follows. Section 2 discusses the hardware description for the quadrotor setup. Section 3 explores the overall architecture of the UAV system, detailing the integration and interaction of various components essential for autonomous operation. Section 4 focuses on a key element in the UAV's data processing, enabling efficient real-time surface reconstruction for spatial awareness: the Delaunay triangulation core. Section 5 presents the results, including key performance metrics, while Section 6 concludes the paper and discusses potential future enhancements.[1]

2.0 Hardwear Description

The finalised quadrotor project integrates a variety of essential components that are crucial to its successful operation [19]. This section offers a detailed overview of the key hardware elements selected for

[1]This work is a substantially extended version of the authors' 2024 ICUAS conference paper, titled 'A Modular UAV Hardware Platform for Aerial Indoor Navigation Research and Development', with the implementation of the Delaunay triangulation core on the FPGA SoC mission computer for real-time spatial mapping.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

the system, alongside an analysis of the decision-making process underpinning their selection. Each component was carefully assessed based on specific criteria, including performance, compatibility, power efficiency and its overall contribution to the system's functionality. By examining the rationale behind each choice, this section underscores the considerations and trade-offs involved in constructing a reliable and efficient quadrotor UAV.

2.1 Inertial measurement unit

An inertial measurement unit (IMU) is a device that utilises a combination of accelerometer, gyroscope and magnetometer sensors to determine a body's orientation, velocity and gravitational forces. Often, it is paired with other sensors, such as vision-based or wireless technologies, to enhance pose estimation accuracy. When other sensors are unavailable, the IMU can act as the primary sensor. The IMU provides measurements of a quadrotor's orientation through the three Euler angles: roll, pitch and yaw. The Pixhawk PX4 flight control unit (FCU) includes three internal sensor chips, namely the Invensense MPU 6000 as the primary 3-axis accelerometer/gyroscope, the ST Micro L3GD20 3-axis 16-bit gyroscope and the ST Micro LSM303D 3-axis 14-bit accelerometer/magnetometer.

2.2 Flight controller unit

The FCU is responsible for controlling the UAV's movement and regulating the power delivered to each motor. Flight controllers assess the UAV's level and speed, using this data to correct its orientation during flight. For this project, the Readytosky Pixhawk PX4 2.4.8 32-bit flight controller was chosen due to its advanced capabilities beyond the standard gyroscope and accelerometer sensors found in most flight controllers. The Pixhawk incorporates a barometer, magnetometer and IMU, enhancing the precision and reliability of flight performance. Furthermore, the Pixhawk is a 32-bit controller equipped with I2C and SPI interfaces and supports both the PX4 and ArduPilot flight stacks.

2.3 Onboard computer options

The quadrotor's onboard computer, running the robot operating system 2 (ROS 2), manages critical tasks such as processing stereo camera frames for image recognition, handling ground ultra-wideband (UWB) sensor data, analysing point clouds from the light detection and ranging (LiDAR) sensor, integrating measurements from the flight controller, transmitting commands to the flight controller and streaming video to the ground station. To meet the need for a powerful, lightweight and energy-efficient computer, the FZ3 card was selected.

The FZ3 card [20], featuring an AMD Zynq UltraScale+ ZU3EG MPSoC, is employed for real-time point cloud processing. Leveraging a dedicated Delaunay triangulation core [16], developed by the authors in prior work, within the programmable logic (PL) FPGA portion of the Zynq UltraScale+ ZU3EG MPSoC further enhances its efficiency in processing point cloud data.

2.4 Sensors: camera, LiDAR, ultrasonic sensors

Camera-based landmark detection is a well-established method for robot localisation. This technique employs a camera mounted on the quadrotor and a set of landmarks whose positions are known relative to the global reference frame. By analysing objects detected by the camera, the quadrotor's position can be estimated. Using a monocular camera, depth information about the landmarks can be determined by processing successive frames. This enables estimation of the relative position between a landmark and the camera. Once the camera's relative position is known, the quadrotor's relative position, to which the camera is attached, can be calculated and expressed within the global reference frame.

Initially, we explored using a single basic RGB camera to capture video data for an object detection algorithm that would identify object locations. However, this setup posed challenges in accurately determining object distances. To address this limitation, we selected the ZED 2 stereo camera [21], which computes distances by analysing pixel differences between two images. The stereo camera connects

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

via USB 3.0 to the FZ3 card. A drawback of this solution is the significant processing power required, particularly when simultaneously running a computationally intensive object detection algorithm.

Alternatively, a LiDAR sensor was considered as a less computationally intensive approach compared to using two RGB cameras. The point cloud generated by the LiDAR sensor allows object recognition while also determining object distances through the reconstructed surface created by the Delaunay triangulation core [16], implemented on the FZ3 accelerator card.

The LiDAR sensor is invaluable for detecting obstacles within the environment [22]. For our system, we integrated a Hokuyo URG-04LX-UG01 2D LiDAR [23, 24] onto the UAV. This sensor offers a range of 20mm to 5500mm (with a resolution of 1mm) and a 240° scan window with an angular resolution of 0.36°. Its low power consumption (2.5W) and lightweight design (138g) make it an ideal choice for UAV applications.

For additional range sensing, we selected the Atmel ATMEGA328P microcontroller, mounted on an Arduino Pro Mini board, to manage four laterally mounted HC-SR04 ultrasonic range sensors. This 8-bit, 16 MHz microcontroller is well-suited for driving ultrasonic sensors and transmitting distance data to the processing module via UART or I2C communication. Its compact form factor, lightweight build, low power consumption and user-friendly development environment make it an excellent choice for this application. Moreover, the extensive documentation available for the Arduino platform enhances its integration and reliability in distance measurement tasks.

2.5 Communication modules: Wi-Fi and ultra-wideband (UWB)

Wi-Fi connectivity is essential for enabling communication between the onboard UAV computer and the ground control computer, particularly for the transmission of video data. As the FZ3 card does not include an integrated Wi-Fi module, the AC1300 Archer T3U Mini Wireless MU-MIMO USB adapter was selected. The choice of a USB Wi-Fi adapter, rather than a dedicated Wi-Fi card, was made to simplify system integration, as the latter would require an external antenna for effective operation. The compact USB module incorporates a built-in antenna, thereby reducing hardware complexity and easing installation.

The selected adapter also supports multi-user multiple-input multiple-output (MU-MIMO) technology, which enables two simultaneous data streams. When used in conjunction with a compatible MU-MIMO router, this feature improves overall throughput and network efficiency.

Ultra-wideband (UWB) technology was additionally incorporated to provide accurate localisation capabilities. UWB is a wireless communication technology that employs short, low-energy pulses transmitted over a wide bandwidth, making it inherently robust to multipath interference. In contrast to conventional radio-frequency identification (RFID) systems that operate within narrow frequency bands, UWB spans a broad spectrum, resulting in low power spectral density and reduced interference with other radio-frequency signals.

A key advantage of UWB lies in its precise time-of-flight (ToF) measurement capability, which enables accurate distance estimation and makes the technology particularly well suited for indoor localisation. A typical UWB localisation setup consists of wireless transmitters, referred to as anchors, positioned at fixed locations, and a receiver, known as a tag, mounted on the quadrotor. The tag measures the arrival times of signals transmitted by the anchors to estimate the quadrotor's position in three-dimensional space. Conversely, if the anchors are arranged in a known geometric configuration and the initial position of the tag is specified, the anchors' positions may also be inferred.

The UWB system employed in this work is based on the Pozyx platform [25], which utilises the Decawave DWM1000 transceiver.

In addition to Wi-Fi and UWB communication, radio control functionality is provided by a Radiolink R8EF receiver. This 8-channel receiver operates at a frequency of 2.4 GHz and is connected to the quadcopter's FCU. Specifically, the SBUS output (channel 1) is interfaced with the Pixhawk RC input. The receiver relays control commands generated by the radio transmitter based on pilot stick inputs, thereby enabling precise manual control of the quadcopter when required.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

2.6 Power components: battery, DC-to-DC converter

The quadrotor is powered by a lithium polymer (LiPo) battery, connected to a power distribution board (PDB) that distributes power to the various components. The selected battery is a 3S1P LiPo battery (three cells connected in series) with a nominal voltage of 11.1 volts, a capacity of 4400 mAh, a 30C discharge rate, and a mass of 453g. The choice of this battery was based on its balance of output voltage and mass.

To power the FZ3 card from the UAV's power distribution board, we employed a buck converter. A buck converter, also known as a step-down converter, is a DC-to-DC converter that reduces **input** voltage while increasing output current. This type of converter falls under the category of switched-mode power supplies.

The FZ3 card requires a 12V DC power supply with a maximum current of two amps. Ensuring adequate power is critical, particularly during computationally intensive tasks such as running object detection algorithms, to prevent a CPU shutdown.

2.7 Propulsion and control system

Four A2212 brushless DC motors (BLDC), rated at 1000 kV (1000 rpm/V)[2], were selected for the propulsion system, each exhibiting a maximum efficiency of approximately 80%. The motors are driven by 30 A electronic speed controllers (ESCs) and are equipped with 10×4.5 inch propellers in both clockwise (CW) and counter-clockwise (CCW) configurations. Each motor is capable of producing a maximum thrust of approximately 7.85 N, corresponding to 800 gram-force (gf).

The propulsion system components operate in unison to ensure stable and efficient flight. Despite their name, brushless DC motors are not true DC machines but rather polyphase synchronous motors supplied by electronically commutated currents. In these motors, DC power is converted into alternating currents through electronic switching, generating a rotating magnetic field within the stator. In contrast to brushed DC motors, which rely on mechanical commutation via brushes, BLDC motors achieve commutation electronically, resulting in improved efficiency, reduced wear, and enhanced reliability.

3.0 System Architecture

This section details the UAV system architecture, emphasising the central processing units and sensor modules that enable autonomous operation (navigation, obstacle detection and control). By outlining the architectural framework, this section lays the groundwork for understanding how individual subsystems work together to ensure reliable quadrotor performance.

3.1 UAV overall architecture

The UAV system architecture is designed for efficient, autonomous operation and features several key components. Central to this architecture is the FZ3 card, equipped with a Zynq UltraScale+ ZU3EG MPSoC, which manages high-level computations, decision-making and surface reconstruction implementation on the PL side. The flight controller, specifically the Readytosky Pixhawk, connects to the FZ3 card and is responsible for managing the UAV's movement and motor control. The system architecture is shown in Fig. 1.

Sensory input is provided by various sensors, including a stereo camera for image recognition, an IMU for orientation data, UWB sensors for localisation and a LiDAR sensor for point cloud data. These sensors collectively provide essential information for navigation and obstacle avoidance. The system is powered by a LiPo battery connected to a PDB, which supplies power to all components. For communication, the AC1300 Archer T3U Mini Wireless MU-MIMO USB adapter enables data exchange with a

[2]A three-cell LiPo battery, with a nominal voltage of 11.1 V, would theoretically cause a 1000 kV motor to spin at approximately 11,100 rpm under no-load conditions.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

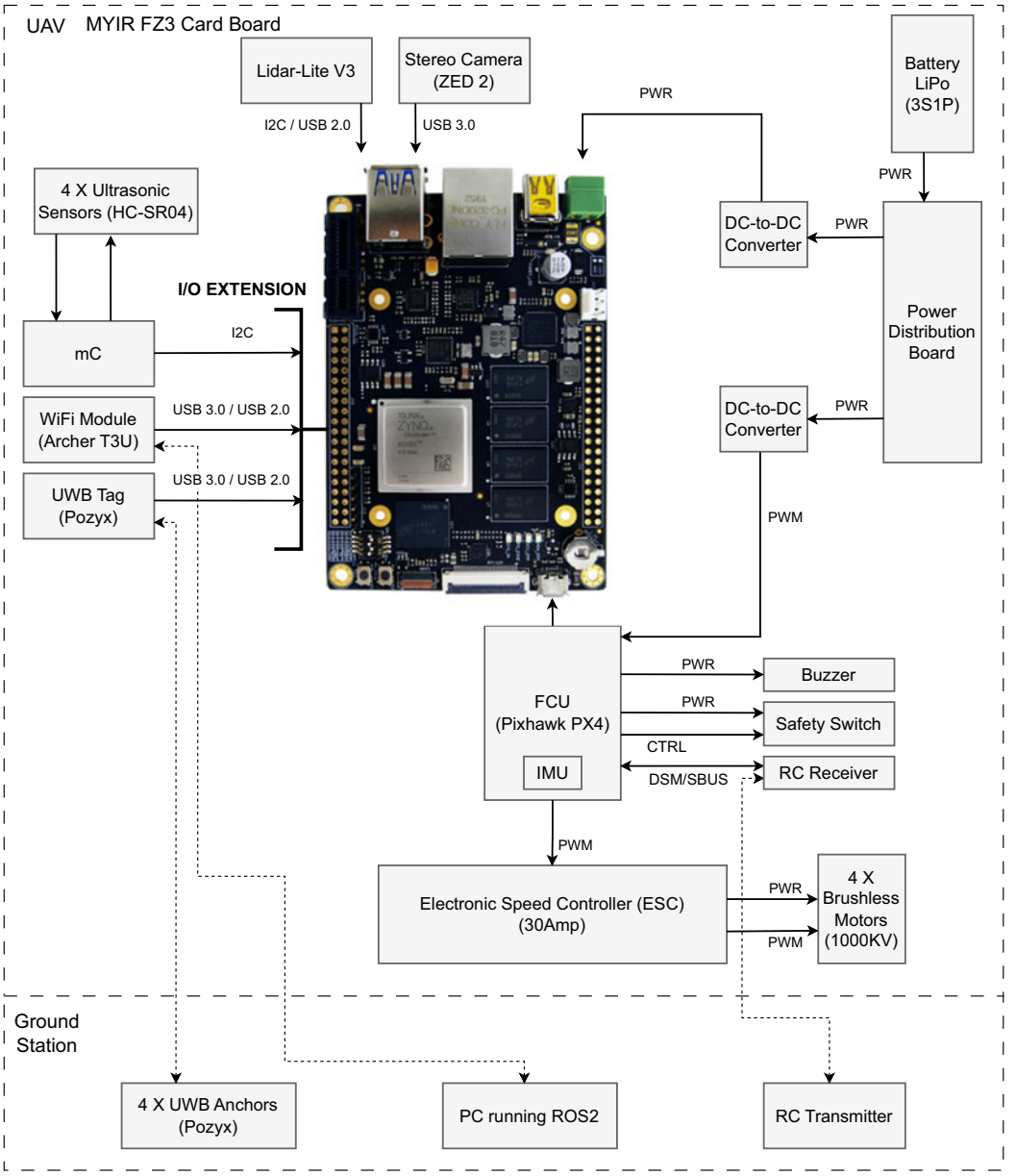

Figure 1. *Quadrotor system architecture.*

ground control station. The overall architecture is designed for modularity, offering flexibility to adjust configurations for different scenarios and experiments.

3.2 Coordinate systems for UAV positioning

To accurately determine the position of the quadrotor in space and the relative positions of surrounding objects, two coordinate frames are defined using the standard right-handed robotics convention, as illustrated in Fig. 2. The Earth's inertial frame $\{E\}$ adopts the East-North-Up (ENU) reference system, where the $+x$ axis points east, $+y$ points north and $+z$ points upwards in accordance with the right-hand rule. The body frame of the quadrotor $\{B\}$ is coincident with the origin and thus represents the quadrotor's

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

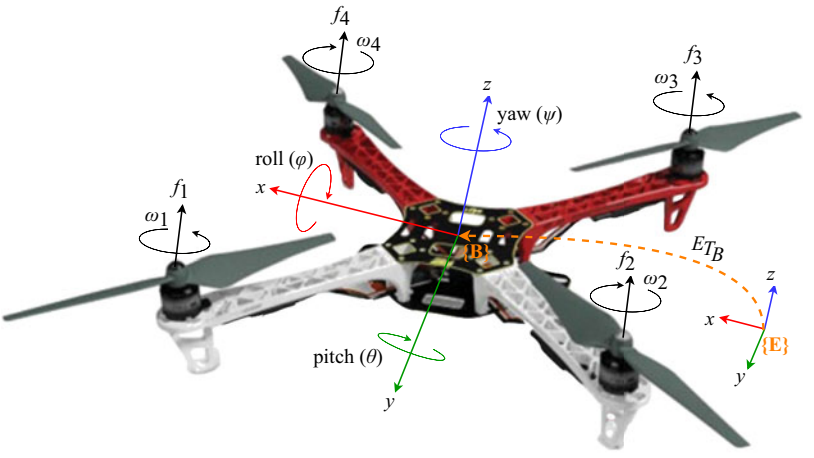

Figure 2. *F450 quadrotor model with coordinate frame.*

absolute position, $[x, y, z]$. It adheres to the Forward-Left-Up (FLU) convention, which defines forward horizontal movement along the $+x$ axis, leftward horizontal movement along the $+y$ axis and upward vertical movement along the $+z$ axis.

3.3 Quadrotor dynamics

The quadrotor is an inherently unstable, non-linear and complex system comprising four rotors, as indicated by its name. Each rotor includes a propeller and a motor that generates an angular velocity ω_i, producing a thrust force f_i, where i denotes the motor number, as illustrated in Fig. 2. Two rotors rotate clockwise, while the other two rotate counterclockwise, counteracting any undesired rotation of the quadrotor body in the yaw (ψ) direction due to the conservation of angular momentum.

The angular velocities of the rotors govern specific rotational coordinates, $\eta = (\phi, \theta, \psi) \in \mathbb{R}^3$, which in turn control the quadrotor's motion in the translational coordinates, $\xi = (x, y, z) \in \mathbb{R}^3$, relative to the Earth's inertial frame $\{E\}$. The quadrotor's orientation is defined by the Euler angles ϕ, θ and ψ, where ϕ (roll) is the angle about the x-axis, θ (pitch) is the angle about the y-axis, and ψ (yaw) is the angle about the z-axis. The translational coordinates x, y and z represent the centre of mass of the quadrotor with respect to the Earth's inertial frame.

The quadrotor's translational equations of motion in the Earth's frame are derived using the Newton–Euler formalism [26, 27], while the rotational equations are derived using the same formalism [28]. These equations assume relatively small quadrotor movement angles:

$$\begin{aligned}
\ddot{x} &= \frac{f_T}{m}\left(c\phi s\theta c\psi + s\psi s\phi\right), \quad & \ddot{\phi} &= \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} - \frac{J\dot{\theta}\Omega_i}{I_x} + \frac{\tau_x}{I_x}, \\
\ddot{y} &= \frac{f_T}{m}\left(c\phi s\theta s\psi - c\psi s\phi\right), \quad & \ddot{\theta} &= \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \frac{J\dot{\phi}\Omega_i}{I_y} + \frac{\tau_y}{I_y}, \\
\ddot{z} &= \frac{f_T}{m}\left(c\phi c\theta\right) - g, \quad & \ddot{\psi} &= \frac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \frac{\tau_z}{I_z}.
\end{aligned} \tag{1}$$

Here, m represents the quadrotor's mass, f_T is the total thrust force and g is the gravitational acceleration. The torques τ_x, τ_y and τ_z correspond to the roll, pitch and yaw torques, respectively. J denotes the rotor inertia, while Ω_i represents the angular velocity of the i^{th} rotor [29, 30]. Additionally, I_x, I_y and I_z are the moments of inertia of the quadrotor's symmetric rigid body about its three principal axes.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

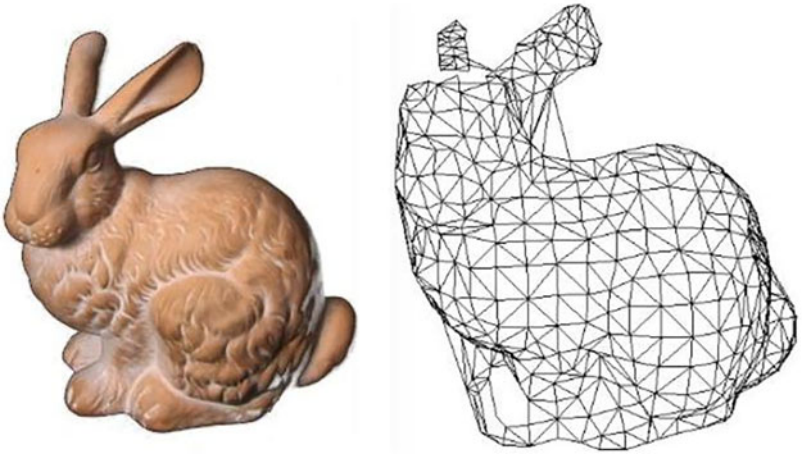

Figure 3. *The Stanford Bunny model (left) has been reconstructed (right, comprising 397 points). For improved viewing clarity, the DT figure on the right has been post-processed and does not depict the convex hull of the surface points.*

A simplified linearised model is obtained around the hovering equilibrium point via small-angle approximation (i.e. since we assume relatively low navigation speed, the rotational angles of the system are small also). This approximation leads to $\dot{\phi}, \dot{\theta}, \dot{\psi} \simeq 0$, and $s\phi \simeq \phi, s\theta \simeq \theta, s\psi \simeq \psi$, with $c\phi = 1, c\theta = 1$ and $c\psi = 1$. The system state vector is defined as $\mathbf{x} = [xyz\phi\theta\psi\dot{x}\dot{y}\dot{z}\dot{\phi}\dot{\theta}\dot{\psi}]^T$. The hovering equilibrium point, $\bar{\mathbf{x}} = [\bar{x}\bar{y}\bar{z}000000000]^T$ is reached when the total thrust corresponds to a constant control input of $f_T = mg$, providing the force necessary to sustain the quadrotor's hover at an arbitrary position $(\bar{x}, \bar{y}, \bar{z})$. The linearised system derived from this analysis is expressed as:

$$\ddot{x} = g\theta, \quad \ddot{y} = -g\phi, \quad \ddot{z} = \frac{f_T}{m},$$
$$\ddot{\phi} = \frac{\tau_x}{I_x}, \quad \ddot{\theta} = \frac{\tau_y}{I_y}, \quad \ddot{\psi} = \frac{\tau_z}{I_z}. \tag{2}$$

4.0 Delaunay Triangulation Core Implementation For Surface Reconstruction

Surface reconstruction is a fundamental computational technique with extensive applications across numerous fields. It involves sampling an object's surface and applying algorithms to reconstruct its structure (e.g. Fig. 3). This work concentrates on surface reconstruction for surfaces representable in the form $z = f(x, y)$, commonly referred to as a terrain [31]. Such surfaces are frequently utilised in digital elevation models but also appear in various other disciplines, as they can represent local charts of manifolds – sufficiently small patches that can be appropriately 'glued' together to form a larger surface embedded in $\mathbb{R}^3$.

4.1 Delaunay triangulation algorithm

This work utilises the Delaunay triangulation algorithm for surface reconstruction. In any triangulation, a triangle's circumcircle must not contain any other points from the set within its boundary; the interior of the circumcircle should include only the triangle's vertices. The incremental algorithm offers a step-by-step approach to constructing Delaunay triangulations. Starting from an initial triangulation, it adds new points sequentially, updating the triangulation with each insertion to maintain the Delaunay property.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

A critical step in this method involves subdividing affected triangles when a new point is introduced, followed by edge flipping to ensure compliance with the Delaunay condition. Each point insertion results in localised adjustments, preserving the Delaunay property throughout the triangulation process.

To enable efficient real-time surface reconstruction, the algorithm is implemented on an FPGA via the DT-core (Delaunay triangulation core).

The process commences by projecting all sampled points onto the x-y plane while retaining their original height values. A Delaunay triangulation is subsequently applied to the projected points, segmenting the surface into multiple triangles. The points within the triangulation are then elevated to their recorded heights, creating a piecewise linear representation of the surface using these triangles. An increase in the number of points results in the formation of more triangles, thereby improving the quality of the surface reconstruction.

4.1.1 Incremental algorithm

To reconstruct the surface, the incremental algorithm is employed. This algorithm begins by creating an initial super-triangle that encompasses all points in the dataset. For each new point inserted, the triangle containing the point is identified and subdivided into three new triangles based on the newly added point. Subsequently, the circumcircles of these new triangles are examined to ensure they do not enclose any other points from the dataset. If the Delaunay condition is violated, edge-flipping is performed to restore compliance. This process continues for each new point until all triangles satisfy the Delaunay condition. The algorithm has a time complexity of $O(n\log n)$, where n is the number of points in the dataset.

The Delaunay triangulation is detailed in Algorithm 4.1.1. In line 1, a triangle is constructed to encompass all the points on the surface being reconstructed. Lines 2–3 involve processing each point from the set P, iterating over all triangles to find the one containing the current point. If such a triangle is found, it is subdivided into three new triangles in line 5, with their vertices formed by the newly inserted point. The set of triangles is then updated accordingly in line 6. The new triangles are marked as unprocessed for further investigation in line 7.

Lines 8–14 of the algorithm are dedicated to verifying the Delaunay condition for all unprocessed faces. If a face fails to satisfy the condition (line 9), the flipping procedure is applied in line 10 to transform it into a valid Delaunay triangle. The updated faces are marked for further investigation in line 11, while the edges and triangles are updated in lines 12–13. This iterative process continues until all triangles comply with the Delaunay condition. Finally, line 15 outlines the return of $D(P)$, the edges of the completed Delaunay triangulation [16].

4.2 On the use of two-dimensional delaunay triangulation for mapping and navigation

The Delaunay triangulation (DT) approach adopted in this work operates in a two-dimensional domain, where triangulation is performed using only the planar (x, y) coordinates. Each triangulation vertex retains its own elevation value, which may vary across the surface and is used to reconstruct surfaces of the form $z = f(x, y)$.

This design choice is intentional and well aligned with typical indoor UAV navigation requirements. In many practical scenarios, motion planning and obstacle avoidance are carried out predominantly in the horizontal plane, while altitude is regulated independently by the flight control system. Under these conditions, a two-dimensional triangulated representation provides sufficient geometric information for mapping, free-space estimation and collision avoidance, while significantly reducing computational complexity.

Compared to full three-dimensional Delaunay triangulation, the 2D formulation offers advantages in terms of determinism, memory footprint and suitability for real-time implementation on resource-constrained SoC FPGA platforms. Although full 3D triangulation may be required for volumetric mapping or multi-level environments, such functionality lies beyond the scope of this work. The proposed DT core nevertheless provides a scalable foundation that can be extended towards higher-dimensional triangulation approaches in future research.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

Algorithm 1 Incremental Delaunay Triangulation Algorithm

Data: P: set of points, T: set of legal triangles
Result: $D\,(P)$: edges of the Delaunay triangulation

Initialise T with a sufficiently large triangle t
for each $x \in P$ **do**
  for each $t \in T$ **do**
    If $t \ni x$ **then**
      $T_{\text{sub}} \leftarrow \{t_1, t_2, t_3\}$ ▷ Subdivide t into t_1, t_2, t_3
      $T \leftarrow (T \setminus \{t\}) \cup T_{\text{sub}}$ ▷ Remove t and update T
      $F_{\text{u}} \leftarrow F_{\text{u}} \cup T_{\text{sub}}$ ▷ Mark newly created faces as unprocessed
    end if
  end for
  for each $f \in F_{\text{u}}$ **do**
    if f is illegal **then** ▷ Does not satisfy the Delaunay condition
      FlipEdge (f) ▷ Flip the illegal edge
      $F_{\text{u}} \leftarrow F_{\text{u}} \cup$ UpdatedFaces (f) ▷ Mark all updated faces as unprocessed
      $D\,(P) \leftarrow D\,(P) \cup$ Edges (f) ▷ Update $D\,(P)$
      $T \leftarrow T \cup$ Triangles (f) ▷ Update T
    end if
  end for
end for
return $D\,(P)$ Final edges of the Delaunay triangulation

4.3 Delaunay triangulation core high-level architecture

The DT algorithm was implemented on an FZ3 accelerator card, which is built around the AMD Zynq UltraScale+ ZU3EG MPSoC. This platform integrates a SoC that combines a processing system (PS) – featuring a 1.2 GHz Quad-Core ARM Cortex-A53 and a 600 MHz Dual-Core Cortex-R5 – with PL, where the DT core was deployed. This architecture enables real-time surface reconstruction.

4.3.1 Hardware platform: FZ3 accelerator card

This work utilised the FZ3 accelerator card as the onboard computer, whose technical specifications are summarised in Table 1.

The onboard computer leverages the hardware capabilities of the AMD Zynq UltraScale+ MPSoC FZ3 accelerator card [32]. This platform is a general-purpose board that integrates a heterogeneous computing system consisting of a PS and PL, which communicate via a 128-bit AMBA AXI4 interconnect.

Processing system (PS)

The PS integrates several heterogeneous processing units:

- **Cortex-A53 application processing unit (APU):** a 64-bit quad-core Arm v8 CPU with an 8-stage, 2-way superscalar in-order pipeline. Each core includes separate 32 KB L1 instruction and data caches, a shared 1 MB L2 cache and a memory management unit (MMU).
- **Cortex-R5 real-time processing unit (RPU):** a 32-bit dual-core processor (Arm v7 architecture) with 32 KB of L1 instruction and data caches per core. Each core also has a local 128 KB tightly coupled memory (TCM). The cores can operate independently or in lockstep mode and support error correction code (ECC), enabling detection of up to two errors and correction of one.
- **Mali-400 MP2 graphics processing unit (GPU):** includes pixel and geometry processors with a 64 KB L2 cache.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

Table 1. *Features of the FZ3 accelerator card*

Feature	Description
EMCPU	AMD Zynq Ultrascale+ XCZU3EG-1SFVC784E (ZU3EG, 784-pin package) MPSOC - 1.2 GHz 64-bit quad-core Arm® Cortex ™-A53 - 600 MHz dual-core Arm® Cortex ™-R5 processor - Arm Mali ™-400 MP2 graphics processor - 16 nm FINFET + FPGA fabric
RAM	4 GB DDR4 (64-bit)
Flash	8 GB EMMC, 32 MB QSPI, 32 KB EEPROM

Programmable logic (PL)

The PL section provides significant reconfigurable computing resources for hardware acceleration and sensor interfacing, as summarised below:

- Programmable logic cells: 154 K
- Look-up tables (LUTs): 71 K
- Flip-flops: 141 K
- Distributed RAM: 1.8 Mb
- Block RAM: 7.6 Mb
- DSP slices: 360

A block-level overview of the SoC architecture and its subsystems is illustrated in Fig. 4.

4.3.2 Implementation in programmable logic

The DT core is implemented on the PL side to enable point cloud surface reconstruction using point cloud datasets (PCD) received from the stereo camera or LiDAR on the UAV. This approach leverages the programmability and performance of FPGA hardware to efficiently process the incoming PCD data for real-time reconstruction.

To develop the DT core, a high-level synthesis (HLS) process is employed. This process begins with a high-level description of the algorithm, written in languages such as C/C++, Python, or MATLAB. The code is then analysed, with architectural constraints and scheduling strategies applied to generate an optimised register-transfer level (RTL) HDL. Subsequently, the RTL is synthesised into gate-level circuits using logic synthesis tools, transforming the high-level algorithm into efficient hardware logic.

For high-level synthesis, Vitis HLS was used as the synthesis tool, packaging the resulting RTL into an intellectual property (IP) core on an AXI4 bus. This process is illustrated in Fig. 5.

Compared to the original software implementation, this design significantly improves surface reconstruction speed, with the IP core demonstrating substantial performance gains over the software-based approach, as shown in Fig. 12 [16].

For efficient point handling, an incremental insertion process was developed, controlled by the 'tri2D_loop1' function, which calls the 'insert_site' function in each iteration. The 'insert_site' function runs in parallel with the loop and manages the insertion of each surface point. The number of iterations in the loop depends on the total number of surface points, directly influencing the loop's latency cycles. Within the 'insert_site' function, the 'insert_site_loop1' loop is responsible for locating the legal triangle (stored in T) that contains the current point. To verify whether the identified triangle is valid, the loop invokes the 'in_triangle' function. If the required triangle is difficult to locate, the number of iterations in the loop may increase; however, the maximum number of iterations is governed by the

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

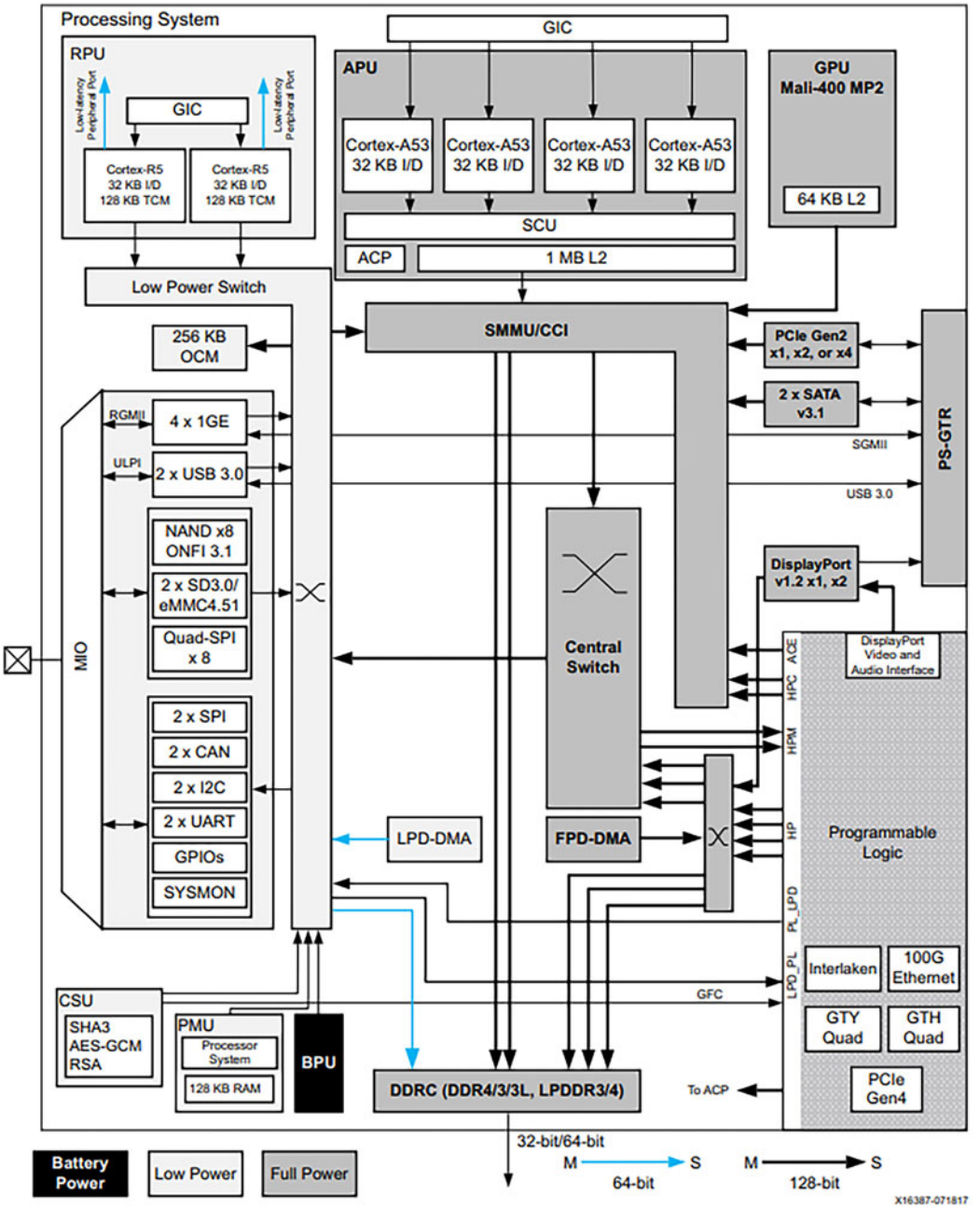

Figure 4. *Block-level overview of the Zynq UltraScale+ SoC architecture and its subsystems. Adapted from Ref. (32).*

equation $2n - 2 - k$, which represents the maximum number of triangles that can be constructed during the Delaunay triangulation based on the number of points n.

After the legal triangle is found, the 'insert_site' function calls the 'legalise_edge' function to verify the legality of each new edge. During this step, 'initialisation_loop2' initialises the table containing triangles to be checked. The 'legalise_edge_loop1' loop is responsible for checking all affected triangles until

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

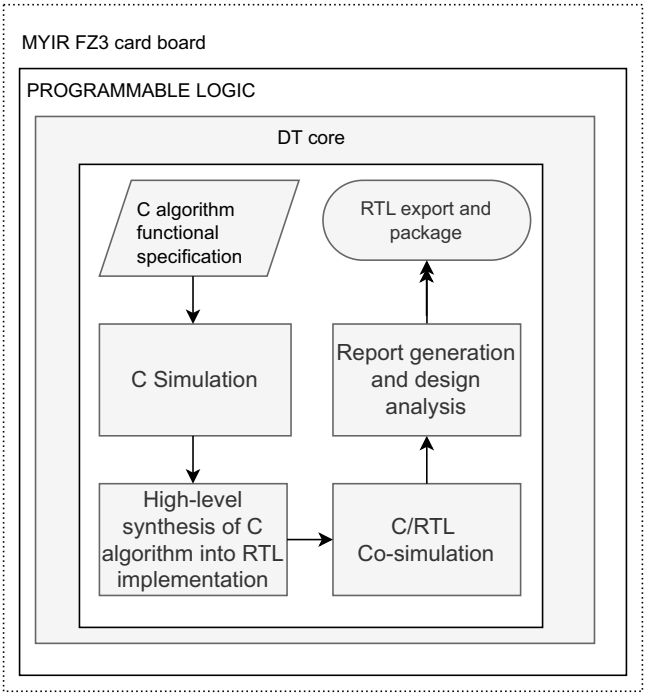

Figure 5. *DT core implementation on PL side.*

the verification process is complete. Ultimately, the triangulation results are output as 'edge_matrix'. Figure 6 illustrates the overall DT-core architecture.

4.3.3 Real-time data exchange with ROS 2 in the processing system

In the PS, the surface reconstruction and implementation framework utilises the ROS 2 communication architecture to enable real-time data exchange between system components. This section outlines the ROS 2-based communication system and its role in managing data flow across various modules, including sensors and processing units. This configuration is critical for efficient point cloud processing, which serves as the foundation for the UAV's environmental perception and navigation capabilities.

The ROS is an open-source middleware extensively used in robotics. However, ROS 1 is not well-suited for real-time embedded systems due to its inability to meet real-time requirements and its limited operating system compatibility. To address these limitations, ROS 1 has been significantly updated to ROS 2, which leverages the data distribution service (DDS). DDS is better suited for real-time distributed embedded systems due to its various transport configurations, such as deadline management, fault tolerance and scalability.

ROS 2 allows a primary programme to be divided into multiple subprocesses that operate concurrently and interact with each other. In ROS 2, nodes represent processes, each assigned a distinct task to enhance overall system efficiency. Communication among these nodes occurs through a series of topics and services. Topics facilitate the transfer of information, known as messages, from one node to another. Nodes publish and/or subscribe to topics to send and/or receive these messages. Services operate on the request/response model: one node offers a service that another node invokes using the service name. The node creating the service is the server, while the nodes calling the service are the clients. Topics are suitable for managing continuous data streams, while services are ideal for blocking calls, making them suitable for remote procedure calls that terminate quickly.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

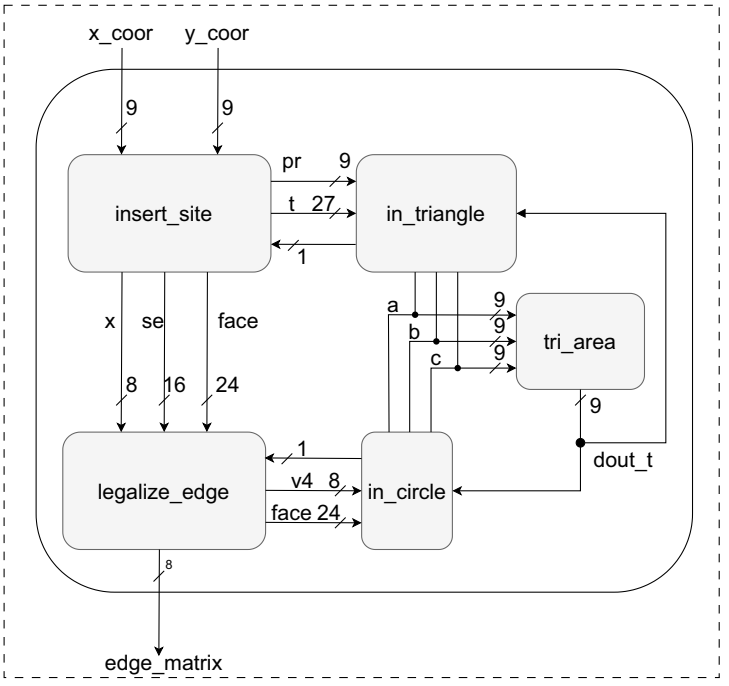

Figure 6. *DT core architecture.*

The uXRCE-DDS middleware is employed within PX4 to enable the publication and subscription of uORB messages on the FZ3 card, mimicking ROS 2 topics. This facilitates swift and dependable integration between PX4 and ROS 2, simplifying the process for ROS 2 applications to access UAV information and issue commands. PX4 utilises an XRCE-DDS implementation that takes advantage of eProsima Micro XRCE-DDS [33]. The uXRCE-DDS middleware comprises a client operating on PX4 and an agent functioning on the FZ3 card. These components facilitate bidirectional data exchange through a serial or UDP connection. Acting as a proxy for the client, the agent allows it to both publish and subscribe to topics within the global DDS data space. For PX4 uORB topics to be accessible on the DDS network, the uXRCE-DDS client on PX4 must connect with the micro XRCE-DDS agent on the FZ3 card. The PX4 UXRCE_DDS_CLIENT manages the publication and subscription of designated uORB topics within the global DDS data space. On the FZ3 card, the eProsima micro XRCE-DDS agent serves as a proxy for the client within the DDS/ROS 2 network.

A critical aspect of the system implementation is the interface between the PX4 flight controller and the onboard computer realised on the FZ3 card, enabling the UAV to operate safely indoors. Serial communication is employed for this interface via the TX and RX lines between the onboard computer and the PX4, connected through the TELEM2 port. The PX4 runs firmware version 1.15 to ensure proper interfacing, incorporating the uXRCE-DDS middleware on the PX4 side. The same uXRCE-DDS middleware is executed on the FZ3 card, integrated with the ROS 2 Humble release. To establish this interface, a PetaLinux project was developed, incorporating the hardware implementation, the ROS 2 environmentand a bootable image that includes an Ubuntu root file system.

The communication architecture is based on ROS 2 between the UAV modules (onboard computer, FCU and microcontroller (μC)) and the ground station (computer and RC controller). The IMU, UWB, LiDAR, stereo camera and ESC nodes send (i.e. publish) and receive (i.e. subscribe) data via several topics, while the buzzer and safety switch nodes send and receive data via services. Figure 7 presents the implementation on the PS and the interface with the PL through the AMBA Interconnect/AXI4 protocol [34].

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

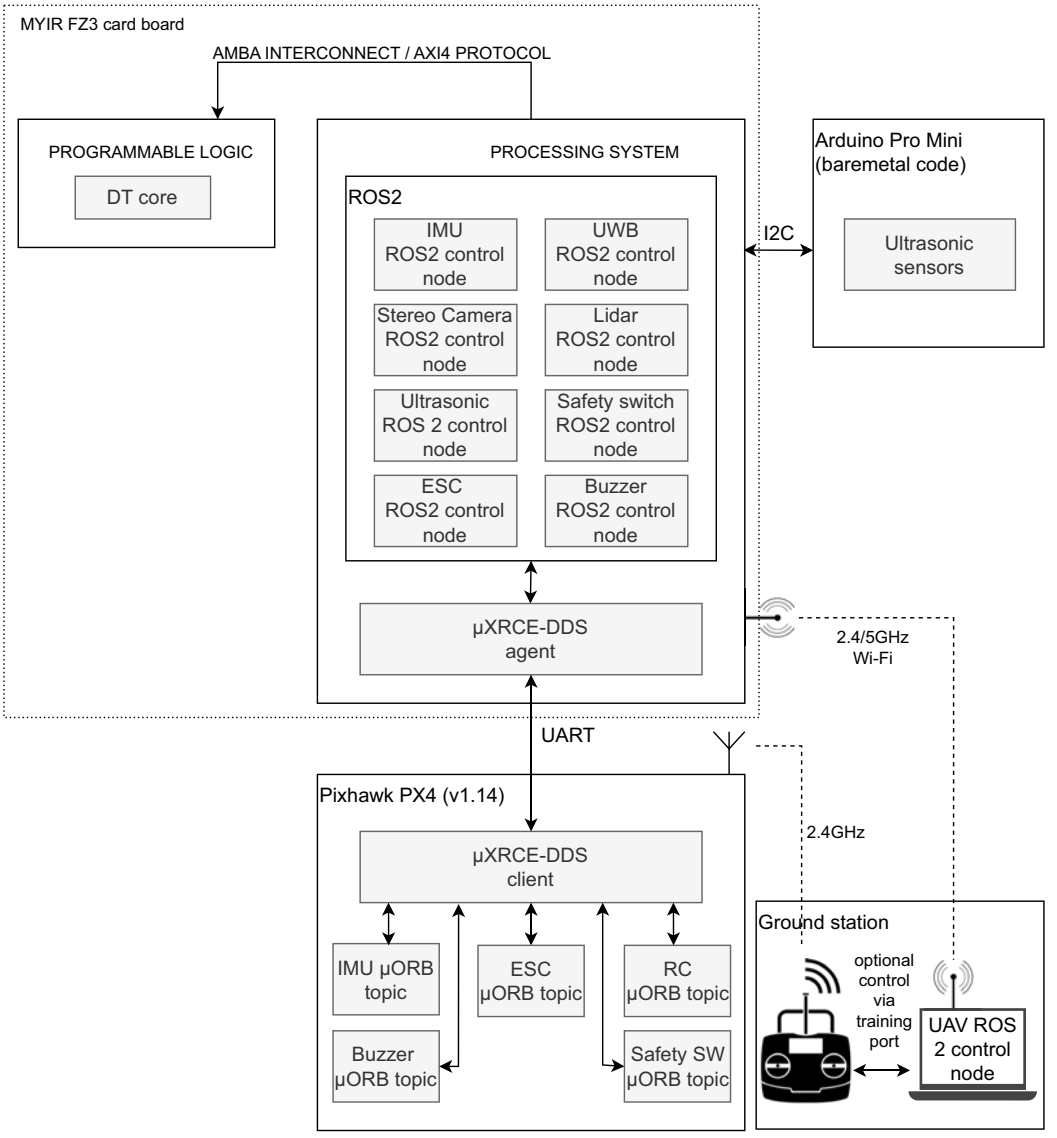

Figure 7. *ROS 2-based communication architecture.*

4.4 PetaLinux and Vivado workflow process design

To build the system framework, two distinct processes are followed: the PetaLinux workflow and the Vivado workflow, as illustrated in Fig. 8 [35, 36]. PetaLinux operates in the PS of the Zynq SoC, enabling the creation of customised operating systems for SoC designs. Vivado, a design suite from AMD for adaptive FPGAs and SoCs, is employed for design entry, synthesis, place-and-route and verification/simulation of HDL designs implemented in the PL. Vivado provides comprehensive tools that facilitate the entire design process, including design entry, synthesis, place-and-route and verification/simulation [35, 37].

4.5 Indoor navigation and environment reconstruction

The indoor navigation framework of the proposed UAV platform relies on a sequential sensing-to-mapping pipeline that enables real-time perception and decision-making in GPS-denied environments.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

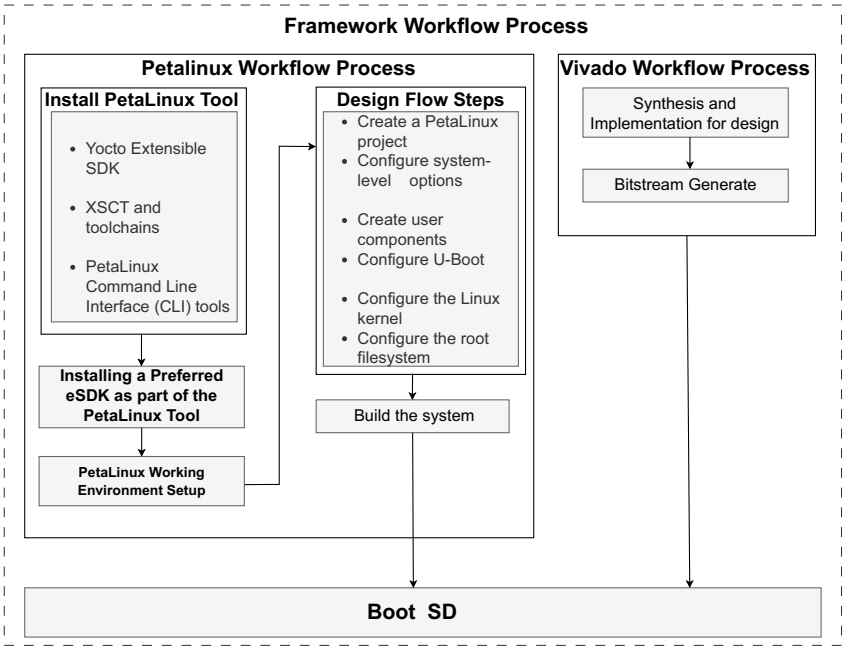

Figure 8. *Framework workflow process.*

The process begins with data acquisition from multiple onboard sensors, including the LiDAR, stereo camera, UWB, IMU and ultrasonic range sensors. The LiDAR and stereo camera provide dense spatial information that is fused into a three-dimensional point-cloud representation of the surrounding environment, while the UWB and IMU modules supply localisation and attitude data to maintain a consistent reference frame.

The generated point cloud is transmitted to the FZ3 accelerator card, where the FPGA-based DT core performs incremental triangulation of the incoming data. Each new set of sampled points is processed in real-time, producing a topologically consistent triangular mesh. This mesh converts the unstructured point cloud into a continuous surface representation, effectively reconstructing the surrounding indoor environment. By updating the triangulation incrementally as new sensor data arrive, the system maintains a dynamic geometric model of the UAV's immediate surroundings.

The reconstructed surface serves as the foundation for environment modelling. The triangulated mesh delineates free-space and obstacle regions based on geometric features such as height gradients and surface normal vectors. These features are subsequently utilised by the navigation and control modules to perform path planning and obstacle avoidance. The onboard ROS 2 nodes interpret the reconstructed model to determine safe flight corridors, compute feasible trajectories and issue corresponding motion commands to the flight controller.

This continuous loop, spanning from sensor acquisition through Delaunay-based surface reconstruction to navigation control, enables the UAV to achieve accurate, robust and real-time indoor navigation without external positioning systems.

5.0 Results and Analysis

This section presents the findings of the quadrotor project, focusing on key performance metrics. The Delaunay triangulation core simulation and performance subsection demonstrates the simulation and performance results of the implemented core for various PCD sizes and how these compare to the

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

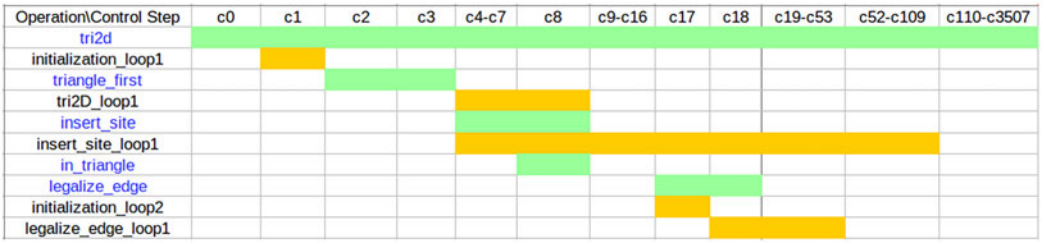

Figure 9. *FPGA design module performance: latencies in terms of clock cycles and time (ns).*

software counterpart implementation. The mass analysis subsection examines the overall mass distribution and its impact on the quadrotor's flight dynamics. The power estimation subsection evaluates the system's power consumption, providing insights into the UAV's efficiency and endurance.

5.1 Delaunay triangulation core simulation and performance

The following FPGA implementation and performance evaluation of the DT core build upon the authors' previous work [16].

The primary objective of the hardware implementation is performance optimisation, with particular emphasis on reducing latency cycles. Vitis HLS was used to translate the DT algorithm described in Algorithm 4.1.1 into RTL logic, enabling a detailed analysis of the design's timing behaviour. Following synthesis, the latency associated with each function and loop was identified, and an in-depth analysis of the cycle distribution was conducted. Figure 9 illustrates the main functions and loops contributing to the overall execution latency.

Among the key functions, TRI2D serves as the top-level controller, managing all inputs (such as X_COOR and Y_COOR) and producing the final triangulation output (EDGE_MATRIX). As the top-level function, tri2d encapsulates the complete execution latency of the design, and its internal processes therefore have a dominant impact on overall performance.

The triangulation process begins with an initialisation phase implemented in INITIALISATION_LOOP1, which prepares table T for storing the set of legal triangles. This is followed by the execution of the TRIANGLE_FIRST function, which inserts an initial enclosing triangle that contains all surface points. This operation is performed only once and establishes the initial triangulation domain, after which auxiliary edges associated with the enclosing triangle are removed.

Following initialisation, the TRI2D_LOOP1 iteratively inserts surface points into the triangulation. During each iteration, the INSERT_SITE function processes a single point and updates the triangulation accordingly. Within this function, the INSERT_SITE_LOOP1 searches table T to identify the triangle that contains the current point. This search relies on repeated calls to the IN_TRIANGLE function and constitutes a significant portion of the overall latency, as its iteration count depends on the number of triangles present in the triangulation.

Once the containing triangle has been identified and subdivided, the LEGALIZE_EDGE function is invoked to restore the Delaunay property. An initialisation step (INITIALIZATION_LOOP2) prepares the set of affected triangles, which are then processed by LEGALIZE_EDGE_LOOP1. This loop iteratively examines and updates neighbouring triangles through edge-flipping operations until all affected edges satisfy the Delaunay criterion. Upon completion of this process for all inserted points, the final triangulation is produced in the form of the EDGE_MATRIX output.

More information about the iterations of each loop or the total number of latency cycles for every function and loop of the design are presented in detail in Table 2. Furthermore, the table presents the optimisation with the directives we used in order to achieve more reduction in the clock cycles of the design.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

Table 2. *Latency analysis of the DT core architecture*

Operation and control step	Latency (cycles)	Iteration latency (cycles)	Trip count	Pipelined	Unroll
tri2d	1,393,079	—	—	—	—
Initialization_loop1	800	1	800	no	—
Triangle_first	7	—	—	—	—
Tri2D_loop	1,392,279	3,507	397	no	—
Insert_site	3,507	—	—	—	—
Insert_site_loop1	1,680	106	16	yes	50
In_triangle	4	5	—	—	—
Insert_site_loop2	602	602	1	—	3 (full)
Legalize_edge	602	—	—	—	—
Initialization_loop2	5	1	6	no	—
Legalize_edge_loop	595	35	17	no	—

Figure 10 shows a waveform snapshot captured from Siemens' (formerly Mentor Graphics) ModelSim digital simulation tool during the HDL project co-simulation process. The HDL simulator processes simulation requests (e.g. an input PCD file) generated by the C++ testbench. The PCD file, which represents point cloud data captured from the UAV's stereo camera or LiDAR sensor, provides the input for simulation. The core's inputs and outputs, which accept an array of PCD points (*x_xcoor*, *y_coor*) and produce the computed Delaunay triangulation edges matrix (*edge_matrix*), are implemented using the Vitis_HLS tool with the *ap_memory* port interface. This interface facilitates communication with RAM blocks, particularly when implementations, such as the 2D Delaunay core, require random access to memory addresses. The control signal *ap_start* ensures computation does not begin until it is asserted (high), while the signals *ap_done* and *ap_idle* indicate the completion of computation and an idle state, respectively. Due to the pipelined nature of the design, the output signal *ap_ready* signifies when the core is prepared to accept new inputs.

The hardware implementation was evaluated using a standard benchmark, the Stanford Bunny model, with varying point densities (477, 953, 1906 and 3811 points), as depicted in Fig. 11. The results of the comparison between the software and FPGA core are shown in Fig. 12. The evaluation demonstrated a substantial speed-up over the software implementation, achieving a 20× improvement in execution time. The FPGA implementation successfully managed real-time surface reconstruction, attaining a frame rate of 70 FPS for a point cloud consisting of 477 points.

Compared to the original software implementation, this design significantly improves surface reconstruction speed, with the IP core demonstrating substantial performance gains over the software-based approach.

Table 3 illustrates the comparison between the software and hardware implementations, showing both the execution time and the speed-up achieved through hardware acceleration. These results confirm the suitability of the design for real-time applications.

5.2 Mass analysis

The total takeoff mass (m_{TO}) of the electric multirotor UAV is represented as [38]:

$$m_{TO} = m_0 + m_b + m_{pl} \tag{3}$$

Where m_0 is the mass of the frame, avionics and propulsion system (excluding the battery), m_b is the battery mass and m_{pl} is the payload mass, including the onboard computer, camera, LiDAR, ultrasonic sensors and UWB tag.

Equation (3) was applied during the design stage to balance component selection, payload capacity and battery sizing, ensuring the total takeoff mass remained within the propulsion system's thrust limits.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

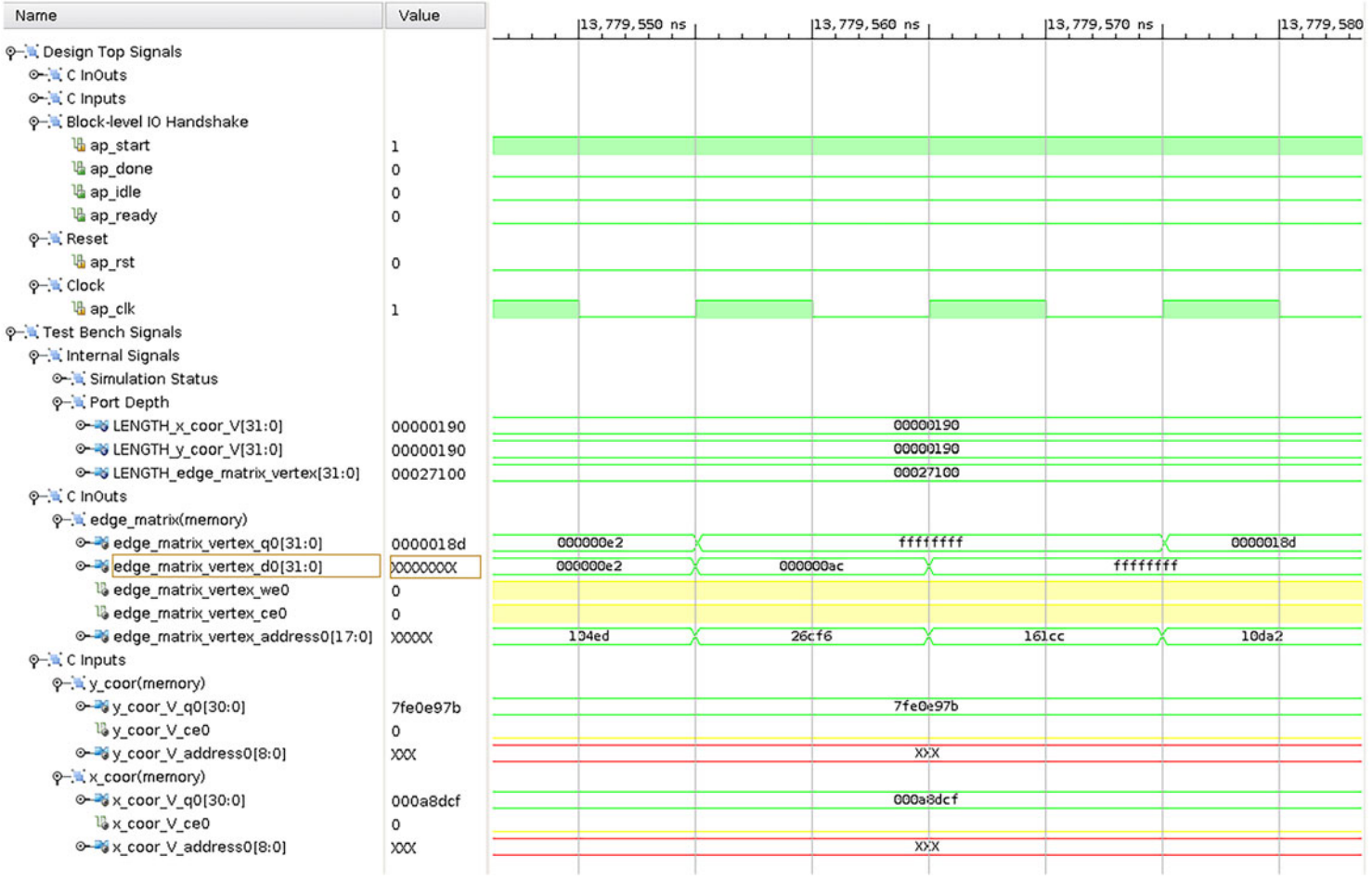

Figure 10. *HDL model co-simulation.*

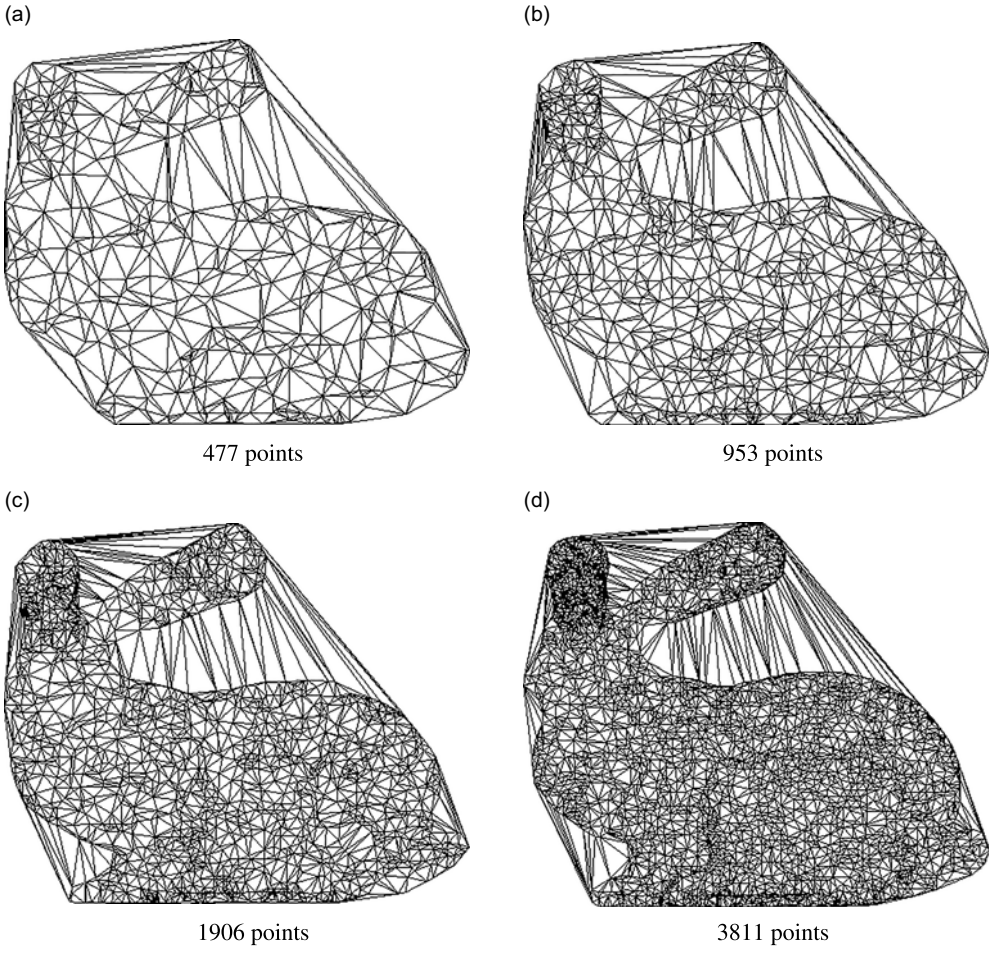

Figure 11. *Stanford Bunny, sampled at different numbers of points.*

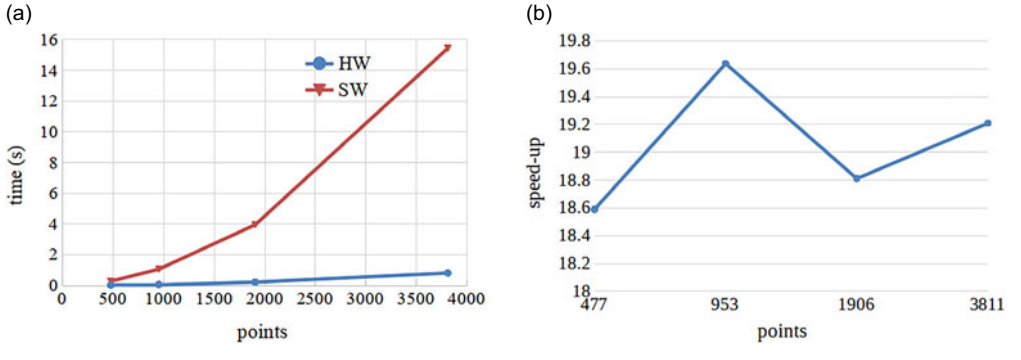

Figure 12. *Comparison between software and FPGA core.*

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

Table 3. *Performance results (time in ms)*

Number of points	Software (ms)	Hardware (ms)	Speed-up
477	268.55	14.45	18.5894
953	1056.49	53.8	19.6381
1906	3951.38	210.05	18.8113
3811	15448.8	804.26	19.2087

Table 4. *Mass breakdown of the various UAV platform components*

Quantity	Component description	Mass (g)
1	F450 frame	270
4	Propeller	56
4	Brushless DC motor (1000KV)	208
4	Electronic speed controller	92
1	14.8V 4400mAh 4S1P LiPo battery	453
1	Pixhawk P4 flight controller	16
1	FZ3 accelerator card	52
2	DC-to-DC converter	10
1	RC receiver	6
4	Ultrasonic sensors	34
1	Arduino Pro mini board	2
1	UWB Pozyx tag	42
1	ZED 2 stereo camera	160
1	Hokuyo URG-04LX-UG01	138
1	**Total mass**	**1539**

Table 4 summarises the mass distribution of the final configuration. The resulting takeoff mass of the quadcopter is approximately 1.5 kg when equipped with the FZ3 card.

5.3 Power estimation

The power consumption of a BLDC motor depends on its electrical input and mechanical output, which are governed by the applied voltage, current, load torque and rotational speed. The electrical input power P_e (W) is defined as

$$P_e = VI, \tag{4}$$

while the corresponding mechanical output power P_m (W) is given by

$$P_m = T\,\omega = \frac{T\,N}{9.55}, \tag{5}$$

where T is the torque (N m), ω is the angular velocity (rad s $^{-1}$), N is the motor speed in revolutions per minute (RPM), and the constant $9.55 = 60/\,(2\pi)$ converts RPM to rad s $^{-1}$. The motor efficiency η_m is therefore expressed as

$$\eta_m = \frac{P_m}{P_e} = \frac{T\,N}{9.55\,VI}. \tag{6}$$

LiPo batteries are widely used in small UAVs owing to their high specific energy and high discharge capability [39, 40]. The available battery energy E_b (Wh), accounting for conversion losses and the allowable depth of discharge, is defined as

$$E_b = E_{\text{spec}}\,m_b\,\eta_b\,f_{\text{DOD}}, \tag{7}$$

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

Table 5. *Expected maximum power draw of the UAV components*

Component	Voltage (V)	Current (A)	Power (W)
BLDC (1000KV)	11.1	13	144.3
Pixhawk PX4 (USB) incl.			
Safety switch and buzzer	5	0.18	0.9
FZ3 accelerator card	12	0.84	10
RC receiver	5	0.03	0.15
Ultrasonic sensor	5	0.006	0.03
Arduino Pro mini	5	0.016	0.08
LiDAR	5	0.5	2.5
ZED 2 stereo camera	5	0.38	1.9

where E_{spec} is the battery specific energy (Wh kg $^{-1}$), m_b is the battery mass (kg), $\eta_b \in (0, 1]$ represents the overall battery efficiency (including wiring, regulator and thermal losses), and $f_{\text{DOD}} \in (0, 1]$ is the usable depth-of-discharge (DOD) fraction, typically limited to 0.8 to prolong battery lifespan. The specific energy $_{\text{spec}}$ is determined from Equation (10) using the nominal battery capacity C (Ah) and voltage v (V) provided by the manufacturer.

The electrical power required for hovering, P_{h_e}, is estimated using momentum theory [41] and a propeller figure of merit η_p, typically ranging between 0.5 and 0.7:

$$P_{h_e} = \frac{1}{\eta_m} \frac{W_{TO}^{3/2}}{\eta_p \sqrt{2\rho\pi N_r}\, r_{\text{prop}}}, \tag{8}$$

where W_{TO} is the take-off weight (N), ρ is the air density (kg m $^{-3}$), N_r is the number of rotors, and r_{prop} is the propeller radius (m). The term $1/\eta_m$ accounts for the combined motor and ESC efficiency, which is assumed to be 0.8 for the configuration considered in this study.

The estimated endurance E (h) of the UAV is then computed as

$$E = \frac{E_b}{P_{h_e}} = \frac{E_{\text{spec}}\, m_b\, \eta_b f_{\text{DOD}}}{P_{h_e}}. \tag{9}$$

This formulation provides a practical endurance estimate that incorporates both electrical and mechanical efficiencies, as well as operational battery limitations. For higher-fidelity predictions, Peukert-type corrections and discharge-rate-dependent efficiency factors may be introduced [42, 43].

$$E_{\text{spec}} = \frac{Cv}{m_b}. \tag{10}$$

Here, C is the nominal battery capacity, v is the nominal voltage, and m_b denotes the battery mass [38]. To extend battery lifespan, the depth of discharge is limited to 80% [42, 44].

Using Equations (7), (8) and (9), together with W_{TO} from Table 4, $\rho = 1.225\text{kg/m}^3$, $r_{\text{prop}} = 0.127\text{m}$, $\eta_p = 0.6$, $\eta_m = 0.8$, and $\eta_b = 95\%$, an initial hover endurance of approximately 8 minutes is obtained. More accurate endurance predictions could be achieved using Peukert-based battery models; however, such analysis lies beyond the scope of this study.

Table 5 summarises the power consumption of the onboard UAV components that directly affect flight endurance. Ground station elements, such as the RC transmitter, are therefore intentionally excluded. The Pozyx UWB tag is also omitted, as it is powered by a dedicated replaceable CR2450 battery (3 V, 620 mAh), for which the manufacturer reports an autonomy of up to five years. It is worth noting that powering the Pixhawk PX4 FCU via USB significantly reduces power consumption compared to supplying it from the main battery through the ESC, owing to the lower efficiency of the ESC's voltage regulator.

The complete multirotor UAV configuration is shown in Fig. 13.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

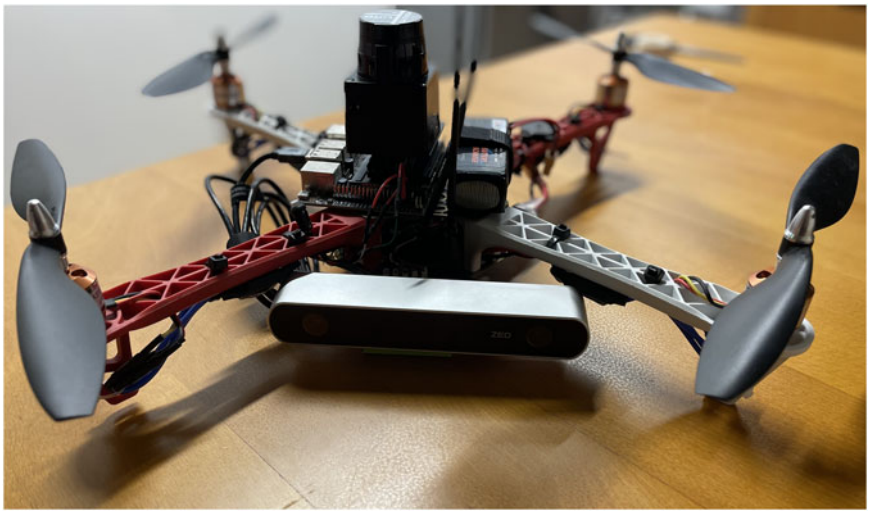

Figure 13. *Implementation of the multirotor UAV.*

6.0 Conclusions and Future Directions

This work presents a bespoke, low-cost and modular hardware platform designed to support research in indoor UAV navigation. Many existing approaches rely heavily on simulations that may not accurately represent real-world conditions. The proposed platform enables flexible and scalable experimentation, allowing researchers to adapt it to various indoor environments while maintaining appropriate payload constraints.

An incremental algorithm was implemented to compute Delaunay triangulations for real-time surface reconstruction and mapping. The onboard computer incorporates an FPGA-based DT core capable of achieving real-time triangulation at approximately 70 frames per second (FPS) for point clouds captured by the UAV's camera. Evaluation using the Stanford Bunny model with varying point densities (477–3811 points) demonstrated a 20× speed-up compared to the software implementation, confirming the FPGA's suitability for onboard, real-time processing. The UAV platform has a total mass of about 1.5 kg and an estimated endurance of approximately 8 minutes, illustrating its practicality for short-duration indoor experiments.

The FPGA-based deployment of the 2D Delaunay triangulation method, optimised through HLS based on the authors' previous work [16], enables efficient and real-time processing essential for UAV-based scanning and obstacle detection. Future research will focus on integrating a compact data structure and bin-sorting technique into the DT core to further enhance memory efficiency, processing speed, and scalability.

The proposed UAV platform aims to advance research through real-world testing in complex indoor environments. Ongoing work builds on the authors' previous research [17, 18], extending it to evaluate the stability and performance of filtering algorithms in realistic navigation scenarios.

Data availability statement. Data supporting the work in this paper can be made available from the corresponding author on reasonable request.

Financial support. The authors declare that no funds, grants or other support were received during the preparation of this manuscript.

Competing interests. The authors declare no competing interest.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

References

[1] Mohsan, S.A.H., Othman, N.Q.H., Li, Y., Alsharif, M.H. and Khan, M.A. Unmanned aerial vehicles (UAVS): practical aspects, applications, open challenges, security issues, and future trends, *Intell. Serv. Rob.*, 2023, **16**, pp 109–137.

[2] Chi, C., Zhan, X., Wang, S. and Zhai, Y. Enabling robust and accurate navigation for UAVS using real-time GNSS precise point positioning and IMU integration, *Aeronaut. J.*, 2021, **125**, pp 87–108.

[3] Kose, O. and Oktay, T. Simultaneous design of morphing hexarotor and autopilot system by using deep neural network and SPSA, *Aircr. Eng. Aerosp. Technol.*, 2023, **95**, pp 939–949.

[4] Arik, S. Redesign of morphing UAV for simultaneous improvement of directional stability and maximum lift/drag ratio, *Advanc. Electr. Comput. Eng.*, 2018, **18**, pp 57–62.

[5] Oktay, T., Arik, S., Turkmen, I., Uzun, M. and Celik, H. Neural network based redesign of morphing UAV for simultaneous improvement of roll stability and maximum lift/drag ratio, *Aircr Eng. Aerosp. Technol.*, 2018, **90**, pp 1203–1212.

[6] Oktay, T. and Köse, O. Hexarotor longitudinal flight control with deep neural network, PID algorithm and morphing, *European J. Sci. Technol.*, 2021.

[7] Pritzl, V., Vrba, M., štěpán, P. and Saska, M. Cooperative navigation and guidance of a micro-scale aerial vehicle by an accompanying UAV using 3d LIDAR relative localization, *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp 526–535.

[8] Telli, K., Kraa, O., Himeur, Y., Ouamane, A., Boumehraz, M., Atalla, S. and Mansoor, W. A comprehensive review of recent research trends on unmanned aerial vehicles (UAVS), *Systems*, 2023, **11**, p 400.

[9] Uzun, M. and Oktay, T. Simultaneous UAV having actively sweep angle morphing wing and flight control system design, *Aircr. Eng. Aerosp. Technol.*, 2023, **95**, pp 1062–1068. eprint: https://www.emerald.com/aeat/article-pdf/95/7/1062/111615/aeat-09-2022-0259.pdf

[10] Sandino, J., Vanegas, F., Maire, F., Caccetta, P., Sanderson, C. and Gonzalez, F. UAV framework for autonomous onboard navigation and people/object detection in cluttered indoor environments, *Remote Sens.*, 2020, **12**, p 3386.

[11] Elshakhs, Y.S., Deliparaschos, K.M., Charalambous, T., Oliva, G., and Zolotas, A., A comprehensive survey on delaunay triangulation: applications, algorithms, and implementations over CPUs, GPUs, and FPGAs, *IEEE Access*, 2024, **12**, pp 12562–12585. Conference Name: IEEE Access.

[12] Dancila, R. and Botez, R., New flight trajectory optimisation method using genetic algorithms, *Aeronaut. J.*, 2021, **125**, pp 618–671.

[13] Rowell, N., Dunstan, M.N., Parkes, S.M., Gil-Fernández, J., Huertas, I. and Salehi, S. Autonomous visual recognition of known surface landmarks for optical navigation around asteroids, *Aeronaut. J.*, 2015, **119**, pp 1193–1222.

[14] Kallis, C., Deliparaschos, K.M., Moustris, G.P., Georgiou, A. and Charalambous, T. Incremental 2D Delaunay triangulation core implementation on FPGA for surface reconstruction via high-level synthesis, *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept. 2017, pp 1–4.

[15] Hadjiloizou, L., Deliparaschos, K.M., Makridis, E. and Charalambous, T. Onboard real-time multi-sensor pose estimation for indoor quadrotor navigation with intermittent communication, *2022 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2022, pp 154–159.

[16] Hadjiloizou, L., Makridis, E., Charalambous, T. and Deliparaschos, K.M. Maximum Correntropy Criterion Kalman Filter for Indoor Quadrotor Navigation under Intermittent Measurements, in 2023 31st Mediterranean Conference on Control and Automation (MED), June 2023, pp 170–175.

[17] Deliparaschos, K.M., Loizou, S.G., and Zolotas, A.C., A modular UAV hardware platform for aerial indoor navigation research and development, *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp 398–404.

[18] Tech, M. FZ3 Deep Learning Accelerator Card Hardware Manual, June 2020.

[19] Stereolabs. *ZED 2 - AI Stereo Camera*, 2024.

[20] Zheng, L., Zhang, P., Tan, J. and Li, F. The obstacle detection method of UAV based on 2d LIDAR, *IEEE Access*, PP 2019, pp 1–1.

[21] Kneip, L., Tache, F., Caprari, G. and Siegwart, R. Characterization of the compact Hokuyo URG-04LX 2D laser range scanner, *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp 1447–1454.

[22] Krejsa, J. and Vechet, S. The evaluation of Hokuyo urg-04lx-ug01 laser range finder data, *Eng. Mech. 2017*, Svratka, CZ, May 2017.

[23] Labs, Pozyx. Pozyx Labs - Accurate Positioning, 2024.

[24] Kendoul, F., Lara, D., Fantoni, I. and Lozano, R. Nonlinear control for systems with bounded inputs: real-time embedded control applied to UAVS, *IEEE Conf. on Dec. and Con.*, 2006, pp 5888–5893.

[25] Voos, H. Nonlinear control of a quadrotor micro-UAV using feedback-linearization, *IEEE International Conference on Mechatronics*, 2009, pp 1–6.

[26] Bouabdallah, S. Design and Control of Quadrotors with Application to Autonomous Fly*ing*, tech. report, Ecole Polytechnique Fédérale de Lausanne (EPFL), 2007.

[27] Jia, Z., Yu, J., Mei, Y., Chen, Y., Shen, Y. and Ai, X. Integral backstepping sliding mode control for quadrotor helicopter under external uncertain disturbances, *Aerosp. Sci. Technol.*, 2017, **68**, pp 299–307.

[28] de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M. Delaunay Triangulations, in Computational Geometry: Algorithms and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp 191–218.

[29] AMD, *Zynq UltraScale MPSoC*, 2024.

[30] eProsima, *eProsima Micro XRCE-DDS Agent*, 2024.

[31] AMD, *Petalinux Tools Documentation: Reference Guide UG1144 (UG1144 v2024.2)*, 2024.

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

[32] Nyboe, F.F., Malle, N.H. and Ebeid, E. Mpsoc4drones: An open framework for ros2, px4, and FPGA integration, *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp 1246–1255.
[33] AMD, *Vivado Design Suite User Guide: Using the Vivado IDE (UG893) (UG893 v2024.2)*, 2024.
[34] Gatti, M. Complete preliminary design methodology for electric multirotor, *J. Aerosp. Eng.*, 2017, **30**.
[35] Saravanakumar, Y.N., Sultan, M.T.H., Shahar, F.S., Giernacki, W., Lukaszewicz, A., Nowakowski, M., Holovatyy, A. and Stepien, S. Power sources for unmanned aerial vehicles: a state-of-the art, *Appl. Sci.*, 2023, **13**, p 11932.
[36] Townsend, A., Jiya, I.N., Martinson, C., Bessarabov, D. and Gouws, R. A comprehensive review of energy sources for unmanned aerial vehicles, their shortfalls and opportunities for improvements, *Heliyon*, 2020, **6**, p e05285.
[37] Bauersfeld, L. and Scaramuzza, D. Range, endurance, and optimal speed estimates for multicopters, *IEEE Rob. Autom. Lett.*, 2022, **7**, pp 2953–2960.
[38] Biczyski, M., Sehab, R., Whidborne, J., Krebs, G., and Luk, P. Multirotor sizing methodology with flight time estimation, *J. Adv. Transp.*, 2020, **2020**, pp 1–14.
[39] Hwang, M.-H., Cha, H.-R. and Jung, S.Y. Practical endurance estimation for minimizing energy consumption of multirotor unmanned aerial vehicles, *Energies*, 2018, **11**, p 2221.
[40] Liscouët, J., Pollet, F., Jézégou, J., Budinger, M., Delbecq, S. and Moschetta, J.-M. A methodology to integrate reliability into the conceptual design of safety-critical multirotor unmanned aerial vehicles, *Aerosp. Sci. Technol.*, 2022, **127**, p 107681.
[41] Jin, Q., Hu, Q., Zhao, P., Wang, S. and Ai, M., An improved probabilistic roadmap planning method for safe indoor flights of unmanned aerial vehicles, *Drones*, 2023, **7**, (2), p 92.
[42] Mayoral-Vilches, V., Neuman, S.M., Plancher, B. and Reddi, V.J. RobotCore: an open architecture for hardware acceleration in ROS 2, *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp 9692–9699.
[43] Rinaldi, M., Primatesta, S. and Guglieri, G. A comparative study for control of quadrotor UAVS, *Appl. Sci.*, 2023, **13**, (6), p 3464.
[44] Sier, H., Yu, X., Catalano, I., Queralta, J.P., Zou, Z. and Westerlund, T. UAV tracking with LIDAR as a camera sensor in GNSS-denied environments, *2023 International Conference on Localization and GNSS (ICL-GNSS)*, June 2023, pp 1–7.

Cite this article: Deliparaschos K.M., Huang X., Neofytou M., Loizou S.G. and Zolotas A. Versatile UAV hardware platform for accelerating indoor aerial navigation research. *The Aeronautical Journal*, https://doi.org/10.1017/aer.2026.10143

https://doi.org/10.1017/aer.2026.10143 Published online by Cambridge University Press

https://dspace.org/

Versatile UAV hardware platform for accelerating indoor aerial navigation research

Deliparaschos, Kyriakos M.; Huang, Xuefei; Neofytou, Michalis; Loizou, Savvas G.; Zolotas, Argyrios

2026-12-31
Cambridge University Press (CUP)
https://doi.org/10.1017/aer.2026.10143
Deliparaschos KM, Huang X, Neofytou M, et al., (2026) Versatile UAV hardware platform for accelerating indoor aerial navigation research. The Aeronautical Journal, Available online 25 March 2026

Downloaded from a DSpace repository. https://dspace.org/