Tbilisi Free University

School of Computer Science, Mathematics and Engineering
(MACS[E])

Electrical and Computer Engineering Program

Luka Pkhaladze and Luka Balanchivadze

Junior Project

Lidar Drone

Head of project: Zviad Sulaberidze

Tbilisi
2024

Contents

List of Tables, Graphs, Formulas, and Figures

Annotation

This project focuses on the development of a drone equipped with a **YDLIDAR T-mini Plus sensor**, designed to perform **real-time 3D mapping** and **spatial data transmission**. The drone is built around an **ESP32 microcontroller**, coupled with an **NRF24L01 wireless module** to facilitate the collection and transmission of Lidar data to a ground station. At the ground station, the data is processed to create a **3D point cloud visualization** of the environment.

The Lidar sensor is positioned to capture data within a specific angular range of **60° to 120°**, selected based on the physical placement of the Lidar on the drone. This narrowed range optimizes the scanning area by focusing on the forward-facing region beneath the drone, reducing redundant or irrelevant data from other directions. This selective data acquisition improves **computational efficiency** and **reduces bandwidth usage** during transmission, while still maintaining the required accuracy for effective mapping.

A key aspect of the project lies in managing the **high-speed data flow** generated by the Lidar sensor, which is transmitted over a **limited-bandwidth communication system** using the NRF24 module. This challenge required the design and implementation of a **custom buffering algorithm** to handle potential **data overflow** issues, ensuring that no crucial data is lost during the transmission process. The algorithm incorporates **error detection and correction** mechanisms, which help mitigate data loss due to interference or signal degradation over longer distances.

Moreover, the system employs **data compression techniques** to minimize the size of each transmission packet, further enhancing the efficiency of the communication link. This balance between data size and transmission frequency was carefully calibrated to ensure the real-time performance of the system, which is critical for applications like **dynamic terrain mapping**.

At the ground station, the transmitted data is processed using a multithreaded software approach, where one thread manages **real-time data reception** while another handles **point cloud rendering**. By separating these tasks, the system can seamlessly visualize data as it is received without introducing delays. The 3D point cloud is continuously updated, giving users an accurate, real-time representation of the scanned area. Advanced filtering techniques are also applied to eliminate noise and enhance the clarity of the final 3D model.

The project demonstrates an innovative approach to creating a **cost-effective**, **portable**, and **adaptable solution** for aerial 3D mapping. The lightweight and power-efficient hardware configuration, combined with efficient data handling techniques, make the system highly suitable for **real-time field use** in a variety of sectors. Potential applications include **precision agriculture**, where drones equipped with Lidar can be used for **crop monitoring** and **land surveying**; **urban planning**, where real-time 3D maps assist in infrastructure development; and **environmental monitoring**, where drones can assess forest growth, water bodies, or disaster-struck area

Introduction

In recent years, advancements in autonomous technologies have led to increased use of drones for various applications, ranging from aerial photography to environmental monitoring. Among these technologies, Lidar (Light Detection and Ranging) systems have proven to be crucial for precise distance measurement and 3D mapping, especially in aerial vehicles. The project presented here focuses on the development of a Lidar-equipped drone, capable of collecting, processing, and transmitting spatial data for real-time 3D point cloud visualization.

The motivation behind this project stems from the growing demand for efficient and cost-effective methods to map and monitor environments with high accuracy. Existing solutions for 3D mapping often rely on large, expensive systems. For instance, the DJI Lidar drone is one of the top-performing drones on the market today. However, its price, which ranges in the thousands, makes it practically unaffordable for a large portion of the population, especially for residents of Georgia. So, this project aims to provide a more affordable and portable solution by integrating a YDLIDAR T-mini Plus sensor with an ESP32 microcontroller and a NRF24L01 module for data transmission. The drone is designed to collect Lidar data from specific angles and send it to a ground station for real-time visualization.

In addition to the reasons mentioned above, our decision to create a Lidar drone was also influenced by the fact that this was our first large-scale project, and we wanted to gain knowledge and experience across various aspects. Both team members had a strong interest in the field of drones, but we wanted our drone to serve a real purpose beyond just entertainment. It was this combination of our interests, preliminary research, and the limited use of Lidar drones in Georgia that motivated us to be among the first to implement such a project

The novelty of this project lies in its focus on optimizing the data collection and transmission process. The project also addresses challenges related to buffer overflow during data transmission and implements effective solutions to ensure seamless data flow.

This drone has the potential to be used in various fields, such as agriculture, urban planning, and disaster management, where accurate mapping of environments is essential. The potential applications of Lidar drones are highly relevant in Georgia, as agriculture is one of the most widespread industries in the country. However, Lidar drones are currently not utilized in various economic processes. Natural disasters are also common in Georgia due to the abundance of water and glaciers. The use of Lidar drones for disaster prevention and response is not yet a common practice. We believe the high price of existing Lidar drones is the primary reason for this. The low-cost prototype we have developed, priced between 400 and 600 dollars, would make it feasible to deploy in these fields. Moreover, the low cost and adaptability of the system make it suitable for a wide range of users, from researchers to hobbyists.

The primary goal of this project is to create a functional prototype of a Lidar drone capable of collecting and transmitting spatial data for real-time 3D mapping. The project utilizes several advanced techniques, including multi-threading on the ESP32 for efficient data handling. Despite the technical challenges involved, such as managing high-speed data flow and ensuring reliable communication between the drone and ground station, the project demonstrates a practical and innovative approach to drone-based 3D mapping.

Problem Definition and Methodology

Research Conducted in Connection with the Topic of the Project

In recent years, the integration of Lidar technology with drone systems has transformed various industries by enabling high-resolution mapping and data collection. This section aims to provide the necessary background to understand the purpose, relevance, theoretical, and practical value of this project, highlighting key aspects of the field and addressing the research conducted in connection with the project topic.

Technologies in the Field and Market Overview: Lidar technology, particularly when combined with unmanned aerial vehicles (UAVs), has gained significant traction in fields such as agriculture, forestry, urban planning, and disaster management. These systems offer a range of applications, including terrain mapping, vegetation analysis, and structural assessments. The market for drone-based Lidar systems is expanding rapidly, driven by technological advancements, decreased costs, and increased accessibility for smaller enterprises and individual users.

Recent advancements in microelectronics and sensor technology have led to the development of compact and affordable Lidar sensors, making them suitable for integration into drones. Furthermore, improvements in data processing software have enhanced the ability to analyze and visualize the data collected, providing actionable insights for various industries.

Challenges in the Field: Despite the progress, several challenges remain in the implementation of Lidar drones. One of the primary issues is the high cost associated with existing Lidar systems, which can be prohibitive for small businesses or individual users. Additionally, the integration of Lidar data with existing workflows often requires specialized knowledge and technical expertise, creating a barrier to entry for potential users.

Another challenge is the management of the vast amounts of data generated by Lidar sensors. Efficient data processing and storage solutions are needed to handle this influx, particularly for real-time applications. Furthermore, environmental factors such as weather conditions and obstacles can affect the accuracy and reliability of Lidar data collection.

Development Prospects in the Field: The future of drone-based Lidar technology appears promising, with ongoing research aimed at improving sensor accuracy, data processing

capabilities, and system integration. As costs continue to decrease, we can expect broader adoption across various sectors, including environmental monitoring, land surveying, and infrastructure management.

Moreover, advancements in artificial intelligence and machine learning are poised to enhance data analysis capabilities, allowing for more sophisticated interpretations of Lidar data. This could lead to new applications and improved decision-making processes in fields such as precision agriculture and urban planning.

Task Definition: This project aims to develop a low-cost Lidar drone system capable of real-time 3D mapping and data visualization. The proposed system will utilize an affordable Lidar sensor integrated with an ESP32 microcontroller and an NRF24L01 module for data transmission. By optimizing the data collection process and addressing existing challenges in the field, this project presents an innovative approach to enhancing the accessibility and practicality of drone-based Lidar technology.

Innovative Aspects of the Project:

- **New and Innovative Idea:** The project offers a cost-effective solution for real-time 3D mapping, making advanced Lidar technology accessible to a wider audience. By focusing on affordability without sacrificing performance, the project opens up opportunities for small businesses and individual users.
- **Innovative Problem-Solving:** The project addresses the high costs and complexities associated with existing Lidar systems by developing a prototype that combines readily available components. This approach streamlines the process of data collection and analysis, making it easier for users to implement Lidar technology in their workflows.
- **Process Optimization:** By integrating a custom algorithm for data collection and optimizing the transmission of Lidar data, the project enhances the efficiency of data handling and visualization. This not only improves system performance but also reduces the required bandwidth for wireless communication.

Technical Task of the Project: The technical objectives of this project include:

- Developing a functional prototype of a Lidar drone capable of collecting and transmitting spatial data.
- Implementing algorithms for data compression and transmission optimization.
- Creating software for real-time visualization of 3D point cloud data

Material and Technical Tools Needed: The project will utilize the following components:

- Flight controller configured using BETAFLIGHT and ESC board, configured using BLHELI.
- FlySky transmitter and receiver for controlling the drone.
- YDLIDAR T-mini Plus sensor for data collection.
- ESP32 microcontroller for data processing and communication.

- NRF24L01 module for wireless data transmission.
- Python and Matplotlib for data visualization.

Need/Relevance of the Project: Given the increasing demand for efficient and cost-effective mapping solutions, this project addresses a critical gap in the market by making Lidar technology more accessible. The relevance of the project is underscored by the growing need for accurate environmental monitoring and mapping capabilities in various sectors.

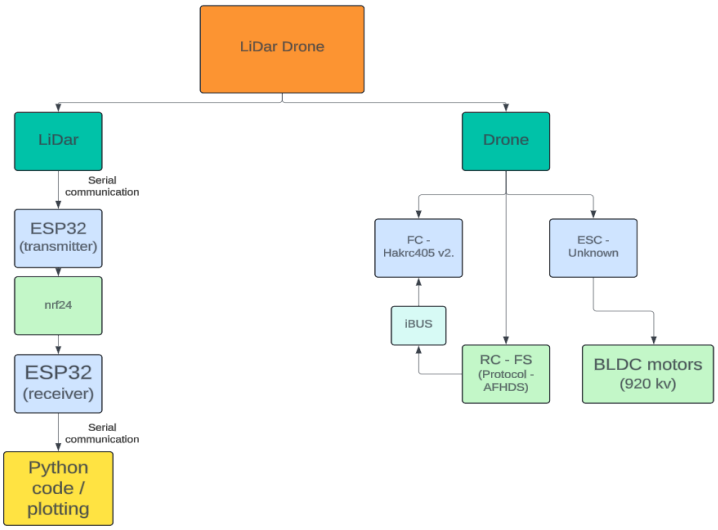
Potential Commercialization of the Project: The successful development of this Lidar drone system opens the door for potential commercialization. Target markets include agricultural businesses, environmental agencies, and urban planners who require affordable and effective mapping solutions. By providing a scalable product, the project has the potential to contribute significantly to the emerging market for drone-based Lidar systems.

Technical Side of the Project

This section outlines the comprehensive technical aspects of the Lidar drone project, detailing the methodologies employed, analyses conducted, visual representations, software components, challenges faced, and the solutions implemented. Each stage of the project is described to provide clarity on the technical task and its execution.

Our project consists of two main components. The first component focuses on the 3D modeling of the drone, its assembly, and performance testing. The second component involves debugging the Lidar system, establishing communication with the sensor, collecting data, transmitting the information wirelessly over long distances, processing the received data, and displaying a live 3D point cloud visualization. Finally, the integration of these two components involves mounting the Lidar sensor onto the drone and incorporating the microcontroller into a cohesive system. This step also focuses on enhancing the performance of the Lidar during the drone's flight.

Picture 1: Block Diagram



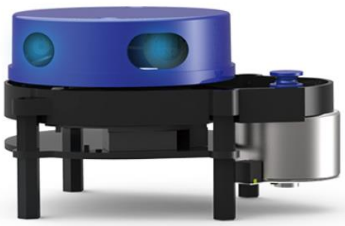
Drone Design and Modeling:

When you purchase a drone kit, assembling it is usually straightforward. However, in our case, this wasn't an option because our drone wasn't just a standard drone; it needed to support a Lidar sensor. We had to carefully consider its design and structure to ensure it could lift the relatively heavy weight of the Lidar while maintaining balance. Additionally, we had to account for the rotation of the Lidar to prevent any negative impact on the drone's performance.

Initially, our original plan was to use the YDLIDAR X4 (later it turned out to be faulty). Since this model is larger in size compared to the T-mini-pro, the drone's size was quite large and designed accordingly. The initial concept looked like this:

Picture 2: The first prototype of the drone

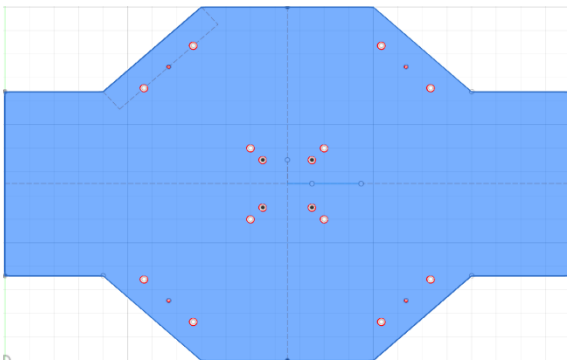
Picture 3: Ydlidar X4



However, in the end, we opted for a different Lidar, which is much smaller and more compact. This decision resulted in significant changes to the design and dimensions of the drone.

- We machined the main body of the drone from textolite using a CNC machine and painted it black.

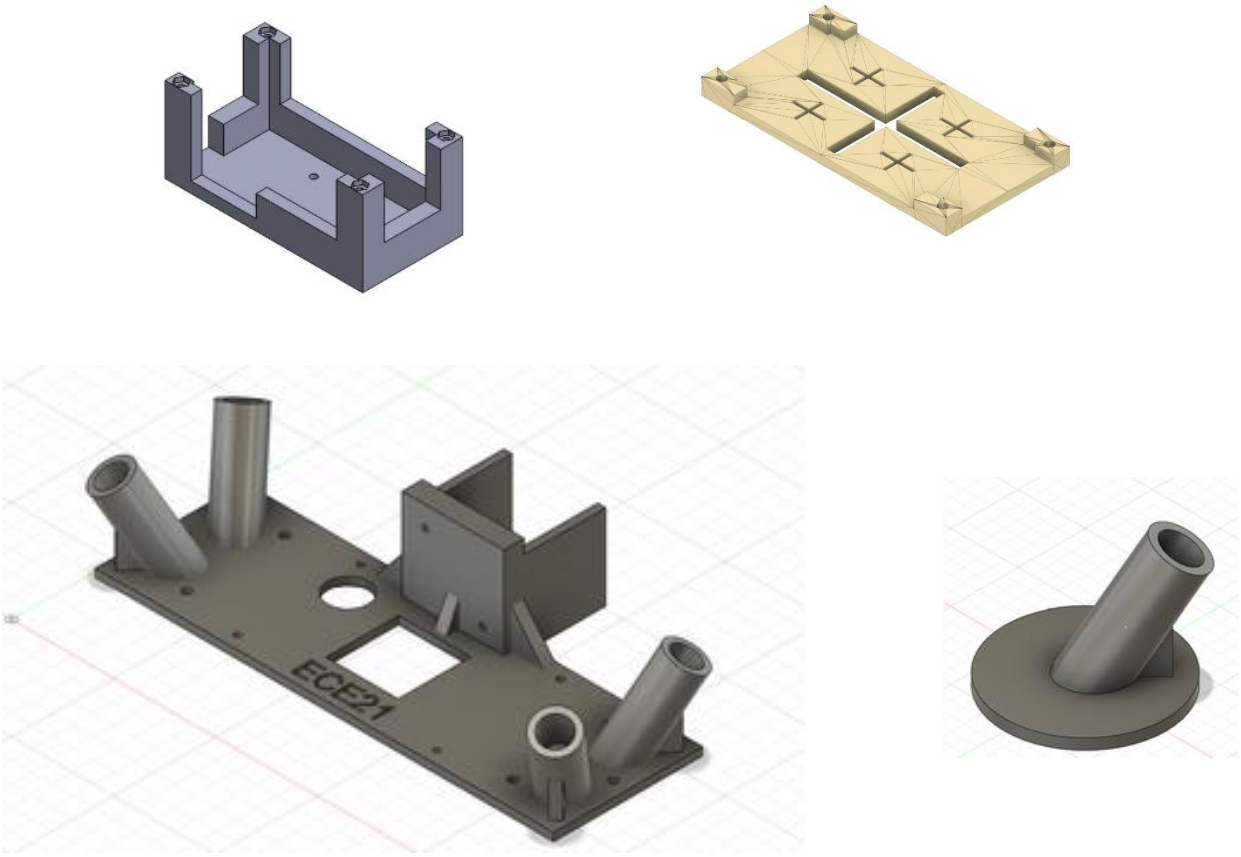
Picture 4: Model of Drone body



Picture 5: Drone body

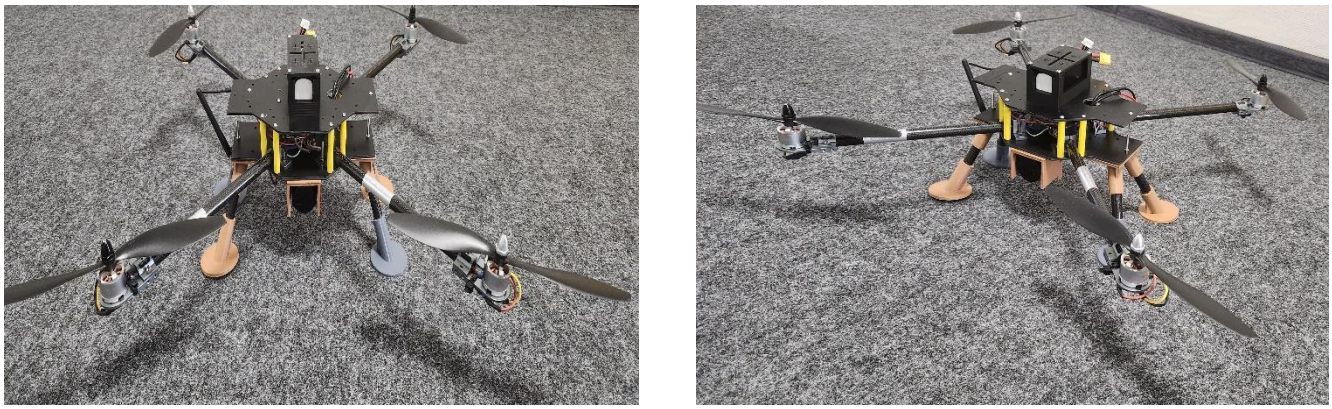
- We designed and built 3D models for the battery mount, Lidar mount, and drone legs to ensure proper fit and functionality. These components were customized to securely hold the equipment and optimize.

Picture 6: 3D models of drone parts: battery holder, drone leg, and Lidar holder



Integrating the 3D models for the battery mount, Lidar mount, and drone legs resulted in the final visual design of the drone. This design is well-suited for the project and provides a stable flight experience during Lidar scanning.

Picture 7: Final appearance of drone



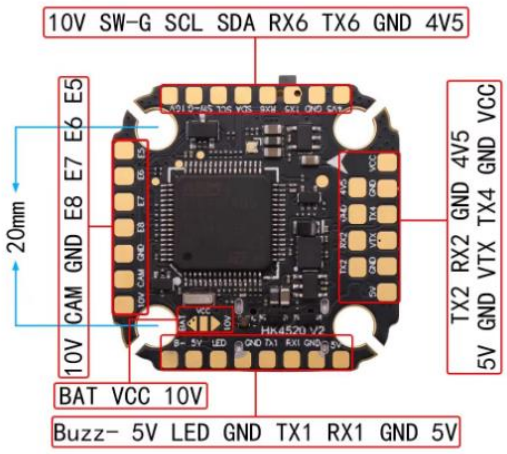
Flight Controller / Betaflight

The **Flight Controller (FC)** is the central component of our drone, often referred to as its "brains." It plays a critical role in managing inputs from the Radio Controller (RC) and translating those inputs into outputs that are sent to the Electronic Speed Controllers (ESCs), which regulate the speed of the motors. This intricate relationship between the FC, ESCs, and motors enables the drone to perform a variety of maneuvers, such as hovering, turning, and changing altitude, ensuring responsive and controlled flight.

For our project, we initially planned to use the **Pixhawk v2.4.8** as the flight controller, drawn by its advanced features and capabilities, particularly its ability to process sensor data in-flight. This would have allowed for onboard analysis and processing of the Lidar data in real-time, significantly enhancing our data collection capabilities. Unfortunately, we encountered a significant setback when a faulty circuit in the Pixhawk caused it to short out during the initial setup, rendering it unusable. This unexpected issue forced us to seek an alternative flight controller that could still meet our project's rigorous requirements.

Picture 8: Pixhawk 2.4.8

Picture 9: HAKRC F405 v2



Ultimately, we opted for the While it lacks some of the advanced processing power and functionalities of the Pixhawk, it has proven to be a reliable and cost-effective alternative for managing the drone's core functions. The HAKRC F405 v2 Mini comes equipped with several onboard sensors, including an accelerometer, gyroscope, and sometimes a barometer, which are essential for maintaining flight stability and ensuring the drone operates effectively.

The **accelerometer** measures the drone's acceleration along various axes, helping the flight controller maintain stable flight, while the **gyroscope** tracks the drone's angular velocity, providing critical data regarding its orientation and rotation. Together, these sensors allow the flight controller to make rapid adjustments to the motor speeds, ensuring a smooth and controlled flight experience. Additionally, in some configurations, a **barometer** is utilized to measure atmospheric pressure, facilitating more accurate altitude gauging, particularly during autonomous missions or altitude hold functions.

Given that our drone is not equipped with an onboard FPV (First Person View) camera, operating it from the ground presents unique challenges. Therefore, we rely on **angle mode** for piloting the drone, making the onboard sensors even more crucial for functionality.

For our setup, we employed **Betaflight** as the configuration software for the flight controller. Betaflight is a widely-used firmware that specializes in tuning and configuring drones, particularly for FPV capabilities and racing applications. Within the Betaflight Configurator, we focused on several key areas, including configuring the flight controller's communication with the ESCs and tuning the **PID (Proportional, Integral, Derivative)** settings. These PID settings are vital for fine-tuning how the flight controller responds to changes in the drone's position, ensuring a balance between responsiveness and stability during flight.

Additionally, we configured the **radio receiver (RX)**, set up flight modes, and calibrated the onboard sensors to guarantee optimal drone performance. This process included establishing the arming sequence to ensure the motors would only spin once the drone is in a safe state to fly. We also mapped the controls from the radio transmitter to the corresponding actions on the drone, such as adjusting throttle, pitch, yaw, and roll.

While the HAKRC F405 v2 Mini does not boast as many features as the Pixhawk, it provides all the necessary functionality to control the drone's movement effectively and respond to inputs in real time. The shift from an advanced flight controller to this simpler model redirected our project focus from processing Lidar data onboard to managing the data externally. This change allowed us to leverage a more powerful ground station for real-time analysis and mapping, ultimately enhancing the overall effectiveness of our drone.

The main tabs of the BETAFLIGHT we used were:

1. setup tab -> in this tab, users can calibrate accelerometer, which is a crucial part of the FC. Moreover, one can see how the drone reacts to the movement of the frame, i.e. how the brain perceives the orientation of the drone.
2. ports tab -> in this tab you can configure the essential serial communications for peripheria. For instance we use this tab for connection between receiver and FC (more on this in the Transmitter/Receiver part).
3. configuration tab -> according to our design, the FC is mounted on the top plate of the drone, therefore it is rotated. It is the user's responsibility to tell software to adjust for this change in the "Board and Sensor Aligment" part of this tab.
4. the battery tab has not been changed.
5. PID -> this is presumably the most important tab in the configurator. This tab is responsible for pid tuning of the drone, how it reacts to the RC input, how fast is the reaction, how much overshoot is done, etc.
6. Receiver tab -> this tab was used for checking the health of the transmitter and receiver.
7. Modes tab -> in this tab we configured angle mode and arming and aAssigned special switches to them.

8. Motors tab -> the orientation the motors spin should be considered carefully. It is important to match the given orientation in the software to the hardware setup. In this tab one can adjust these settings.
9. OSD and Video Transmitter tabs were skipped, because we are not using FPV camera.

Transmitter/Receiver – FlySky FS-ia6/IA6B

Picture 10: FlySky FS-ia6/IA6B

For controlling our drone, we utilize the **FlySky FS-ia6 transmitter** and its corresponding **IA6B receiver**. This combination is well-known for its reliability and affordability, making it a popular choice among both hobbyists and professionals. The FS-ia6 transmitter is a **6-channel remote control** that operates on the **2.4 GHz frequency**, providing a good range with minimal interference from other devices operating within the same spectrum. This frequency band is particularly advantageous for aerial applications, as it delivers robust performance, even in urban environments where potential signal congestion may occur.

The **IA6B receiver** is designed to work seamlessly with the FS-ia6 transmitter, establishing a stable and secure link between the operator and the drone. The receiver's compact design facilitates easy integration into our drone's frame, while its ability to support up to **10 channels** allows us to customize control inputs for various functionalities, including throttle, yaw, pitch, and roll, as well as additional channels for activating other systems, such as lights or camera triggers.

One of the significant advantages of the FlySky system is its **easy binding process**, which allows for quick and efficient pairing between the transmitter and receiver. This feature is particularly

useful for troubleshooting and maintenance, enabling rapid reconfiguration if either component needs replacement or reset.

In terms of operational features, the FS-ia6 transmitter is equipped with various controls, including **dual-rate settings**, which allow the user to adjust the sensitivity of the control sticks for smoother flight or more responsive handling, depending on the situation. This flexibility is crucial for novice pilots, as it permits gradual skill development while also providing the option to switch to more sensitive settings for advanced maneuvers.

Additionally, the transmitter includes a built-in **LCD screen**, which displays real-time telemetry data such as battery voltage, signal strength, and model settings. This feedback is vital during flight operations, as it helps the operator maintain awareness of the drone's status and make informed decisions while in the air.

Integration with the Flight Controller

To integrate the FlySky FS-ia6 transmitter and IA6B receiver with the HAKRC F405 v2 Mini flight controller, we connected the receiver's output channels to the corresponding input channels on the flight controller. This configuration allows the flight controller to receive and interpret signals from the transmitter, translating them into actionable commands for the drone's motors and other systems.

We utilized an **I-BUS** connection for this setup, which, while not as widely used today as it once was, remains functional and reliable. I-BUS is a form of serial communication that streamlines the process of transmitting control signals. When configuring the flight controller in Betaflight, it is crucial to select the correct form of connection between the receiver and FC during the firmware installation; otherwise, the connection may not be established correctly. Unfortunately, we encountered this issue during our setup, which caused significant delays in our project, pushing back our timeline by over a week.

Despite these challenges, the combination of the FlySky FS-ia6 transmitter, IA6B receiver, and HAKRC F405 v2 Mini flight controller has ultimately provided a robust and effective system for controlling our drone.

Lidar Part and software:

As mentioned earlier, our initial plan was to use the YDLIDAR X4, which is quite large. However, after receiving the scanner, we discovered that it was faulty and non-functional. After discussions with the manufacturers and reviewing technical data, we decided to switch to the YDLIDAR T-mini Plus. The key advantages of this model were its compact size and lightweight design, meaning the motor rotation had virtually no impact on the drone's flight stability. Additionally, it was specifically designed for use in airborne devices and could be easily communicated with without the need for specialized software.

Picture 11: Ydlidar t-mini plus

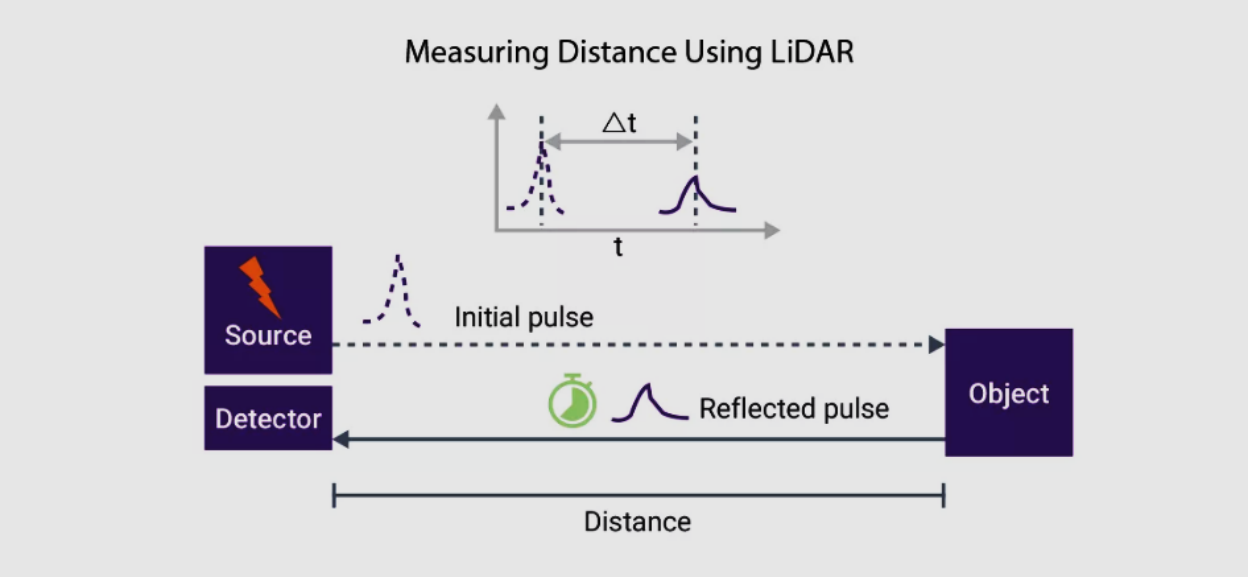
How Lidar Sensors Work:

1. Overview of Lidar Technology: Lidar, which stands for Light Detection and Ranging, is a remote sensing technology that uses laser light to measure distances to objects. It emits laser pulses and measures the time it takes for the light to reflect back to the sensor, allowing for precise distance calculations. This technology is widely used in various fields, including mapping, surveying, forestry, autonomous vehicles, and robotics.

2. Basic Principle: The fundamental principle of Lidar involves emitting laser beams and detecting the time it takes for the emitted light to bounce back from an object. The time delay is used to calculate the distance using the speed of light. **The basic formula is:**

$$Distance = VelocityOfLight \times time$$
$$Velocity\ Of\ Light = 300 \times 10^3\ km/s$$

Picture 12: the way lidar works

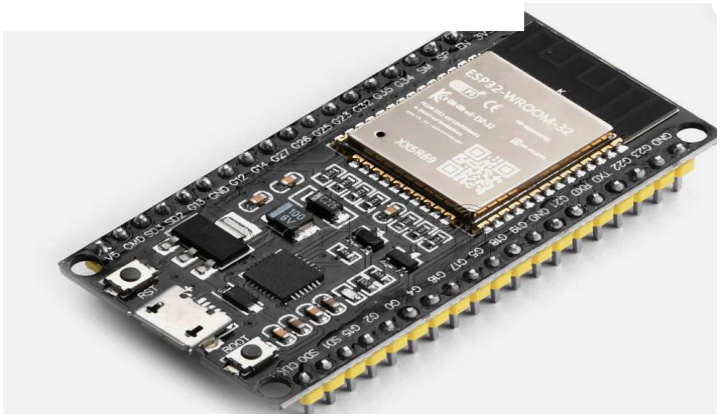


ESP32 microcontroller:

The ESP32 is a powerful and versatile microcontroller developed by Espressif Systems. It is designed for a wide range of applications in the Internet of Things (IoT), embedded systems, and robotics. The ESP32 features built-in Wi-Fi and Bluetooth capabilities, making it an excellent choice for projects that require wireless communication. Here are some key specifications and features:

- Dual-Core Processor: The ESP32 includes a dual-core processor (up to 240 MHz), which allows for efficient multitasking and processing.
- Integrated Wi-Fi and Bluetooth: The microcontroller supports both Wi-Fi (802.11 b/g/n) and Bluetooth (Classic and BLE), enabling easy connectivity.
- Rich Peripheral Support: It includes various interfaces, such as UART, SPI, I2C, PWM, ADC, and DAC, allowing it to connect to multiple sensors and devices.
- Low Power Consumption: The ESP32 is designed to be power-efficient, making it suitable for battery-operated applications.
- Flexible Programming: It can be programmed using various environments, including the Arduino IDE, PlatformIO, and Espressif's own ESP-IDF (IoT Development Framework)

Picture 13:ESP32



We utilize two ESP32 microcontrollers in the project due to their high processing speed and dual-core architecture, which allows for the execution of two parallel processes. This capability enables rapid and simultaneous data transmission and reception. One ESP32 is dedicated to the transmitter circuit mounted on the drone, where it interfaces with the Lidar sensor. The second ESP32 is positioned at the ground station, receiving the information sent from the drone and relaying it to a computer for further visualization of the Lidar data.

Transmitter side code review:

The code is designed to facilitate the transmission of data from a YDLIDAR sensor to a ground station using the NRF24L01 wireless communication module. The system leverages the capabilities of the ESP32 microcontroller to collect and process Lidar data, employing FreeRTOS for efficient concurrent task management.

At the beginning of the code, essential libraries are included to provide the necessary functionality. The Arduino.h library serves as the foundation for standard microcontroller operations, while SPI.h enables communication with the NRF24L01 module over the Serial Peripheral Interface. The nRF24L01.h and RF24.h libraries facilitate the interaction between the ESP32 and the NRF module, allowing for wireless data transmission. Additionally, HardwareSerial.h is used to establish serial communication with the Lidar sensor.

To handle the data being transmitted, a Payload structure is defined, containing an array of 32 bytes. This structure is crucial for organizing the Lidar data into manageable packets that can be sent wirelessly. The NRF module is configured using the RF24 library, with the chip enable (CE) and chip select not (CSN) pins specified, and a unique address is assigned for communication.

The FreeRTOS framework is employed to create a queue for buffering data between the Lidar reading task and the NRF transmission task, ensuring a seamless flow of information. Two tasks are established: one for reading data from the Lidar and another for sending the data via the NRF module. This setup allows for concurrent processing, enhancing the efficiency of data handling.

In the setup() function, serial communication is initialized at a baud rate of 230400, both for the main serial monitor and the Lidar sensor. The Lidar's UART buffer is set to 32KB to accommodate the high-speed data output. The NRF module is initialized with specific configurations, such as disabling automatic acknowledgment of packets, setting the data transmission rate to 2 Mbps, and establishing the transmission power to maximum for extended range. A queue is created to hold the Lidar data, and tasks are created and pinned to specific cores of the ESP32 for optimized performance.

The lidarTask function is responsible for continuously reading data from the Lidar sensor. It checks if data is available and reads it in 128-byte chunks, subsequently adding this data to the queue for later transmission. If the queue is full, the optional flush command clears the buffer to prevent overflow, ensuring that data integrity is maintained.

On the other hand, the nrfTask function retrieves data from the queue and processes it for transmission. The received data buffer is divided into four 32-byte chunks, each of which is copied into the payload structure. These chunks are then transmitted over the NRF24L01 module, allowing for wireless communication with the ground station. The task includes minimal delays to maximize the speed of data sending, thus optimizing the overall system performance.

Finally, the loop() function remains empty, as all functionality is handled by the FreeRTOS tasks. This design choice underscores the efficiency of the task-based architecture, allowing the system to operate smoothly without the need for code within the main loop.

Overall, this code effectively manages the collection and transmission of Lidar data, utilizing the strengths of the ESP32 and FreeRTOS to create a robust solution for real-time environmental mapping. By addressing high-speed data flow and efficient communication, the system is well-equipped for practical applications in various fields, such as agriculture, urban planning, and disaster management.

Receiver side code review:

This code snippet is designed to receive data transmitted from an NRF24L01 module, acting as a receiver on an ESP32 microcontroller. It facilitates wireless communication, allowing the system to handle incoming data and manage connection status effectively.

The code begins by including essential libraries necessary for the operation of the NRF24L01 module. The SPI.h library enables communication with the NRF module over the Serial Peripheral Interface, while the RF24.h and nRF24L01.h libraries provide functions specifically for managing data transmission and reception using the NRF24L01.

Next, specific pin configurations are defined. The CE_PIN is set to 5, and the CSN_PIN is set to 26. These pins are crucial for controlling the NRF module's operations. The code also sets constants for signal management, including INTERVAL_MS_SIGNAL_LOST (1000 milliseconds) to define the time threshold for detecting a lost connection and INTERVAL_MS_SIGNAL_RETRY (250 milliseconds) for retry intervals.

A payload structure is defined to hold the incoming data. This structure contains an array of 32 bytes (s[32]), which matches the payload size of the NRF24L01 module. An instance of this structure, named payload, is created to store the received data. The variable lastSignalMillis is initialized to keep track of the last time data was successfully received.

In the setup() function, serial communication is initiated at a baud rate of 250,000, which is suitable for high-speed data transmission. The NRF24L01 module is then initialized with various configurations. The setAutoAck(false) function disables automatic acknowledgment for received packets, which can simplify the communication process in certain applications. The data transmission rate is set to 2 Mbps using setDataRate(RF24_2MBPS), optimizing the speed of data transfer.

Furthermore, the setPALevel(RF24_PA_MAX) function sets the transmission power to maximum for extended range. The maximum payload size of 32 bytes is defined using setPayloadSize(sizeof(payload)), ensuring that the module can handle the structure correctly. The NRF module is then configured to open a reading pipe with the specified address ("00001"), and it begins listening for incoming data.

The loop() function is where the core functionality occurs. It continually checks for available data from the NRF24L01 module. When data is detected, the code reads the incoming data into the payload structure using radio.read(&payload, sizeof(payload)). The received data is then written to the serial monitor for debugging and verification purposes. The Serial.flush() command is called to ensure that the data is sent out immediately.

The system tracks the current time using millis() to determine if a connection has been lost. If the difference between the current time and lastSignalMillis exceeds INTERVAL_MS_SIGNAL_LOST, the lostConnection() function is triggered to handle the disconnection.

In the lostConnection() function, the system waits for the specified retry interval before continuing. This can be useful for attempting to re-establish a connection or simply allowing some time before the next check for incoming data.

Overall, this code provides a straightforward implementation of an NRF24L01 receiver, managing the reception of wireless data efficiently while monitoring the connection status. The design is suitable for applications that require reliable data transmission and reception, such as remote sensing or telemetry in robotics and drone technology.

Python code of data processing:

This Python code is crafted to facilitate the collection, processing, and visualization of 3D Lidar data transmitted from a drone via serial communication. It employs threading to ensure that the reading and processing of incoming data occur simultaneously, enabling real-time updates to a 3D scatter plot that effectively visualizes the environment.

At the outset, the code initializes serial communication with the drone, establishing a connection on COM9 (may change in different situation and for different computer) at a high baud rate of 250,000. This setup is essential for efficiently handling the rapid stream of data produced by the Lidar sensor. A queue is implemented to temporarily store the incoming data, which allows for asynchronous processing and ensures that the system remains responsive.

To manage the substantial volume of data generated during operation, three deques are utilized—one for each coordinate axis (X, Y, and Z). These deques are configured with a maximum length, allowing the system to maintain a fixed size for data storage. This not only optimizes memory usage but also ensures that the most recent data points are readily available for visualization.

The code specifies critical parameters for the drone's altitude and position, laying the groundwork for accurate 3D mapping. Minimum and maximum distance thresholds are defined to filter out irrelevant data points. The maximum distance is calculated based on the drone's altitude and a specified angle, ensuring that only valid Lidar readings are processed.

A key feature of this implementation is the use of locks to manage access to shared resources across threads, enhancing data integrity. Two threads are created to handle different tasks: the first thread, designated for reading data from the serial port, continuously checks for available data and places it into the queue. The second thread processes this data, looking for specific header bytes that indicate valid Lidar packets. Upon confirming the packet's validity, the processor extracts distance and angle information to compute the 3D coordinates of detected objects.

During data processing, readings are filtered based on distance and angle criteria. Only those within a designated range and within specific angles (from 60° to 120°) are converted into 3D coordinates. The calculated X, Y, and Z values are subsequently added to their respective deques in a thread-safe manner, ensuring that the visualization remains accurate even as new data arrives.

As the main program loop executes, it initializes a 3D scatter plot using Matplotlib. By turning on interactive mode, the program allows for dynamic updates to the plot as new data points are processed. Each time new coordinates are available, the scatter plot is refreshed, and the axes are adjusted to accommodate the new data. This provides a real-time visualization of the Lidar's findings, giving insight into the surrounding environment as the drone operates.

To ensure smooth operation, the program is designed to handle graceful termination. If interrupted, it clears the running state, stops all threads, and closes the serial connection. This ensures that no resources are left dangling and that the program exits cleanly, maintaining system integrity.

In summary, this code presents a sophisticated approach to real-time data collection and visualization from a Lidar-equipped drone. Through the effective use of threading, queues, and deques, it manages the complexities of high-speed data streams while providing an engaging graphical representation of the scanned environment. The design is particularly well-suited for applications in robotics, autonomous navigation, and environmental monitoring, where real-time analysis of spatial data is critical for informed decision-making.

Challenges and Solutions

During the development of the project, several challenges arose:

- **Data Buffer Overflow:** The rapid influx of data from the Lidar sensor led to buffer overflow issues in the ESP32. To address this, a queuing mechanism was implemented, using FreeRTOS queues to manage incoming data effectively and ensure that data is processed at a manageable rate.
- **Wireless Transmission Limitations:** Achieving reliable data transmission over the NRF24L01 module was challenging due to signal interference and range limitations. To mitigate this, the transmission power was optimized, and error-checking mechanisms were incorporated to ensure data integrity.

- **Real-time Processing:** Ensuring that the data was processed in real-time without significant lag was crucial for effective visualization. By utilizing multi-threading in the ESP32 and efficient algorithms for data handling, the project maintained low latency in data processing and visualization

Final Result of the Project

The culmination of our project is the successful development of a Lidar drone system that can efficiently collect and transmit spatial data in real-time. By integrating the YDLIDAR T-mini Plus with dual ESP32 microcontrollers, we created a compact and lightweight drone capable of performing accurate 3D mapping and environmental scanning. The final prototype features:

- **Stability in Flight:** The drone is designed to maintain stable flight while carrying the Lidar sensor, allowing for precise data collection without disruptions caused by weight or balance issues.
- **Real-Time Data Transmission:** Using one ESP32 as a transmitter on the drone, we achieved seamless communication with the second ESP32 positioned at the ground station. This setup enabled the rapid transfer of data for immediate processing and visualization.
- **3D Point Cloud Visualization:** The collected data is processed and visualized in real-time on a computer, allowing users to analyze the environment in three dimensions. The visualization includes the ability to manipulate the point cloud for better understanding and interpretation of the scanned area.

Data Processing

The data obtained from the Lidar sensor was processed to convert raw distance measurements into meaningful 3D coordinates. The processing steps included:

- **Data Collection:** The ESP32 collected distance and angle data from the Lidar sensor, handling data in real-time while managing incoming data streams efficiently.
- **Coordinate Transformation:** The collected polar coordinates (distance and angle) were transformed into Cartesian coordinates (X, Y, Z) using trigonometric functions, allowing for the construction of a 3D point cloud.
- **Filtering and Optimization:** Distance measurements were filtered based on predefined criteria (minimum and maximum distance thresholds) to ensure only relevant data was included in the final visualization. Additionally, the processing algorithms were optimized to handle data efficiently, ensuring that the visualization remained responsive.
- **Visualization:** The processed data was transmitted to a computer, where it was visualized using Python and Matplotlib, resulting in an interactive 3D representation of the scanned environment.

Conclusions Based on Data

Based on the data collected and processed during the project, several conclusions can be drawn:

- **Performance Evaluation:** The Lidar drone demonstrated reliable performance in capturing and transmitting spatial data, achieving a high level of accuracy in 3D mapping. The decision to use the YDLIDAR T-mini Plus proved beneficial, as its compact size and design optimized flight stability and data collection efficiency.
- **Real-Time Capability:** The dual ESP32 architecture allowed for effective multitasking, enabling simultaneous data acquisition and transmission without significant delays. This setup proved essential for real-time applications, making it suitable for various uses in environmental monitoring and mapping.
- **Visualization Effectiveness:** The ability to visualize Lidar data in real-time provided valuable insights into the scanned environment, facilitating immediate analysis and decision-making.

Meeting Project Goals and Technical Task

The project successfully met its defined goals and technical tasks:

- **Goal Achievement:** The primary objective of developing a functional Lidar drone capable of real-time 3D mapping was accomplished. The final product meets the specifications outlined in the project plan and effectively demonstrates the intended capabilities.
- **Technical Task Fulfillment:** The technical tasks, including data acquisition, processing, and visualization, were effectively implemented. The integration of hardware components and software algorithms was executed successfully, leading to a cohesive and efficient system.

In summary, the project has resulted in a successful prototype of a Lidar drone, demonstrating both technical proficiency and innovative application of Lidar technology. The outcomes of the project provide a solid foundation for future developments and potential commercial applications in the field of drone-based mapping and environmental monitoring.

Contribution of Group Members to the Project:

Both team members were equally involved throughout the entire project, engaging in discussions about critical issues and collaboratively developing solutions. This collaborative approach allowed us to make informed decisions and contributed significantly to the project's success. However, effective distribution of work is essential in group projects, as it facilitates rapid and high-quality development.

Luka Pkhaladze:

- **Design and Modeling:** Luka was responsible for designing the visual appearance of the drone's main body and creating all the 3D models referenced in the project. His work ensured that the drone was not only functional but also aesthetically appealing.

- **Flight Controller Management:** He worked extensively with the flight controller and its software, fine-tuning its settings to meet the specifications required for a quality and stable flight. This involved adjusting parameters to optimize the drone's performance in the air.

- **Remote Control and Connectivity:** Luka handled the integration of the drone's remote control and receiver, ensuring a reliable and stable connection essential for safe flight operations. His attention to connectivity was crucial for maintaining control during flights.

- **Pilot Function Implementation:** He played a key role in incorporating the pilot function into our project, marking his first experience working with drones. This contribution not only expanded his skill set but also added a practical dimension to our drone's operational capabilities.

- **Data Processing Code Development:** Luka also wrote the primary structure and main skeleton of the data processing code that handles the information received from the drone. This foundational code is crucial for transforming the raw Lidar data into meaningful 3D coordinates, enabling effective visualization and analysis.

Together, these contributions reflect a strong collaborative effort and a well-distributed workload, ultimately leading to the successful completion of the Lidar drone project.

Luka Balanchivadze:

- **Collaboration on Specifications:** Luka assisted his partner in identifying and refining the final specifications of the drone, contributing to a well-defined project scope and ensuring that all requirements were met effectively.

- **Lidar Sensor Handling**: He actively engaged with the Lidar sensor, exploring various methods and protocols for communication, which facilitated smooth integration into the drone's system.

- **Transmitter Circuit Construction:** Luka built the transmitter circuit by connecting the Lidar and NRF module to the ESP32 microcontroller. He wrote the transmitter code to ensure lossless reception and rapid transmission of data from the Lidar sensor.

- **Receiver Circuit Development:** Luka built the receiver circuit and, with the essential help of his partner, wrote the receiver code. He ensured the flawless reception of data sent from the drone and facilitated its transfer to the computer for live processing.

- **Data Processing Optimization:** Luka improved the data processing code, enhancing its performance to allow for faster real-time processing and minimizing glitches when creating live 3D point cloud images, despite handling large volumes of data.

These contributions highlight Luka's critical role in the project, showcasing his technical skills and dedication to achieving the project's goals.

Conclusion

Main Aspects of the Project

The primary goal of our project was to design, develop, and implement a Lidar-equipped drone capable of real-time 3D mapping and data transmission. The project required integrating a YDLIDAR sensor, ESP32 microcontrollers, and wireless communication modules to create a reliable and efficient system. The drone's design focused on stability, data accuracy, and seamless communication between the airborne Lidar unit and a ground station for live data processing and visualization.

Key components included the physical design and construction of the drone, the implementation of transmitter and receiver circuits, and the development of software for data acquisition, processing, and visualization. This multidisciplinary approach combined hardware engineering, software development, and communication systems to achieve the project's objectives.

Obtained Result

The project successfully met its objectives. We developed a fully functional drone system that can collect spatial data using the Lidar sensor and transmit it in real-time to a ground station. The data is then processed and visualized as a live 3D point cloud, providing valuable insights into the scanned environment.

- The Lidar-equipped drone performed stable flights while maintaining reliable data transmission.
- The receiver circuit at the ground station allowed for real-time data processing, resulting in a smooth and detailed 3D visualization of the surrounding area.
- The use of dual ESP32 microcontrollers enabled parallel processing, ensuring fast and lossless data transmission between the drone and the ground station.

Knowledge and Experience Gained

Throughout the project, we gained significant knowledge and practical experience in several areas:

- **Hardware Design and Integration:** We learned how to design and assemble custom drone parts, including 3D modeling for components such as the battery mount, Lidar mount, and drone legs. Additionally, we became proficient in integrating various sensors and communication modules into a unified system.
- **Microcontroller Programming**: Working with the ESP32 microcontrollers taught us valuable skills in embedded systems programming. We gained experience with multi-threading, real-time data handling, and wireless communication protocols such as NRF24L01.
- **Lidar Technology:** We deepened our understanding of Lidar systems, including how to communicate with sensors, process their data, and convert it into meaningful 3D coordinates for visualization.
- **Wireless Communication:** We developed skills in establishing reliable wireless data transmission over long distances, ensuring efficient communication between the drone and the ground station.
- **Real-Time Data Processing:** We gained expertise in optimizing data processing pipelines to handle large amounts of information in real time, ensuring smooth visualization without glitches.

Future Plans

Looking ahead, there are several potential developments and improvements for this project:

- **Further Miniaturization:** We aim to optimize the drone's design by further reducing its size and weight, making it even more portable and efficient.
- **Extended Range**: Enhancing the wireless communication range will allow the drone to operate in larger environments while maintaining real-time data transmission.
- **Improved Data Processing Algorithms:** We plan to refine the data processing algorithms to further reduce latency and improve the accuracy of the 3D point cloud.
- **Autonomous Flight Capabilities:** Future iterations could integrate GPS-based autonomous flight functionality, enabling the drone to map environments without manual control.
- **Commercialization:** With further improvements, this project could be developed into a low-cost solution for industries requiring 3D mapping, such as agriculture, urban planning, and environmental monitoring.

References

Espressif Systems. (2019). *ESP32 technical reference manual (version 4.2)*. Retrieved from https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

Espressif Systems. (2021). *ESP32 datasheet*. Retrieved from https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

YDLIDAR. (2023). *YDLIDAR T-mini Plus user manual*. Retrieved from https://www.ydlidar.com/Public/upload/files/2024-05-24/YDLIDAR%20T-mini%20Plus%20User%20Manual_V1.0%20(231222)%20.pdf

YDLIDAR. (2023). *YDLIDAR T-mini Plus development manual*. Retrieved from https://www.ydlidar.com/Public/upload/files/2024-05-24/YDLIDAR%20T-mini%20Plus%20Development%20Manual_V1.0%20(231222).pdf

YDLIDAR. (2023). *YDLIDAR T-mini Plus data sheet*. Retrieved from https://www.ydlidar.com/Public/upload/files/2024-05-24/YDLIDAR%20T-mini%20Plus%20Data%20Sheet_V1.1%20(240131).pdf

Betaflight. (2023). *Betaflight firmware: User manual and configuration guide*. Retrieved from https://github.com/betaflight/betaflight/wiki

Video series about Betaflight flight controller of Joshua Bardwell. Retrieved from https://www.youtube.com/watch?v=LkBWRiEGKTI&list=PLwoDb7WF6c8nT4jjsE4VENEmwu9x8zDiE

National Institute of Standards and Technology (NIST). (2020). *Lidar data processing methods for precision mapping*. Retrieved from https://www.nist.gov/lidar-data-processing

Python Software Foundation. (2023). *Python documentation*. Retrieved from https://docs.python.org/3/

Matplotlib Development Team. (2023). *Matplotlib documentation*. Retrieved from https://matplotlib.org/stable/users/index.html

Thank You

We would like to extend our heartfelt gratitude to all those who contributed to the successful completion of this project.

First and foremost, we would like to thank the head of our project, Zviad Sulaberidze, for his invaluable advice and for providing us with various materials that were essential to the development of our drone. His guidance was instrumental in overcoming many challenges throughout the project.

We are also deeply grateful to our lab technician, Davit, for assisting us in manufacturing the drone's body using the CNC machine. His technical expertise and support were crucial in bringing our design to life.

Special thanks go to our coursemates, the ECE21 students, for their continuous help and support during the project. Their collaboration and willingness to assist made a significant difference in our progress.

We would also like to express our appreciation to our lecturer, Guga Vardiashvili, for making the entire process more interesting and comfortable. His engaging teaching style and support fostered a positive learning environment that greatly enhanced our experience.

To everyone who contributed in any way, we truly appreciate your help and encouragement, which made this project a success.

Attachments

Github link, where are all information, codes, picture of 3D models of our project:
https://github.com/lpkha21/lidarDrone