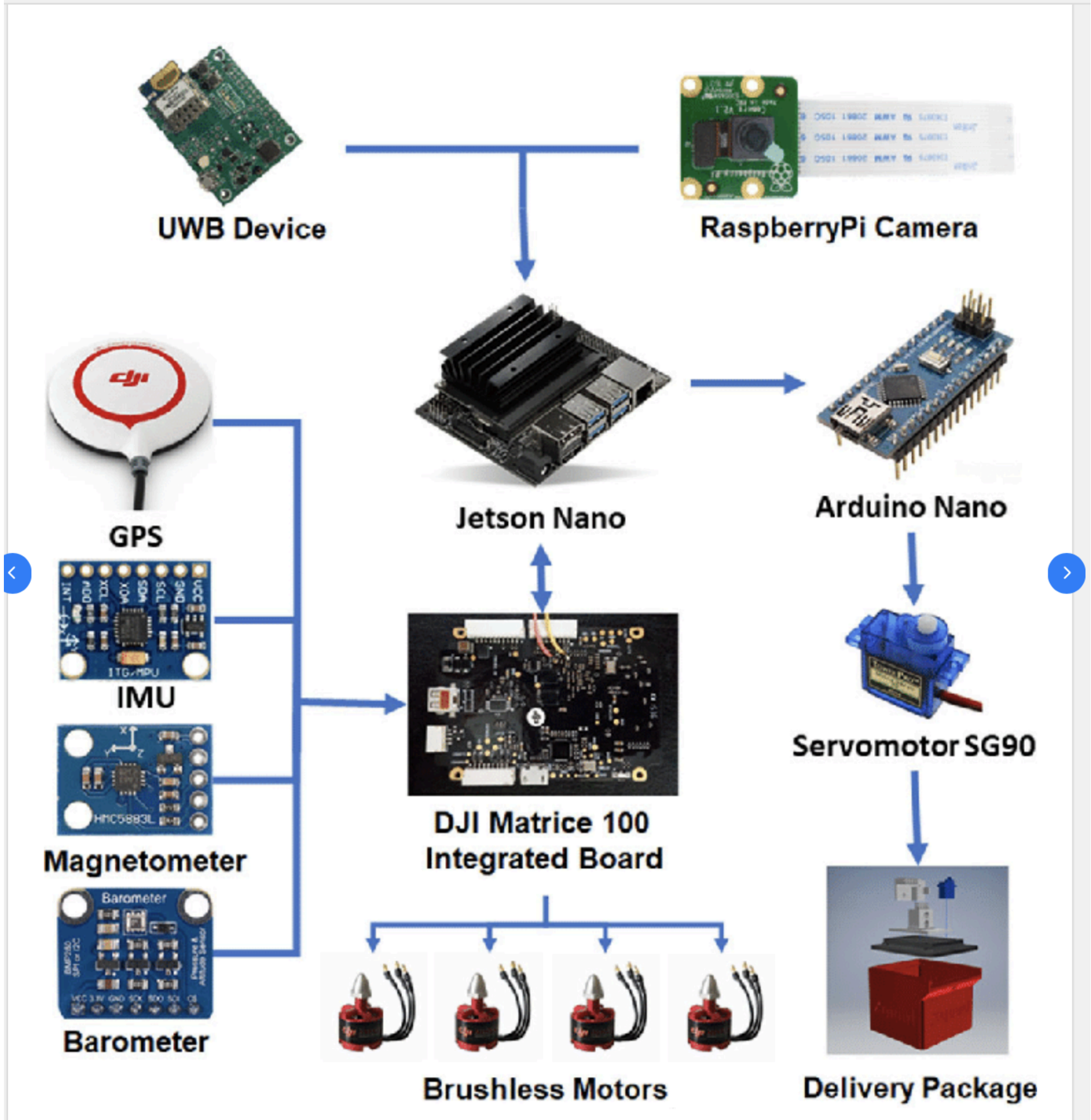
Drone Component Analysis

Raneem Youssef, Ethan Chang, Mahir Bose, Navni Athale

Drone Architecture:



Connection between the drone embedded equipment.

Jetson Nano:

The Jetson Nano plays a central role in the drone system, acting as the main computational hub for processing data from various sensors, peripherals, and other microcontrollers. Specifically, it leverages its 128-core Maxwell GPU and ARM Cortex-A57 CPU for intensive processing tasks, such as real-time image recognition, navigation, and autonomous decision-making. Connected to the Picamera module, the Jetson Nano processes visual data to enable functionalities like obstacle detection, target tracking, and visual-based navigation, crucial for applications like surveillance, photogrammetry, and delivery tasks.

The Jetson Nano also interfaces with sensors such as GPU, IMU, magnetometer, and barometer via the DJI Matrice 100 Integrated Board, providing the drone with orientation, altitude, and positioning information. This integration is vital for stable flight control and navigation, as it processes these data inputs to make real-time adjustments to the drone's flight path, especially when external commands are limited.

Some general security concerns for the Jetson Nano are:

- *Unauthorized Access and Control*: if the Jetson Nano's network interfaces or external connections are compromised, an attacker could potentially control the drone remotely. Unauthorized access could allow attackers to modify navigation paths, tamper with data, or even seize full control of the drone.
- *Data Integrity and Confidentiality*: The Jetson Nano continuously processes sensor and camera data, which may be sensitive, especially in surveillance/delivery contexts. Unsecured data channels could lead to interception or tampering of visual or telemetry data. Proper encryption for data storage and transmission is essential to safeguard against such breaches.
- *Vulnerabilities in Software and Firmware*: The Jetson Nano may be vulnerable to software and firmware exploits within its drivers or APIs. An attacker exploiting these vulnerabilities could disrupt drone operation, corrupt data, or gain unauthorized access to the system.
- *Overload Attacks*: Given the intensive processing requirements, resource exhaustion attacks (such as DoS) could potentially render the Jetson Nano unresponsive, leading to flight instability or crashes.

The Jetson Nano provides enhanced functionality compared to the arduino nano, meaning many of the Zero Trust implementations mentioned in that section are applicable. Jetson Nano Linux operating system enables more complex security measures. Enhanced Identity governance can be used to address the unauthorized access and control issue. By implementing multi-factor authentication, role based access control, and PKI, only authenticated and authorized components have access to the devices network.

Micro segmentation allows the network to be divided into isolated segments, ensuring that sensitive data, like camera data, is only accessible to components that require access. End-to-end encryption through means like TLS/SSL keeps data safe from interception and tampering, mitigating the data integrity and confidentiality concern.

ZTA using network infrastructure and software defined perimeters can be used to address DoS attacks. Monitoring the network traffic and using firewalls, allows the device to notice exhaustion attempts. Strategies such as rate limiting can keep the Jetson Nano responsive and prevent attacks during such attacks. Continuous Diagnostics and Mitigation (CDM) can also be used to detect spikes in network traffic and resources, which indicate a potential DoS attack.

Raspberry Pi Camera:

The purpose of the camera in an embedded drone system is to provide visual data and support various essential functionalities, including

- Navigation and obstacle avoidance
- Surveillance and monitoring
- Mapping and photogrammetry
- Visual inertial odometry

and much more.

The Pi Camera Module provides a convenient and accessible Python interface for working with the Raspberry Pi's camera module, which is what the Jetson Nano uses in the component diagram earlier in this document. It enables users to control the camera's settings programmatically, capture stills and record videos. The Picamera API simplifies camera operations for developers by abstracting the complexities of interfacing with the Raspberry Pi's camera hardware, allowing easy integration of camera functionality.

The Picamera software layer operates as a Python library that communicates with the camera module through the layer below it, the Multimedia Abstraction Layer (MMAL), an API for accessing the camera's hardware via the GPU. The camera mimics a rolling shutter using frames of incrementing counters, continuously streaming frame lines to the GPU where they are assembled and processed. Several tasks run in the background even when the script itself is asleep, such as automatic gain control and white balance.

When the camera is requested to capture an image or record a video, it provides the next complete frame that it assembles. The sensor detects photon counts and has two main operations: resetting or reading a row of elements. The exposure time is controlled by varying the delay between resetting a line and reading it. The processing pipeline on the GPU involves multiple stages, including transposition, noise reduction, color correction, and encoding before the data is passed to the CPU.

When interfacing with a microcontroller, the Pi Camera Module typically requires some adaptation as the library is specifically designed for the Raspberry Pi's camera architecture. In this scenario, the Jetson Nano already has a Pi camera compatible connector with device driver already installed, so interfacing is easy.

Some general security concerns regarding the usage of Picamera in the context of a drone are:

- *Unauthorized Camera Access*: if the microcontroller running the picamera software module is compromised, the camera module itself can easily be controlled by an attacker via its simple API.
- *Exploitation of Camera Hardware Vulnerabilities*: bugs or vulnerabilities in camera drivers or MMAL are frequent in documentation, and they could be exploited to crash or control the system.
- *Data Transmission and Storage Security*: the camera module continuously streams frames, so secure data handling practices are essential to prevent interception or tampering.

Also, many functions of the API have possible security issues (causing display output to crash, writing to any specified writable buffer, etc.) so thorough testing is necessary to secure the camera module.

To address the issue of a compromised host, the ideal solution would be to implement a Device Agent/Gateway-Based architecture, with the camera as the resource and the agent being a process on the microcontroller. An isolated Policy Decision Point (PDP) with a robust trust and management algorithm would be able to limit the resource gateway's contact with the compromised device agent, and even if an attacker were to gain access to the resource gateway, it is only a proxy to the actual camera module.

However, due to the direct connection between the microcontroller and the camera module, a gateway is not possible to implement. Thus, the next best option is application sandboxing through compartmentalization. Using implementations such as virtual machines or containers, camera script processes can be segmented from the rest of the microcontroller, blocking vulnerability probing and malware injection from the host asset. These processes will be the only ones allowed to communicate with a Policy Enforcement Point (PEP) inside the microcontroller's trust zone to request per-session access to the camera.

Unfortunately, there are disadvantages to this approach - ensuring proper sandboxing may require more than just simple monitoring, and operators may not have full visibility of the camera module. However, it will successfully implement a dynamic, least privileges policy that defends from a compromised host.

Regarding bugs or vulnerabilities in the firmware itself, the PiCamera repository has an open source license and is contributor-friendly, so it is susceptible to supply chain attacks. As such, the enterprise will have to actively monitor and measure the security posture of all future commits, or fork the firmware to a private repository. Attack vectors such as buffer overflow exploits, insecure PKIs, improper input validation, insufficient access controls, and so on should be noted when investigating.

To secure the frames streamed by the camera module during active and idle operation, a separate, isolated storage asset would be ideal, separating the camera module and its data into two resources, further minimizing implicit trust zones. Then, multiple gateways could be established and resource isolation would be implemented. However, it is again noted that the direct connection between the camera frame buffer and microcontroller complicates the implementation.

Another perspective could be to minimize the amount of frames streamed by shortening access sessions and increasing the weight of session time in the trust algorithm. However, this could decrease the reliability of the camera.

UWB Device:

The UWB Device can determine location with better precision, speed and cost effectiveness than other wireless technologies. The device uses radio technology and usually operates with a frequency range between 3.1 and 10.6 GHz. These low power signals allow the device to accurately determine relative position of other UWB devices up to 200 meters assuming no obstructions. While low power signals help reduce interference with other wireless devices, it also limits the range, especially in environments with obstacles.

UWB localization works by sending a “blink”, which is a low powered signal. An anchor, typically a stationary UWB device, may respond with a signal, depending on the localization method used. The time it takes for signals to travel between the two devices is used to calculate the distance between the two devices. Additionally, UWB devices typically require communication with a few other UWB devices in order to determine their position in an environment. In the drone use case, usually each UWB device performs its own ranging process with other UWB devices. As the size of the drone fleet increases, delay increases proportional to the number of drones.

Attack Vectors for UWB Devices:

- Ghost Peak: The attacker sends high power signals to overshadow the legitimate signal from the sending device. The UWB device, waiting to receive the signal, miss classifies the attacker's signal as an early copy of the signal, reducing the measured distance.
- Adaptive Injection Attack: The attacker learns the receiver's signal processing behavior. Next, the attacker injects malicious signals to interfere with the receivers regular processing. The attacker continues changing the characteristics of injected signals if the receiver adapts to interference.
- Relay Attack: The attack involves two malicious devices, one located near the sender and one near the receiver. The device near the sender intercepts its signal and forwards it to the device near the receiver, which then transmits it to the receiver. This causes the receiver to believe the sending device is closer than it actually is.
- Distance Enlargement attack: The attacker distorts the original signal to stop the receiving device from identifying it. After a delay proportional to the extra distance the malicious device wants to simulate, the attacker replays the original signal.

Zero Trust principles can be used to prevent these attacks. Firstly, enhanced identity governance can be used to minimize entry points for attackers. This involves unique device identification through using certificates and mutual authentication through public key infrastructure (PKI). Authenticated UWB devices start the communication with a digital signature using their private key. The receiving device can use the public key to verify that the device belongs to its network. This mitigates Relay attacks and adaptive injection attacks as unauthorized devices lack valid credentials.

Next, Continuous Diagnostics and Mitigation (CDM) allows for ongoing monitoring of UWB sensor data. This prevents ghost peak, adaptive injection, relay, and distance enlargement attacks by detecting anomalies in signal strength, timing, and characteristics. For example, unusual changes in signal strength may indicate a ghost peak attack. Unexpected or unusual timing could indicate a distance enlargement or relay attack. Detecting these anomalies allows the device to initiate countermeasures to keep the system safe.

https://techblog.comsoc.org/2022/01/08/ieee-802-15-3-ultra-wideband-uwb-technology-consumer-applications-and-use-cases/
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9611295
UWB Security and Enhancements - TechRxivTechRxivhttps://www.techrxiv.org › UltraSec_White_Paper-2

Arduino Nano:

In the context of a drone, the Arduino Nano is responsible for controlling servo motors like the SG90 and communicating with other higher-level processors such as the Jetson Nano. It acts as a bridge between sensors and actuators, managing real-time input from sensors and output to control components, like the servomotor.

The Arduino is a flexible microcontroller that can be easily embedded and read from different types of sensors. It is a staple for many embedded systems due to its inexpensive, versatile, and open-source nature. The Arduino Nano uses an AVR chip, which is based on a modified Harvard architecture. This architecture divides the memory into three types: EEPROM for persistent data storage, SRAM for dynamic memory allocation, and Flash memory for storing program code.
The Arduino reads sensor data from the servomotor and processes flight control algorithms. It then generates Pulse Width Modulation (PWM) signals to control the SG90 servo motor, which is used for tasks like controlling the drone's propeller angles or camera orientation. The Arduino Nano also communicates with the Jetson Nano, which acts as a more powerful processing unit for the entire drone.

The biggest security weakness in the Arduino Nano lies in the lack of memory failure control. The Arduino IDE compiler does not perform security checks during compilation. So, when a program runs out of memory, no warning is issued and the program is still run. This makes the device vulnerable to out-of memory errors and creates opportunities for attackers to exploit memory-related vulnerabilities such as heap buffer overflow and stack overflow.

In a heap buffer overflow, if consecutive memory buffers are allocated on the heap without freeing the old ones, it can lead to an overflow that overwrites critical data. For example, motor control commands could be corrupted, causing the servomotor to behave unpredictably and destabilizing the drone. While the separation of SRAM, which is used for storing data, and Flash, which is used for storing code, offers some security against executable code injection, the AVR chip is still susceptible to code reuse attacks. In addition, the low number of internal memories means that heap and stack can collide when a large number of subroutines with many local variables are called. A stack overflow could result in the overwriting of return addresses, which disrupts the drone's control logic or affects the integrity of sensor data.

https://nms.kcl.ac.uk/guillermo.suarez-tangil/papers/2016mal-iot.pdf

By implementing a Zero Trust Architecture, specifically by leveraging the concept of "Using Micro-Segmentation," interference from different parts of hardware and software can be significantly reduced. Micro-segmentation involves dividing a network into smaller, isolated segments with precise access control policies that govern how each component interacts with others. This isolation limits potential threats, making it more difficult for malicious actions or faulty processes to spread across the system.

One of the "logical components" outlined in Zero Trust Architecture suggests segmenting network parts and maintaining dynamic administrative control over each client’s communication. This enables memory access management, where specific resources are granted or denied access based on the current context. Such segmentation can be achieved through software implementations like firewalls, gateways, and intelligent switches, creating barriers around significant resources and enabling adaptive policy enforcement.

For microcontrollers like the Arduino Nano, which are Systems on Chip (SoCs), they can be treated like a network but due to the lack of a built-in operating system shifts the responsibility for peripheral communication and memory management on to the microprocessor. So, by applying Zero Trust principles to limit peripheral access strictly to relevant memory spaces, you can prevent unauthorized or unintended interactions. I.e. each peripheral can be isolated to access only the memory segments that directly relate to its specific task. When a peripheral attempts to access unrelated or critical memory, access can be blocked to safeguard sensitive operations.

This isolation strategy helps prevent memory leaks, which could otherwise lead to system vulnerabilities such as buffer overflows. By limiting the scope of memory access, you reduce the likelihood that one corrupted component, like motor control, could interfere with other critical systems, such as a camera or GPS module. Isolating memory use and peripheral control in this manner mitigates the risk of cascading failures, and limits the impact of attacks.

One real example of this is Intel's Software Guard Extensions which uses a zero trust like architecture to increase memory security. They do this by enforcing strict isolation and encryption of memory, continuous verification of trust, and access control, through a combination of software and hardware implementations just like a zero trust architecture would propose.
If the Arduino implements these zero trust types of security measures, Hardware and/or software, It would greatly prevent the risks that are involved with buffer overflows and memory leaks. Resulting in a better system and a better drone.

https://dl.acm.org/doi/abs/10.1145/2948618.2954331

GPS:

The purpose of a GPS in an embedded drone system is to determine a drone's precise location and orientation in a 3D space. It provides accurate latitude and longitude coordinates, altitude information, and the speed & direction of movement. This data allows the drone to navigate accurately, maintain a stable position, and follow pre-programmed flight paths.

GPS enables drones to perform various autonomous functions, including:

- Waypoint navigation: Following predefined routes without constant pilot input
- Return-to-home: Automatically returning to the takeoff point if connection is lost or battery is low
- Geofencing: Restricting flight to within virtual geographic boundaries

GPS also heavily contributes to drone safety:

- Position hold: Maintaining a stable hover even in windy conditions
- Obstacle avoidance: When combined with other sensors, it helps in detecting and avoiding obstacles
- Flight logging: Recording flight paths for post-mission analysis and compliance

To achieve these capabilities, GPS relies on a network of satellites orbiting Earth. The GPS receiver in the drone continuously receives signals from multiple satellites.

These signals contain information about the satellite's position and the time the signal was sent.

The GPS receiver uses trilateration to calculate the drone's precise location:
By comparing the time differences between signals from multiple satellites, the receiver can determine its distance from each satellite.
With data from at least four satellites, the receiver can accurately calculate its 3D position (latitude, longitude, and altitude).

The GPS module processes the satellite data in real-time; it continuously updates the drone's position, typically multiple times per second. This allows for accurate tracking of the drone's movement and speed.

The GPS data is integrated with the drone's flight control systems; it provides crucial input for navigation, allowing the drone to maintain stable flight paths and hover precisely. This integration enables features like waypoint navigation, where the drone can follow predefined routes.

GPS enables several advanced features in drones:

- Return-to-Home (RTH): Automatically directs the drone back to its take-off point if it loses connection or has low battery.
- Geofencing: Creates virtual boundaries to restrict the drone's flight area.
- Autonomous flight modes: Enables features like "Follow Me" or circling around a point of interest.

Sources: https://gaotek.com/category/drones/gps-drones/
https://infinidome.com/understanding-the-crucial-role-of-gps-in-navigation/
https://umilesgroup.com/en/gps-in-drones-what-it-is-for-and-when-to-use-it/

Although the gps is a component that has an integral part in navigation and coordination for the drone. It has limited security implementation due to the fact that gps collects most of its data from its environment and sends integrated boards. Because the GPS is an integrated IC most of the security will have to do with filtering out inputs to the GPS and the outputs it gives out that may be harmful rather than securing the IC itself. Due to this zero trust architecture may not be the best fit for the GPS but rather the other components around it to insure the security of the system.

Inertial Measurement Unit (IMU):

In the context of a drone, the Inertial Measurement Unit (IMU) plays a critical role in tracking and measuring the drone’s orientation, velocity, and gravitational forces, which are essential for maintaining stable and controlled flight. IMUs typically include accelerometers to detect linear acceleration, gyroscopes to measure rotation, and often magnetometers to track magnetic fields. These components allow the IMU to provide real-time data on the drone’s position and movement, keeping it steady and responsive to changes in orientation and external forces, such as weather conditions. Because IMUs are compact and lightweight, they do not add unnecessary weight that may affect drone flight efficiency. By continuously measuring and adjusting for the drone's movements, the IMU enhances the system’s robustness, ensuring smooth flight and stability even in unpredictable weather conditions.

However, IMUs also introduce certain security vulnerabilities, particularly because their sensors are susceptible to physical attacks. These attacks exploit the IMU’s sensitivity to vibrations and acoustic frequencies, causing inaccurate readings and disrupting the drone’s stability. For instance, acoustic attacks can interfere with the IMU’s ability to accurately sense orientation and acceleration, which could lead to destabilization or a loss of control. Redundancy, or having multiple IMUs, isn’t effective as all IMUs are generally vulnerable to the same types of acoustic interference.

Due to the physical characteristics of the sensors, traditional cybersecurity defenses are often ineffective for protecting IMUs. Physical attacks are typically more difficult to protect against because they can directly manipulate the sensor outputs through environmental factors, making them particularly stealthy and difficult to detect. For instance, targeted acoustic attacks on IMU components can cause sensor reading inaccuracies. This kind of interference can mask the drone's actual orientation and speed, potentially leading to flight path deviations, crashes, or complete loss of control.

Another challenge with IMU vulnerabilities is that the sensor fusion algorithms, which combine data from the accelerometer, gyroscope, and magnetometer, can complicate the isolation of any single faulty sensor. When a sensor’s data is compromised, the fusion process might smooth out or compensate for the errant readings, masking the failure and making it difficult to identify which component is malfunctioning. This ambiguity in identifying compromised sensors can result in prolonged instability, as the system relies on inaccurate data until the fault is detected and isolated. Thus, advanced fault detection and isolation methods, such as smooth variable structure filters, are increasingly important. These methods monitor changes in physical parameters, such as mass or inertia, to detect anomalies, providing a more granular way to pinpoint and isolate specific faulty sensors.

In addition, alternative recovery methods have been developed to compensate for IMU failures without the cost of additional sensors. For instance, by using geometric recovery techniques that rely on the remaining position and heading information, drones can maintain a basic level of stability and control even if the IMU data is compromised. In this approach, the drone's control system calculates its orientation by analyzing the thrust vector direction, which is determined by the position and heading data. This method allows the drone to remain stable enough to avoid immediate crashes, providing a crucial safeguard until it can either restore normal operation or make a safe landing. Overall, while IMUs are crucial for maintaining drone stability and control, their susceptibility to physical attacks highlights the need for robust security and fault-tolerance strategies.

Sources:
https://friends.cs.purdue.edu/pubs/IROS19.pdf
https://arxiv.org/pdf/2409.16438
https://www.jouav.com/blog/inertial-measurement-unit.html

To implement the Zero Trust principle of multi-factor authentication, the IMU can leverage other sensors in the drone, such as the GPS, barometer, and magnetometer, to cross-check data and detect inconsistencies. This approach is particularly valuable during an acoustic attack, which IMUs are especially vulnerable to. By validating that data is consistent across all sensors, the drone can ensure the integrity of its readings before making adjustments to its flight path.

Isolation can be implemented by treating the individual IMU components (accelerometer, gyroscope, and magnetometer) as separate entities. Each sensor's data should be analyzed independently before combining it for transmission to the DJI Matrice board, which controls the flight path. This separation minimizes the impact of compromised data by containing anomalies within a single component and preventing them from propagating through the system.

To enhance the security of sensor data, authentication and authorization measures are critical. At the sensor level, IMU data can be digitally signed to verify its integrity before processing. Cryptographic keys can be used to ensure that the data has not been tampered with during transmission. In addition, securing internal communications between the IMU and the drone’s main processor is essential to protecting the drone against malicious actors. This can be achieved through end-to-end encryption and the use of authenticated channels, aligning with Zero Trust principles to maintain data confidentiality and integrity. By encrypting the communication pathways, even if attackers intercept the data, it remains unintelligible without the appropriate decryption keys. Moreover, authenticated communication channels should be established to verify the identity of both the sender and receiver, reducing the risk of spoofing or man-in-the-middle attacks.

DJI Matrice 100 Integrated Board:

The DJI Matrice 100 integrated board serves as the central control system for the drone, handling flight control, communication, and data processing.

The integrated board incorporates several key components:

- N1 Flight Controller: This keeps the drone stable and responsive to commands throughout flight[1]. It processes sensor data and controls the motors to maintain stability and execute flight maneuvers.
- DJI Lightbridge: This integrated video transmission system allows for long-range HD video streaming and flight control, with a range of up to 3.1 miles.
- Expansion Ports: The board includes universal power and communication ports like CAN and UART to connect additional sensors, cameras, and other payloads.

The Matrice 100 incorporates several security measures at the hardware and software level:

Secure Boot Chain:
The drone uses a secure boot process to verify firmware integrity.
Step 1. The BootROM code (burned into the chip during manufacturing) verifies the secondary boot loader.
Step 2. The boot loader then verifies and loads the flight control firmware, secure operating system, and Linux kernel.
Step 3. Each level in the boot chain must be verified by the upper level firmware.
This process helps prevent unauthorized firmware modifications or malicious code injection.

Encrypted Firmware:
The firmware is encrypted and signed by DJI to ensure confidentiality and integrity. It can only run after being verified and decrypted[3].

Device Unique Serial Number:
A unique serial number is stored in secure storage to prevent device cloning and impersonation[3].

Debug Interfaces Disabled:
Debugging interfaces like JTAG and serial ports are disabled on production units to prevent unauthorized access to the firmware[3].

There are some potential security issues to be aware of:
- The flight logging system records sensor data and cached images. This data can only be fully decrypted using DJI's API, which requires sending the data to DJI's servers. This raises concerns about data privacy and potential access by third parties.
- While DJI claims end-to-end encryption for data transmission, there have been concerns about the potential for data interception or unauthorized access, especially given DJI's connections to China.
- Like any complex system, there's always the potential for undiscovered software vulnerabilities that could be exploited.
- While not specific to the integrated board, the drone's reliance on GPS for navigation makes it potentially vulnerable to GPS spoofing attacks.

Implementing zero trust principles into a DJI Matrice 100 Integrated Board requires a comprehensive approach that addresses various aspects of security.

Identity & Access Management
Strong Authentication: Implement multi-factor authentication (MFA) for all users accessing the drone's control systems. This could involve combining something the user knows (i.e. a password) with something like a security token or something they are, which would be biometric data.
Least Privilege Access: Grant users only the minimum level of access required to perform their tasks. Regularly review and adjust access permissions to ensure they remain appropriate

Device Security

Device Health Verification: Before allowing the Matrice 100 to connect to your network or control systems, verify its health and compliance status. This includes checking for up-to-date firmware, security configurations, and the absence of any known vulnerabilities15.
Secure Boot: Ensure that the Matrice 100's firmware uses secure boot processes to verify the integrity of the system before startup

Continuous Monitoring and Validation
Encryption at Rest: Encrypt all data stored on the Matrice 100, including flight logs, imagery, and any other sensitive information3.
Secure Data Transmission: When transferring data from the drone to other systems, use secure protocols and consider implementing a virtual private network (VPN) for added protection

Software and Firmware Management
Regular Updates: Keep the Matrice 100's firmware and all associated software up to date with the latest security patches
Controlled Software Installation: Implement strict controls on what software can be installed on the drone and its control systems. Verify the integrity and source of all software before installation.

Sources:
https://www.microsoft.com/insidetrack/blog/implementing-a-zero-trust-security-model-at-microsoft/
https://security.dji.com/data/overview/
https://www.rippling.com/blog/how-to-implement-zero-trust

Magnetometer and Barometer:

Due to the specialized nature of these sensors, there were not many significant vulnerabilities or Zero Trust approaches that could be integrated.

A barometer is used to measure air pressure. Since air pressure is negatively correlated with elevation, it keeps the drone at the optimal altitude. The Magnetometer, by measuring the earth's magnetic field, helps the drone determine its orientation in terms of cardinal directions.

Magnetometers are susceptible to interference from magnetic signal generators. Multiple methods for determining orientation are helpful to ensure that the drone will remain resilient to these attacks.

A couple approaches can be used to implement Zero Trust principles in these devices. Firstly, Micro-segmentation can isolate data streams from the magnetometer and barometer, ensuring that each sensor is on a unique and protected networking segment. This isolation reduces the chance that a breach of one sensor will compromise the whole system, especially when redundancy is built in, through multiple sensors performing similar tasks. For example, using both a GPS sensor and barometers allows the sensor to cross-validate altitude data.

Implementing a Continuous Diagnostics and Mitigation (CDM) system allows ongoing monitoring of sensor data. This enables the fast detection of anomalies and deployment of countermeasures.

https://www.azom.com/article.aspx?ArticleID=21378#:~:text=Barometers%20maintain%20a%20stable%20altitude,magnetic%20field%20strength%20and%20direction.
https://www.uavnavigation.com/company/blog/uav-navigation-depth-magnetometer