FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Micro drone and charging station for precision pollination

Diogo Assunção Chaves



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Filipe Baptista Neves dos Santos

July 29, 2025

© Diogo Assunção Chaves, 2025

Resumo

O declínio dos polinizadores naturais representa uma ameaça severa à segurança alimentar global e à sustentabilidade agrícola. Para abordar este desafio crítico, esta dissertação explora o design e implementação de um sistema autónomo para polinização de precisão, combinando robótica avançada, inteligência artificial e tecnologias de energia sustentável.

O sistema proposto integra um quadricóptero personalizado baseado no microcontrolador RP2040 com uma estação de acoplamento com carregamento por contacto, formando uma solução auto-sustentável para operações em estufa. O veículo aéreo micro é capaz de realizar tarefas de alta precisão em ambientes dinâmicos, com o seu controlador de voo suportando modos de taxa, ângulo e altitude usando algoritmos embebidos leves. O sistema incorpora sensores avançados incluindo a *Inertial Measurement Unit* e *Light Detection and Ranging* para navegação precisa e perceção do ambiente.

A estação de acoplamento energeticamente eficiente emprega um mecanismo de alinhamento passivo baseado em funil e controlo *Proportional-Integral-Derivative* guiado por *Bluetooth Low Energy* para aterragem autónoma, projetada para suportar fontes de energia sustentáveis e alinhamento automático para operação contínua do drone. Em conjunto, o drone e a estação criam uma solução escalável e adaptável que emula a eficiência dos polinizadores naturais enquanto permite operação totalmente autónoma.

Para o acoplamento autónomo, foram empregues técnicas clássicas de visão computacional como uma solução mais simples e confiável. Estas estratégias de controlo determinístico foram escolhidas pela sua fiabilidade e adequação em hardware com recursos limitados, demonstrando viabilidade prática sobre complexidade teórica.

Este trabalho introduz um sistema de polinização autónomo inovador que permite reduzir a dependência de métodos de polinização manual intensivos em mão-de-obra, e suporta práticas agrícolas ambientalmente resilientes. A investigação estabelece as bases para futura integração de inteligência visual e *Reinforcement Learning*, abrindo caminho para avanços inovadores na automação agrícola que integram robótica, inteligência artificial e tecnologias de energia renovável.

Abstract

The decline of natural pollinators poses a severe threat to global food security and agricultural sustainability. To address this critical challenge, this dissertation explores the design and implementation of an autonomous micro drone system tailored for precision pollination, combining advanced robotics, artificial intelligence, and sustainable energy technologies.

The proposed system integrates a custom quadcopter based on the RP2040 microcontroller with a contact-based charging docking station, forming a self-sustaining solution for greenhouse operations. The micro aerial vehicle is capable of performing high-precision tasks in dynamic environments, with its flight controller supporting rate, angle, and altitude modes using lightweight embedded algorithms. The system incorporates advanced sensors including the Inertial Measurement Unit and Light Detection and Ranging for precise navigation and ambient perception.

The energy-efficient docking station employs a funnel-based passive alignment mechanism and Bluetooth Low Energy-guided Proportional-Integral-Derivative control for autonomous landing, designed to support sustainable power sources and automatic alignment for seamless drone operation. Together, the drone and station create a scalable, adaptable solution that emulates the efficiency of natural pollinators while enabling fully autonomous operation.

For autonomous docking, classical computer vision techniques were employed as a simpler and more reliable solution. These deterministic control strategies were chosen for their reliability and suitability on resource-constrained hardware, demonstrating practical feasibility over theoretical complexity.

This work introduces a novel autonomous pollination system that reduces dependency on labor-intensive manual pollination methods, and supports environmentally resilient agricultural practices. The research lays the groundwork for future integration of visual intelligence and reinforcement learning, while paving the way for innovative advancements in agricultural automation that integrate robotics, artificial intelligence and renewable energy technologies.

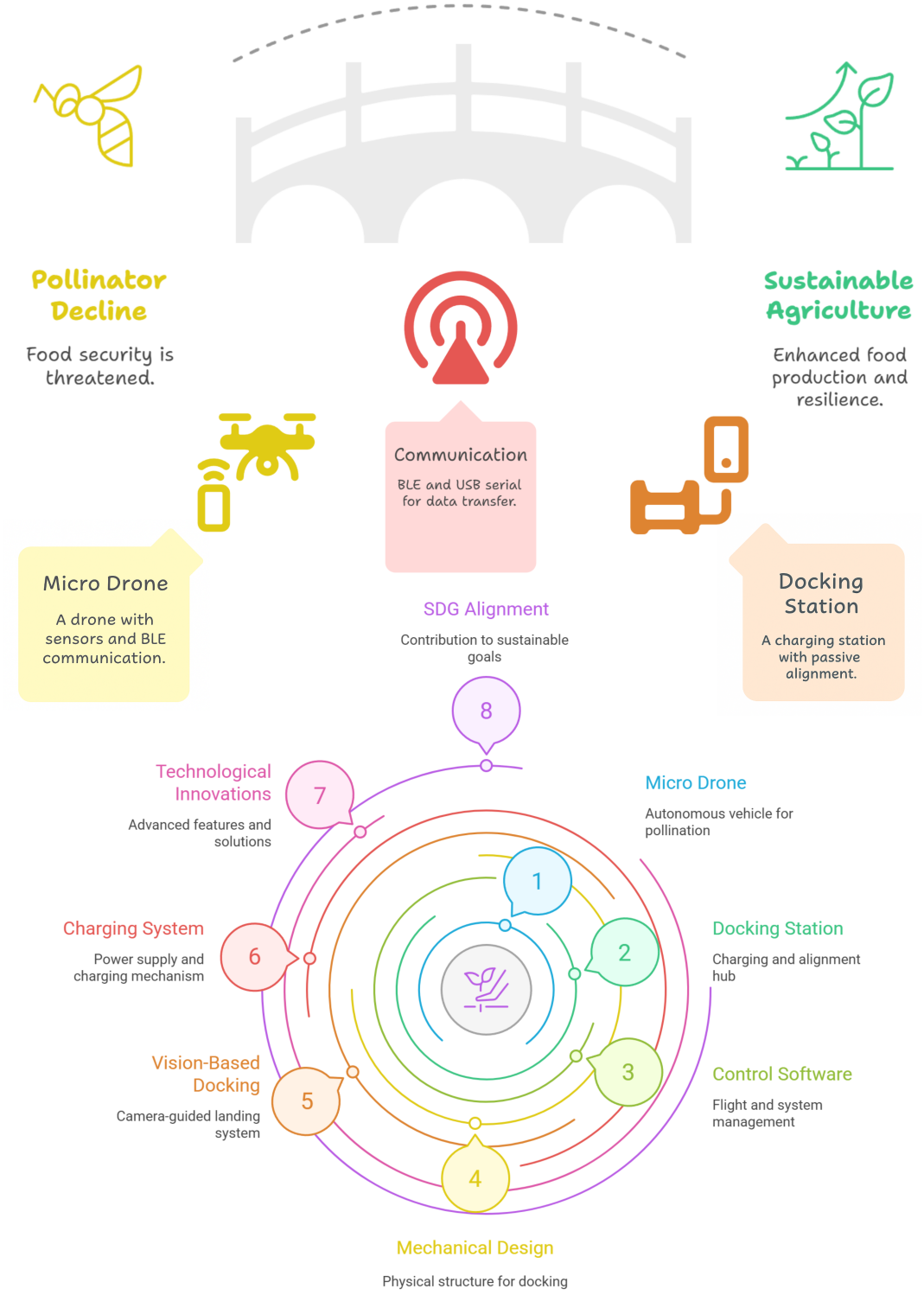


Figure 1: Graphical Abstract

UN Sustainable Development Goals

This dissertation contributes to several of the United Nations Sustainable Development Goals (SDGs) by proposing a novel, sustainable, and technology-driven approach to agricultural pollination using micro drones and autonomous charging stations. Specifically, the project aligns with the following goals:

SDG 2 – Zero Hunger: The proposed system supports sustainable agriculture by addressing the growing issue of pollinator decline, which threatens food security worldwide. By developing a robotic alternative for precision pollination, this work promotes more stable and resilient food production systems.

SDG 9 – Industry, Innovation and Infrastructure: This work contributes to innovation in robotics and automation within the agricultural sector. It promotes the development of sustainable infrastructure through compact, efficient, and scalable systems based on open-source technologies and lightweight autonomous vehicles.

SDG 12 – Responsible Consumption and Production: By integrating low-power electronics, energy-efficient flight controllers, and renewable-energy-compatible docking stations, this project encourages environmentally responsible technological solutions that minimize resource consumption and ecological impact.

SDG 15 – Life on Land: The system aims to reduce reliance on natural pollinators without replacing or harming them, offering a complementary solution that alleviates pressure on ecosystems and helps preserve biodiversity.

SDG	Target	Contribution	Performance Indicators and Metrics
2	2.4	Development of autonomous micro drone systems contributes to sustainable agricultural practices and improves pollination reliability.	Reduction in manual pollination labor; Increase in pollination success rate.
	2.a	Promotes agricultural R&D by integrating robotics and AI to address pollinator shortages.	Research prototypes develop for both the drone and the docking station.
9	9.4	Promotes sustainable industrialization through energy-efficient embedded hardware and modular, replicable designs.	Energy consumption per mission; Drone system under 50 grams and volume under 3.4 dm^3; Charging time under 30 minutes.
12	12.2	Encourages efficient use of materials and energy in autonomous drone systems through lightweight design and battery reuse strategies.	Drone mass under 50 grams; Flight time between 4 and 5 minutes.
15	15.5	Mitigates pressure on declining pollinator populations by offering a non-invasive, technical alternative for targeted crops.	Reduction in artificial pollination aids.

Through the intersection of artificial intelligence, sustainable energy, and autonomous robotics, this dissertation offers an innovative and scalable path forward in supporting environmentally conscious agricultural practices and global sustainability goals.

Agradecimentos

Gostaria de expressar a minha profunda gratidão a todas as pessoas que contribuíram, direta ou indiretamente, para a realização desta dissertação.

Em primeiro lugar, agradeço ao meu orientador, Filipe Neves dos Santos, pelo apoio constante, pela partilha de conhecimento e pela orientação criteriosa ao longo de todas as fases deste trabalho. A sua disponibilidade e entusiasmo foram fundamentais para a concretização deste projeto.

À minha família — em especial aos meus pais e ao meu irmão — agradeço pelo amor, pelo apoio incondicional e pela motivação constante ao longo deste percurso académico. A vossa presença foi essencial nos momentos mais exigentes.

Aos meus colegas e amigos do Mestrado em Engenharia Eletrotécnica e de Computadores, pela companhia, pelas discussões produtivas e pelos momentos de descontração que tornaram esta etapa mais leve e enriquecedora.

Agradeço também aos elementos do *TRIBE Laboratory*, em especial à Isabel Pinheiro, ao Francisco Terra e ao Pedro Moura, que acompanharam mais de perto os desafios enfrentados ao longo do projeto e me ajudaram a encontrar soluções sempre que surgiam obstáculos técnicos.

A toda a equipa e comunidade envolvida no desenvolvimento de plataformas open-source que serviram de base ao trabalho aqui apresentado, pela inspiração, partilha e espírito colaborativo.

Por fim, a todos os que, de alguma forma, contribuíram para a concretização deste projeto, o meu mais sincero agradecimento.

Diogo Assunção Chaves

"Science never solves one problem without raising ten more."

George Bernard Shaw

Contents

Agradecimentos vii

1 Introduction 1

2 Background 3
2.1 Pollination: A Key Process Facing Emerging Challenges and Innovative Solutions 3
2.2 Classification of Unmanned Aerial Vehicles . . . 5
2.3 Basic Components of a UAV . . . 6
2.4 Sensors . . . 9
2.5 Communication . . . 10
2.6 RP2040 Architecture . . . 11
2.7 Artificial Intelligence . . . 13
2.8 Design Tools . . . 13

3 State of the art 15
3.1 Nano Aerial Vehicle Design . . . 15
3.2 Control, Navigation, and Environment Perception . . . 20
3.3 DRL-Based UAV Routing with Charging Stations . . . 22
3.4 Docking Station Design . . . 24
3.5 Challenges and Constraints . . . 29
3.6 Closing Remarks . . . 29

4 Proposed System 31
4.1 System Architecture . . . 32
4.2 Micro-Drone Subsystem . . . 34
4.3 Docking Station Subsystem . . . 35
4.4 Communication Flow . . . 36
4.5 Implementation Status Summary . . . 37
4.6 Design Justifications . . . 37
4.7 Summary . . . 38

5 NAV Hardware and Software 39
5.1 Hardware Architecture . . . 40
5.2 Quadcopter Dynamics . . . 42
5.3 Software Structure . . . 44
5.4 Integration with RP2040_agrolib . . . 44
5.5 BLE UART Communication . . . 45
5.6 Custom IMU Driver: `rp_agrolib_bmi088` . . . 46
5.7 Rate Mode . . . 47

5.8 Angle Mode 50
5.9 Custom LiDAR Driver: `rp_agrolib_vl53l8cx` 53
5.10 Altitude Mode 55
5.11 LEDs Behavior 59
5.12 Flight Results and Mode Evaluation 59
5.13 Telemetry Visualization and Analysis 61
5.14 Challenges, Debugging, and Solutions 62

6 Docking Station **65**
6.1 Mechanical Design 65
6.2 Hardware Architecture 67
6.3 Custom BLE Driver: `rp_agrolib_bc832` 67
6.4 Docking Workflow 69
6.5 Vision-Based Docking Implementation and Challenges 70

7 Conclusion and Future Work **73**

References **77**

A NAB Schematics **79**

B TL.HEART#BLC Schematics **81**

List of Figures

1 Graphical Abstract iv

2.1 Percentage of global food crops reliant on bee pollination, illustrating the scale of dependence on natural pollinators for food security [2] 4
2.2 Overview of major threats to pollinators, including climate change, habitat loss, and pesticide use, contributing to declining populations [3] 4
2.3 Different types of UAV [5] 6
2.4 UAV carbon fiber body frame [6] 7
2.5 Brushless DC motor [6] 7
2.6 Propellers [6] 7
2.7 Electronic speed controllers [6] 8
2.8 ArduPilot APM 2.8 Flight Controller Board [6] 8
2.9 2.4 G 4CH Radio Model RC Transmitter and Receiver [6] 9
2.10 Power Distribution Board [6] 9
2.11 LiPo Batteries [6] 9
2.12 Pollination system configuration using small drones [4] 13

3.1 Example of a MultiWii UAV [10] 15
3.2 Example of an AutoQuad UAV [11] 16
3.3 Example of a LibraPilot UAV [12] 16
3.4 Example of an ArduPilot UAV [13] 16
3.5 Example of a Crazyflie 2.1+ UAV [14] 16
3.6 Example of a MadFlight-equipped UAV [15] 17
3.7 Example of an ArduBee UAV [16] 17
3.8 Madflight V1.3.0 Open-Source Framework [15] 19
3.9 GitHub Star History of Open-Source Flight Controllers [15] 19
3.10 RL framework illustrating the interaction between the agent (drone) and its environment 21
3.11 Classification of Precise Positioning Systems [20] 24
3.12 Conical funnel docking station for precise passive positioning of UAV [21] 26

4.1 Proposed System 31
4.2 UML Component Diagram of the Docking and Drone Subsystems 32

5.1 High Level Architecture 39
5.2 Communication interfaces used in the NAV flight controller, showing I²C and UART peripherals 40
5.3 Visual representation of quadcopter control through differential thrust. Top: Roll (X-axis), Middle: Pitch (Y-axis), Bottom: Yaw (Z-axis). Each maneuver is achieved by varying motor outputs while maintaining overall throttle [26] 43

5.4 Task allocation between Core 0 and Core 1 in the RP2040 microcontroller for drone control . . . 44
5.5 Bluetooth joystick app used to control the NAV . . . 45
5.6 Ranging flow using BMI088 . . . 47
5.7 Feedback control block diagram for rate mode . . . 47
5.8 PID Response[26] . . . 48
5.9 Rate control loop . . . 49
5.10 Feedback control block diagram for stabilize mode . . . 50
5.11 Smartphone interface used to send desired angles via BLE. Left stick controls throttle and yaw; right stick controls pitch and roll. These inputs are used as targets in the outer-loop angle controller . . . 51
5.12 Comparison between raw angle estimation from the accelerometer (red) and the Kalman filter output (blue) while the quadcopter remained stationary on the ground. Motor activity is indicated along the timeline[26] . . . 53
5.13 Ranging flow using VL53L8CX. Adapted from [27] . . . 54
5.14 Feedback control block diagram for velocity mode (altitude hold) . . . 55
5.15 Attitude telemetry plots during manual control . . . 61
5.16 Altitude and vertical velocity over time during manual flight . . . 62
5.17 Drone socket ball joint . . . 63

6.1 Docking Station design . . . 65
6.2 Underside view of the drone . . . 66
6.3 Docking Station PCB design . . . 67
6.4 BLE communication flow . . . 68
6.5 Manual land using smartphone APP Video . . . 70

A.1 Schematic of the microcontroller block . . . 79
A.2 Schematic of the sensorial module block . . . 80
A.3 Schematic of the motor control block . . . 80

B.1 Interfaces module schematic . . . 81
B.2 Solar module schematic . . . 82

List of Tables

2.1 Classification of Unmanned Aerial Vehicles [5] 5

3.1 Comparison of Open-Source Flight Controllers [15] 18
3.2 Overview of commercially available docking station solutions [20] 28

4.1 Communication interfaces between system components - using smartphone . . . 36
4.2 Communication interfaces between system components - using docking station . 36
4.3 Summary of implementation status 37

5.1 Summary of flight mode evaluations with telemetry-based observations 64

List of Acronyms

AI	Artificial Intelligence
BLE	Bluetooth Low Energy
BREMEN	Behavior Regularized Offline Reinforcement Learning
CNN	Convolutional Neural Network
DIY	Do-It-Yourself
DL	Deep Learning
DQN	Deep Q Networks
DRL	Deep Reinforcement Learning
EMF	Electromotive Force
ESC	Electronic Speed Controller
FMCW	Frequency Modulated Continuous Wave
GNSS	Global Navigation Satellite System
GPI	Generalized Policy Updates
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LiDAR	Light Detection and Ranging
MAV	Micro Aerial Vehicle
ML	Machine Learning
MPC	Model Predictive Control
NAV	Nano Aerial Vehicle
PAV	Pico Aerial Vehicle
PCB	Printed Circuit Board
PID	Proportional Integral Derivative
PPO	Proximal Policy Optimization
PWM	Pulse Width Modulation
QR	Quick Response
RADAR	RAdio Detection And Ranging
RL	Reinforcement Learning
SD	Smart Dust
ToF	Time-of-Flight
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
WPT	Wireless Power Transfer
YOLO	You Only Look Once
QVGA	Quarter Video Graphics Array
µUAV	Small Unmanned Aerial Vehicle

Chapter 1

Introduction

Pollination is a vital process within the plant growth cycle, directly influencing the yield and quality of agricultural products. Approximately 75% of global food crops and nearly 90% of wild flowering plants depend on animal pollinators, with bees playing a pivotal role [1]. However, the ongoing decline of natural pollinator populations, exacerbated by climate change, habitat loss, and pesticide use, poses significant risks to global food security. This context highlights the urgency of developing alternative, reliable pollination methods to ensure agricultural productivity and ecosystem stability.

The motivation for this work stems from the urgent need to address pollinator shortages and their impact on global food systems. By leveraging advancements in robotics, AI, and sustainable energy, the proposed system offers a novel approach to support agricultural productivity. The ability of micro drones to perform precision tasks autonomously opens new avenues for large-scale agricultural operations, particularly in regions facing critical pollinator deficits. Additionally, the proposed solution aims to reduce labor costs, chemical dependencies, and ecological vulnerabilities associated with current assisted pollination methods.

The dissertation focuses on the underutilization of micro drones in precision agriculture, particularly in pollination tasks. Although micro drones exhibit high potential for precision and operation in confined environments, the literature reviewed in this work suggests that they remain significantly less explored compared to conventional Unmanned Aerial Vehicles (UAV). The main aim of this work is to develop a micro drone and an autonomous charging station to mimic and augment the role of natural pollinators. This main objective can be divided into three smaller objectives:

1. Study and validate a micro drone hardware for efficient and precise pollination tasks;

2. Study and design the hardware charging docking station; and,

3. Select and adapt drone controllers, from perception to the flight and docking controller, constrained by the drone computational resources.

These goals are addressed with constraints typical of edge computing and embedded systems, including low power consumption, minimal computational resources, and robustness in dynamic, real-world environments.

This dissertation encompasses the full development pipeline, from concept to implementation and testing. It includes:

- A review of relevant literature and existing technologies in micro drone design, AI-based control, and docking mechanisms;
- The complete design and integration of the drone hardware and software stack;
- The design and construction of a docking station with automated positioning features; and,
- Evaluation of performance in real environments, with an emphasis on stability and autonomy.

This document is organized as follows:

- **Background - chapter 2:** Presents foundational knowledge on pollination, UAV classifications, sensor technologies, and embedded systems;
- **State of the Art - chapter 3:** Reviews existing work on micro drone platforms, autonomous navigation, AI-based perception, and docking station technologies;
- **Proposed System - chapter 4:** Describes the overall system architecture and design decisions;
- **NAV Hardware and Software - chapter 5:** Details the development and testing of the drone's embedded systems;
- **Docking Station - chapter 6:** Presents the mechanical and electrical design of the charging station, and its integration with the drone;
- **Conclusions - chapter 7:** Summarizes key findings and outlines directions for future research and improvements.

By the end of this work, a prototype will be presented, reinforcing the viability of autonomous micro drones for precision pollination and laying the groundwork for future improvements and scaling in precision agriculture.

Chapter 2

Background

This chapter provides an overview of the essential role of pollination in agriculture, the challenges faced by natural pollinators, and the technological advancements aiming to address these challenges. Nano Aerial Vehicle (NAV) stand out as a promising innovation, offering precision, adaptability, and a sustainable alternative to traditional methods. With advancements in sensors, flight controllers, and open-source platforms, NAV are well-positioned to revolutionize precision agriculture and ensure global food security.

2.1 Pollination: A Key Process Facing Emerging Challenges and Innovative Solutions

Pollination is a critical process in agriculture, directly influencing the yield and quality of many essential crops worldwide. Approximately 75% of global food crops, including almonds, apples, and berries, rely at least partially on pollinators such as bees [1]. This dependency underscores the importance of pollination for agricultural productivity and global food security. However, this reliance on natural pollinators introduces vulnerabilities in food production, particularly in large-scale farming systems (Fig. 2.1).

The global decline of pollinator populations has raised significant concerns for agriculture. Environmental stressors such as climate change, habitat destruction, and pesticide exposure are primary contributors to this decline. Climate fluctuations disrupt pollinator life cycles, while pesticide use weakens their health and devastates populations [1]. These challenges threaten the natural pollination processes essential for crop production, accelerating the need for innovative artificial pollination solutions (Fig. 2.2).

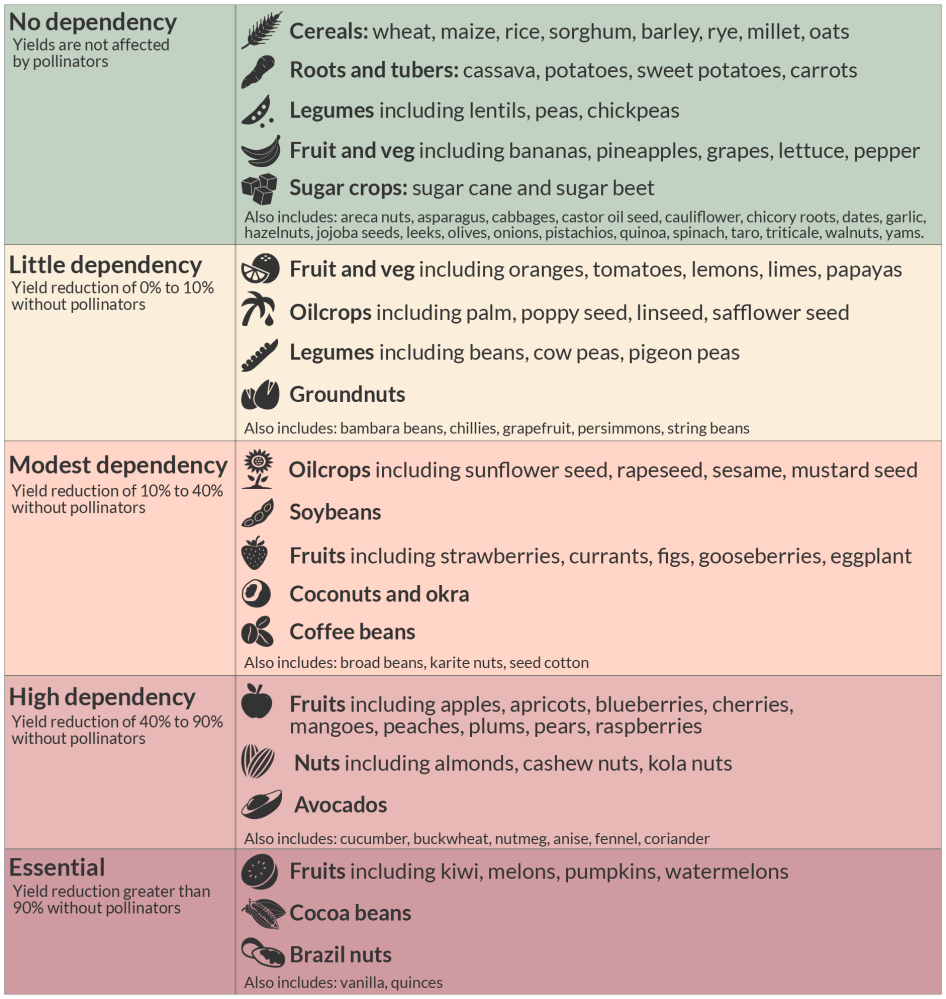


Figure 2.1: Percentage of global food crops reliant on bee pollination, illustrating the scale of dependence on natural pollinators for food security [2]

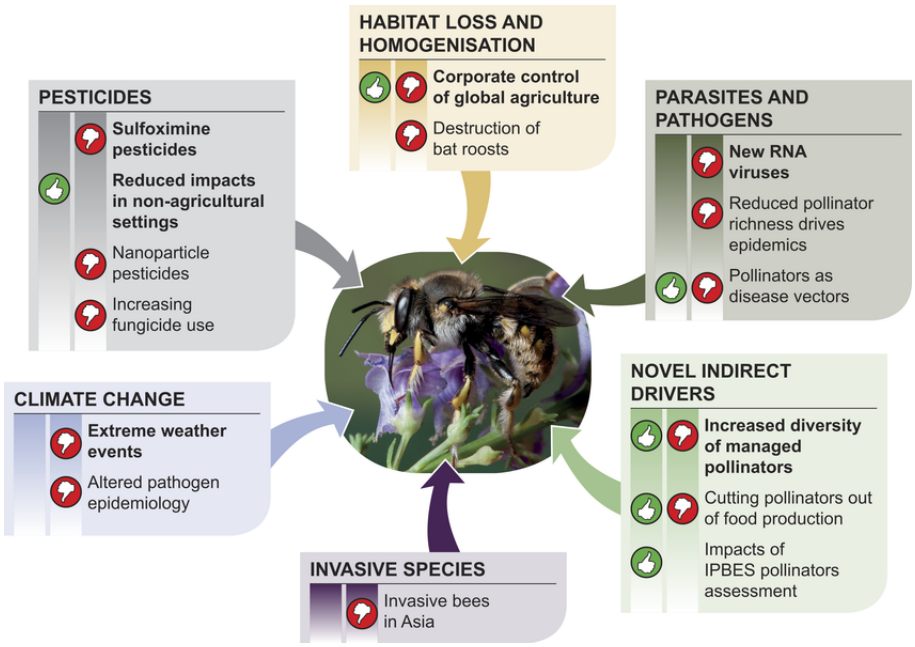


Figure 2.2: Overview of major threats to pollinators, including climate change, habitat loss, and pesticide use, contributing to declining populations [3]

In response to the declining effectiveness of natural pollinators, various artificial pollination methods have been developed. Traditional insect pollination remains effective but is increasingly hindered by environmental pressures. Manual pollination using vibrating tools offers precision but is labor-intensive and costly. Alternatively, synthetic plant hormones can stimulate pollination but risk harming plant quality if misapplied.

Among these alternatives, drone-based pollination has emerged as a promising solution. Drones, particularly NAV, offer precision, efficiency, and scalability for pollination tasks. NAV, defined by their compact size (smaller to fit in a square box of 15 cm) and lightweight design (<50 g), can navigate delicate agricultural environments with minimal disruption. Their ability to perform labor-intensive tasks autonomously reduces labor costs and mitigates risks associated with declining pollinator populations.

Technological advancements in micro drone design have enhanced maneuverability, stability, and power management, enabling NAV to interact with flowers accurately without causing damage. This capability is especially valuable in controlled environments like greenhouses and in challenging field conditions. Despite high initial investment and maintenance requirements, drone pollination presents a sustainable and efficient approach to modern agricultural challenges, offering a viable alternative to natural pollination and assisted pollination methods [4].

By integrating innovative solutions like NAV, agriculture can reduce its dependence on natural pollinators and address the emerging challenges posed by environmental changes and declining insect populations.

2.2 Classification of Unmanned Aerial Vehicles

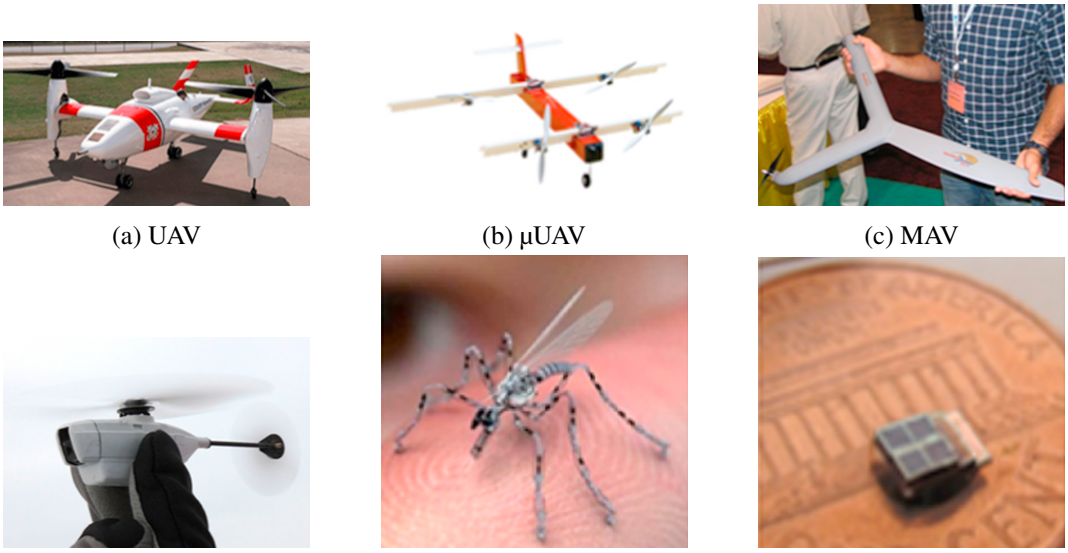
UAV are categorized based on their size, weight, and operational capabilities. These classifications range from large fixed-wing UAV to Nano Aerial Vehicles (NAV) and Smart Dust (SD). NAV, fitting in a square box of 15 cm and weighing less than 50 g, stand out due to their ability to perform precision tasks in confined spaces. Table 2.1 illustrates the classification parameters:

Table 2.1: Classification of Unmanned Aerial Vehicles [5]

Class	**Weight**	**Wingspan**
Unmanned Aerial Vehicle (UAV)	>5 kg	>2 m
Small Unmanned Aerial Vehicle (μUAV)	$\leq$ 5 kg	$\leq$ 2 m
Micro Aerial Vehicle (MAV)	$\leq$ 2 kg	$\leq$ 1 m
Nano Aerial Vehicle (NAV)	$\leq$ 50 g	$\leq$ 15 cm
Pico Aerial Vehicle (PAV)	$\leq$ 3 g	$\leq$ 2.5 cm
Smart Dust (SD)	$\leq$ 0.5 g	$\leq$ 0.25 cm

NAV excel in applications requiring high precision and low environmental impact, making them ideal for agricultural pollination.

Figure 2.3 displays examples of larger unmanned aerial vehicles, spanning from UAV to SD. This range of vehicles demonstrates the diverse scale and capabilities of unmanned aerial technology.



(a) UAV (b) µUAV (c) MAV

(d) NAV (e) PAV (f) SD

Figure 2.3: Different types of UAV [5]

NAV in Precision Agriculture

NAV are increasingly recognized as pivotal tools in precision agriculture. These vehicles are equipped with sophisticated sensors, including LiDAR, GNSS, and vision-based systems, to facilitate autonomous navigation and obstacle avoidance. The integration of AI and machine learning algorithms enhances their capability to execute complex tasks, including efficient pollination strategies and real-time decision-making. Moreover, NAV compact size and minimal environmental footprint make them a sustainable alternative to traditional agricultural methods, aligning with global efforts toward responsible consumption and production.

2.3 Basic Components of a UAV

Certain essential components are required for a UAV to operate effectively. Below is a summary of these components and their functions.

Body Frame

The body frame provides the structural integrity for the UAV and houses the electronic components that connect the motors to the system. It is designed to be lightweight yet spacious enough to allow the propellers to rotate without interference [6].



Figure 2.4: UAV carbon fiber body frame [6]

Brushless DC Motors

Brushless DC motors are responsible for providing the necessary torque to spin the propellers at high speeds. Unlike brushed motors, brushless motors have a high thrust-to-weight ratio but require Electronic Speed Controllers (ESCs). These motors are typically rated by their speed (Kv) and the maximum current they can draw [6].

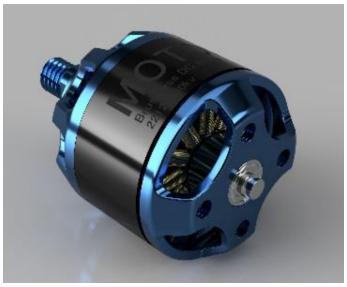


Figure 2.5: Brushless DC motor [6]

Propellers

Propellers are aerofoils that generate lift or thrust when they spin. They are available in various sizes and materials and are usually measured by their diameter and pitch. Correct propeller selection is crucial for achieving the right amount of lift and preventing motor overheating [6].



Figure 2.6: Propellers [6]

Electronic Speed Controllers (ESC)

Each brushless motor requires an ESC to function properly. The ESC interprets the PWM signals from the flight controller and adjusts the current to the motor for the desired speed. ESCs have current ratings that must match the motor's requirements to prevent overheating[6].



Figure 2.7: Electronic speed controllers [6]

Flight Control Board

The flight control board performs the necessary calculations to maintain the UAV's stability. It processes sensor inputs and outputs control signals to the motors. Many boards come with integrated sensors and customized software for easier operation [6].



Figure 2.8: ArduPilot APM 2.8 Flight Controller Board [6]

Transmitter (Radio) and Receiver

The transmitter and receiver facilitate the communication of control signals from the pilot to the UAV. Modern systems typically operate on the 2.4 GHz frequency, though older systems often used 72 MHz [6].



Figure 2.9: 2.4 G 4CH Radio Model RC Transmitter and Receiver [6]

Power Distribution Board (PDB)

The PDB is a printed circuit board that distributes power from the battery to the various components of the UAV. This board ensures that all parts receive the required voltage and current [6].



Figure 2.10: Power Distribution Board [6]

Batteries

Lithium-Polymer (LiPo) batteries are commonly used in UAV due to their lightweight and high energy density. These batteries are rated by their capacity (mAh) and discharge rating (C), which determine how long the UAV can operate and how much current it can provide [6].



Figure 2.11: LiPo Batteries [6]

2.4 Sensors

Given the small size and weight of the NAV, only essential sensors were selected to integrate the solution. Developing an autonomous UAV requires carefully choosing and integrating sensors for tasks such as environmental perception, pose estimation, and obstacle detection. The navigation controller relies on sensor data to carry out and manage missions effectively. In agricultural settings, UAV mainly use vision-based sensors and Light Detection and Ranging (LiDAR) sensors,

along with position-based sensors that deliver accurate location data relative to a known reference point [7].

Vision-Based Sensors

Vision-based sensors produce images of captured scenes, offering a view of the surrounding environment. These images are then processed using algorithms to ensure safe execution of various UAV operations [7]. An example of such a vision sensor is the Quarter VGA (QVGA) camera. QVGA cameras provide a resolution of 320x240 pixels, which offers a balance between image quality and data processing requirements. These cameras are particularly useful for UAV due to their compact size, relatively low power consumption, and moderate cost. Despite their lower resolution compared to higher-end cameras, QVGA cameras are often sufficient for many UAV-based tasks while maintaining efficiency in terms of both hardware and computational resources.

Light Detection And Ranging

LiDAR is an advanced remote sensing technology used to create 3D environmental maps. It operates on the same principle as ultrasonic sensors but uses light instead of sound waves. LiDAR can be combined with GNSS for localization and obstacle avoidance [7].

Inertial Measurement Unit (IMU)

IMU is an electronic device that measures inertial quantities like acceleration, angular motion, and object orientation. These devices are manufactured with diverse features and specifications, and the most suitable choice depends on the UAV's requirements. IMUs work in conjunction with GNSS modules to form the Inertial Navigation System (INS). INS solutions provide essential information for mapping, facilitating navigation, and controlling UAV [7].

2.5 Communication

Efficient communication is essential for the micro drone and charging station. The RP2040 microcontroller supports Inter-Integrated Circuit (I²C), Universal Asynchronous Receiver/Transmitter (UART), and Serial Peripheral Interface (SPI), which are widely used for embedded systems.

Inter-Integrated Circuit

I²C is a synchronous serial communication protocol that provides multi-master, multi-slave communication using a simple two-wire interface. It is widely used for short-distance communication between integrated circuits within a system [8].

Universal Asynchronous Receiver/Transmitter

UART is an asynchronous serial communication protocol that provides full-duplex communication between devices. It is widely used for long-distance, low-speed communication between microcontrollers, computers, and peripherals [8].

Serial Peripheral Interface

SPI is a synchronous serial communication protocol that provides full-duplex communication between a master device and one or more slave devices. It is widely used for short-distance, high-speed communication between microcontrollers and peripherals such as sensors, displays, and memory devices [8].

These protocols enable real-time navigation, data transfer, and system scalability in a low-power environment.

2.6 RP2040 Architecture

The RP2040 is a low-power, high-performance microcontroller developed by Raspberry Pi Foundation. It is designed with flexibility and affordability in mind, making it a strong candidate for compact embedded systems such as UAV. In this project, the raw RP2040 chip in SMD form was directly integrated onto a custom PCB, allowing for greater control over component layout and weight optimization.

Core Architecture and Performance

The RP2040 features a dual-core Arm Cortex-M0+ processor capable of running at up to 133 MHz. This dual-core architecture enables concurrent processing, allowing time-critical control tasks (such as flight stabilization loops) to run independently from less critical functions (like telemetry transmission). The deterministic behavior of Cortex-M0+ cores is especially well-suited for real-time control systems.

Memory and Flash

The microcontroller includes:

- 264 KB of SRAM, organized into six independent banks to optimize parallel access and reduce contention.
- An external QSPI interface for up to 16 MB of flash memory. This memory is mapped into the executable address space using XIP (Execute-In-Place), allowing large firmware binaries without consuming internal RAM.

This memory layout supports robust firmware architectures, including modular drivers, filtering algorithms, and multitasking control logic without exhausting limited internal memory.

I/O and Peripheral Interfaces

The RP2040 supports a rich set of I/O features:

- Up to 30 General Purpose I/O (GPIO) pins (26 usable in most applications)
- 2 × UART, 2 × SPI, and 2 × I^2C interfaces
- 16 PWM channels for motor or actuator control
- 3 ADC channels (12-bit) for analog signal acquisition (e.g., battery voltage monitoring)

This selection of peripherals provides all the functionality needed to interface with critical drone subsystems such as IMU, LiDAR, and sensor inputs.

Programmable I/O (PIO)

A standout feature of the RP2040 is its two Programmable I/O (PIO) blocks, each containing four state machines. These are highly flexible hardware engines that can implement custom protocols (e.g., custom PWM, encoder interfaces, or LED drivers) without burdening the main CPUs. For time-sensitive drone applications, PIO allows reliable signal generation even under heavy CPU load.

Power Efficiency and Thermal Characteristics

The RP2040 is fabricated on a 40 nm process, enabling low active and idle power consumption. With proper firmware design (including the use of sleep and dormant modes), the system achieves efficient power usage — a critical factor in aerial platforms with tight energy budgets.

Development Ecosystem

While the chip itself is minimal and hardware-oriented, it is well supported by an ecosystem of tools and SDKs:

- The official C/C++ SDK with extensive peripheral libraries;
- Third-party support in PlatformIO, Arduino Core, and TinyUSB, and
- Detailed datasheets and community documentation.

These resources accelerate development and debugging while maintaining low-level access to performance-critical features.

The RP2040 provides an ideal balance of processing power, I/O versatility, and low power consumption — all within a compact footprint. Its integration into a custom PCB enabled a lightweight, low-cost, and high-reliability flight controller tailored for the specific requirements of micro-drone navigation and autonomous docking.

2.7 Artificial Intelligence

Artificial Intelligence is essential for enabling drones to autonomously perform complex tasks such as pollination. Machine learning (ML), particularly Deep Learning (DL), forms the backbone of this capability. In agriculture, DL has become increasingly valuable for addressing challenges like object detection and pattern recognition. Deep Neural Networks (DNN) have evolved in their performance and the variety of tasks they can manage. However, excelling in a specific task does not automatically equate to being deployable in real-world scenarios. Battery-powered devices, including edge cameras, robots, and drones, face stringent power consumption constraints. Consequently, research has increasingly focused on developing energy-efficient networks to address these challenges [9].

Figure 2.12 illustrates a pollination system configuration using small drones, where photography aids in locating flowers, followed by AI-driven position estimation, image processing, and precise pollination actions.

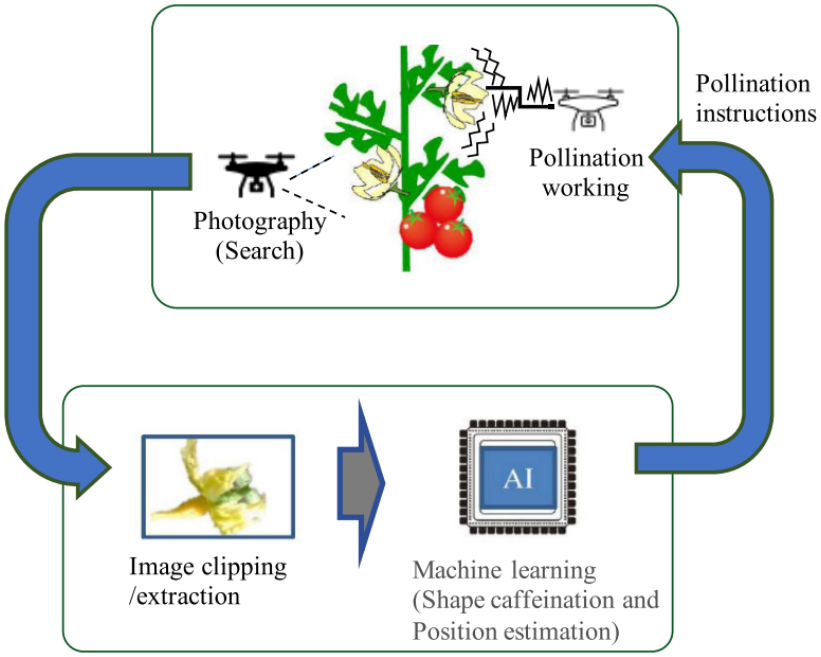


Figure 2.12: Pollination system configuration using small drones [4]

2.8 Design Tools

To support the development of the docking station and drone modifications, two specialized software tools were employed:

EasyEDA: an online-based PCB design platform that was used for the schematic capture and layout of the docking station's electronics. It provided simulation tools, component libraries, and Gerber export capabilities suitable for manufacturing.

Autodesk Fusion 360: used to design all 3D-printed mechanical parts, including the docking station enclosure, drone leg extensions, camera module adapters, and a custom drone support used during Proportional Integral Derivative (PID) controller tuning.

These tools enabled a streamlined workflow from concept to physical implementation and ensured design consistency across both mechanical and electronic components.

Chapter 3

State of the art

3.1 Nano Aerial Vehicle Design

To design an efficient and precise micro drone for pollination, it is crucial to analyze existing solutions in NAV designs. Lightweight materials like carbon fiber and reinforced plastics are widely used for durability and minimal weight. Propulsion systems typically utilize brushless DC motors for higher efficiency and reduced noise [5]. The RP2040 microcontroller is an optimal choice due to its dual-core processing capabilities and low power consumption, enabling real-time control and efficient integration with sensors. Advanced vision sensors, such as QVGA cameras, are employed for object detection and navigation while balancing processing demands.

Open-Source Software Platforms

The development of UAV, particularly NAV, has been significantly enhanced by open-source software platforms. These platforms provide modular and customizable solutions for flight control, navigation, and data processing.

- **MultiWii**: An open-source project inspired by Nintendo Wii sensors, offering flight stabilization and control capabilities [10].



Figure 3.1: Example of a MultiWii UAV [10]

- **AutoQuad**: AutoQuad is an Open Source firmware project with closed source hardware. It is aimed at providing a flight controller board with stabilization, dynamic flight and autopilot features [11].



Figure 3.2: Example of an AutoQuad UAV [11]

- **LibraPilot**: LibrePilot is a software suite for controlling multicopter models. This project aims to develop software and hardware capable of being used in various applications, such as UAV. This project has been built on top of the OpenPilot project [12].



Figure 3.3: Example of a LibraPilot UAV [12]

- **ArduPilot**: A versatile platform supporting a wide range of unmanned vehicles with robust community support [13].



Figure 3.4: Example of an ArduPilot UAV [13]

- **Crazyflie 2.1+**: A lightweight, modular NAV platform equipped with advanced sensors for autonomous navigation [14].



Figure 3.5: Example of a Crazyflie 2.1+ UAV [14]

- **MadFlight**: An advanced drone flight controller designed for precision and adaptability, offering cutting-edge features for various UAV applications [15].



Figure 3.6: Example of a MadFlight-equipped UAV [15]

- **ArduBee**: An innovative platform tailored for small UAV, integrating sensors for obstacle avoidance and precise navigation [16].



Figure 3.7: Example of an ArduBee UAV [16]

These platforms enable researchers and developers to experiment and innovate in UAV design, fostering advancements in areas like autonomous pollination and precision agriculture.

Among the platforms presented, MadFlight demonstrates the greatest potential for our needs. Beyond being open-source, it offers firmware that is already compatible with the RP2040 microcontroller, streamlining integration with the drone's hardware and reducing development time. This compatibility makes MadFlight a more practical and efficient choice compared to other platforms that may require extensive customization.

Flight Controller Open-Source Reference

Comparison of Open-Source Flight Controllers

Table 3.1 summarizes various open-source flight controllers, ranked in increasing order of complexity. MadFlight, in particular, is gaining traction due to its modularity and support for diverse microcontrollers.

Flight Controller	**Features**	**Environment**	**Microcontroller**
Drone-flight-controller	Single 700-line file, no libs	Arduino	ATmega328P
dRehmFlight	Quad, Plane, VTOL	Arduino Teensy 4	Teensy 4
MadFlight	Quad, Plane, VTOL, modular design	ESP-IDF	ESP32, RP2040, STM32
ESP-FC	FPV Quad	PlatformIO	ESP32
Crazyflie	FPV Quad, based on Crazyflie	Custom	STM32F405
ESP-Drone	FPV Quad, based on Crazyflie	ESP-IDF	ESP32
Betaflight	FPV Quad, based on Cleanflight	Custom	STM32 F4/F7/H7
iNav	Plane, based on Cleanflight	Custom	STM32 F4/F7/H7
ArduPilot	Quad, Plane, VTOL	Linux Waf	STM32 F4/F7/H7 or Linux
PX4-Autopilot	Quad, Plane, VTOL	Custom	STM32 F4/F7/H7

Table 3.1: Comparison of Open-Source Flight Controllers [15]

Low-complexity solutions, such as Drone-flight-controller and dRehmFlight, are suitable for educational or constrained embedded applications. They typically consist of a minimal codebase that fits within older 8-bit or 32-bit microcontrollers like the ATmega328P or Teensy. However, their limited feature set and scalability render them impractical for advanced navigation, sensor fusion, or modular expansion required in autonomous micro-drone operations.

On the other end of the spectrum, full-stack systems such as ArduPilot and PX4-Autopilot offer highly robust capabilities, including GPS integration, advanced autopilots, obstacle avoidance, and complex mission planning. These frameworks are best suited for larger UAV platforms with more powerful STM32F7 or Linux-based flight computers. While these systems are mature and reliable, their complexity and resource demands make them unsuitable for sub-50g drones constrained by tight computational and power budgets.

Mid-tier platforms, including Crazyflie, ESP-FC, and iNav, target FPV-style quads and small autonomous UAVs. They provide support for basic autonomy, telemetry, and stability control, but often rely on specific hardware and may lack the modularity required for unconventional applications, such as pollination-specific missions or docking workflows.

MadFlight, by contrast, provides a compelling compromise. It supports a wide range of vehicles (Quad, Plane, VTOL), is fully open-source, and is architected with a modular firmware design. Importantly, it is one of the few platforms that natively supports the RP2040 microcontroller — a key selection criterion in this project.

In summary, while high-end platforms like PX4 provide comprehensive features and low-end controllers offer simplicity, MadFlight occupies a unique space — offering extensibility and multi-platform support while remaining lightweight enough for resource-constrained hardware like the RP2040. This makes it an ideal foundation for prototyping an autonomous micro-drone for precision pollination, where modularity, size, and power efficiency are critical.

Madflight v1.3.0

Madflight v1.3.0 is an open-source flight controller framework designed for building high-performance UAV using affordable hardware components [15]. It supports various microcontrollers, including ESP32-S3, ESP32, RP2040, and STM32, offering flexible and cost-effective drone development.



Figure 3.8: Madflight V1.3.0 Open-Source Framework [15]

GitHub Star History

The GitHub star history highlights MadFlight's steady growth since 2024, emphasizing its increasing adoption within the UAV development community. Although it currently trails behind larger frameworks like PX4-Autopilot and ArduPilot in terms of total stars, MadFlight is poised to become a major player in the ecosystem due to its focus on accessibility and innovation. Figure 3.9 shows the history of the GitHub star for various flight controllers.

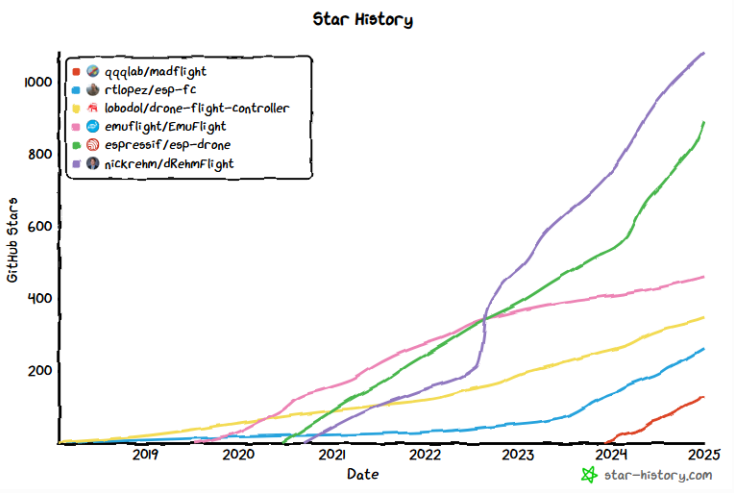


Figure 3.9: GitHub Star History of Open-Source Flight Controllers [15]

Madflight Flight Controller Overview

Madflight is a modular and open-source flight controller designed for high-performance UAV applications. Its adaptable architecture allows developers to customize and expand its features to meet specific project needs. The platform supports seamless integration with various sensors, including IMUs, barometers, and GNSS modules, ensuring accurate flight control and navigation. Madflight offers multiple selectable flight modes, such as RATE and ANGLE, to accommodate different pilot skills and preferences. Additionally, it provides telemetry data and blackbox logging for performance monitoring and issue diagnosis. The active open-source community behind Madflight fosters continuous updates and innovation.

An example of Madflight's implementation is the Quadcopter Advanced setup, which includes a Raspberry Pi Pico (RP2040) microcontroller, MPU6500 gyro/accelerometer, BME280 barometer, INA226 current sensor, and a uBlox M8 GNSS with QMC5883L compass. It also integrates an ExpressLRS (ELRS) radio receiver and a DJI E300 propulsion system, all mounted on an Ideafly IFLY-4 quadcopter frame. This configuration demonstrates how Madflight can efficiently utilize affordable and accessible hardware to build a fully functional flight controller for less than $10 [15].

3.2 Control, Navigation, and Environment Perception

Control and navigation in resource-constrained drones require efficient algorithms. Reinforcement Learning (RL) has shown promising results in enabling drones to autonomously adapt to dynamic environments [17]. Lightweight models like ESPNets enable real-time object detection with limited computational resources [9]. Sensor fusion techniques combining IMU, LiDAR, and vision-based inputs enhance localization and obstacle avoidance [7].

Deep Learning

DL facilitates autonomous pollination by enabling drones to perceive and interact with their environment. A key concept in this domain is the world model, which serves as an internal simulator for intelligent systems. These models are developed through extensive learning from sensorimotor data, allowing drones to predict and adapt to complex agricultural environments. Techniques such as the Joint Multimodal Variational Autoencoder (JMVAE) effectively handle scenarios where certain sensory data might be missing, ensuring robust object detection and decision-making capabilities[17].

Reinforcement Learning

RL complements DL by refining drone behavior for efficient navigation and precise flower targeting, emulating natural pollinators. Algorithms like Behavior Regularized Offline Reinforcement Learning (BREMEN) optimize drone policies using offline data, reducing the need for extensive

real-world data collection. This method leverages learned dynamics models, allowing drones to interact with simulated environments and improve decision-making through iterative policy updates [17].

Generalized Policy Updates (GPI) further enhance adaptability by enabling drones to combine multiple pre-learned policies to tackle new pollination tasks. Successor feature representations allow drones to efficiently generalize from past experiences, facilitating rapid adaptation to varying environmental conditions without the need for exhaustive retraining. This capability is crucial for scaling autonomous pollination solutions in large-scale agricultural operations.

Figure 3.10 shows the RL framework where the drone (agent) interacts with its environment by performing actions and receiving feedback in the form of perceptions and rewards, optimizing its behavior over time.



Figure 3.10: RL framework illustrating the interaction between the agent (drone) and its environment

RL has gained significant attention in robotics and control applications, particularly in autonomous aerial vehicle navigation and landing procedures. Traditional model-based control strategies, such as Model Predictive Control (MPC) and observer-based controllers, require precise system modeling and extensive engineering effort. In contrast, RL provides a model-free approach that enables autonomous agents to learn optimal policies through interactions with their environment [18].

Deep Reinforcement Learning for UAV Control

Deep Reinforcement Learning (DRL), an advanced form of RL, utilizes deep neural networks to approximate value functions or policies, making it suitable for high-dimensional state-action spaces. DRL techniques, such as Deep Q Networks (DQN) and Proximal Policy Optimization (PPO), have been effectively applied to UAV docking and landing tasks, improving robustness in uncertain environments [18].

Deep Q Networks

DQN is a value-based method that estimates the optimal action-value function using deep neural networks. It has been employed in UAV docking tasks, where it learns discrete control actions to minimize landing errors. However, DQN faces challenges such as overestimation bias and sample inefficiency, which can be mitigated using Double DQN and Dueling DQN architectures [18].

Proximal Policy Optimization

PPO is a policy-based method that directly learns the optimal control policy. PPO has demonstrated superior performance in UAV docking by minimizing final impact velocity and improving sample efficiency compared to DQN-based methods [18].

Sim-to-Real Transfer and Domain Randomization

One of the critical challenges in applying DRL to real-world UAV operations is the sim-to-real transfer problem. Policies trained in simulations often fail in real-world scenarios due to discrepancies between simulated and real environments. A promising approach to address this issue is domain randomization, where environmental parameters, such as wind disturbances and wave movements, are varied during training to enhance generalization [18].

In offshore docking applications, the Joint North Sea Wave Project spectrum model has been used to generate realistic wave disturbances during DRL training. This technique has been shown to improve UAV landing success rates in uncertain maritime conditions [18].

Comparative Performance of DRL Agents

Recent studies comparing PPO and DQN-based agents for UAV docking indicate that PPO achieves[18]:

- Lower final impact velocity, ensuring a smoother and safer landing;
- Higher sample efficiency, requiring fewer training episodes to converge to an optimal policy; and,
- More robust generalization, due to its stochastic policy formulation and the use of domain randomization.

Conversely, DQN-based agents, while effective, suffer from higher sensitivity to hyperparameters and longer convergence times, making them less suitable for real-time docking applications [18].

3.3 DRL-Based UAV Routing with Charging Stations

Battery constraints are a major limitation in UAV operations, especially for tasks requiring extended flight durations, such as autonomous docking. Recent advancements in wireless charging technologies and optimized routing algorithms have improved UAV mission endurance [19].

Multi-Head Heterogeneous Attention for UAV Routing

A key challenge in UAV navigation is efficient path planning that considers energy consumption while minimizing travel distance. DRL has been successfully applied to this problem using an encoder-decoder network with a multi-head heterogeneous attention mechanism, which learns to prioritize targets and charging stations dynamically [19].

The Markov Decision Process formulation of the UAV Routing Problem with Multiple Charging Stations ensures that policies are learned based on real-world constraints, improving the UAV's ability to make decisions under uncertain conditions [19].

Curriculum Learning for Generalization

To enhance the robustness of DRL policies, curriculum learning strategies have been implemented. These gradually increase the difficulty of training tasks, improving the generalization of DRL agents across different environments, including offshore docking applications [19].

Comparison with Traditional Heuristics

Compared to classical optimization techniques like Simulated Annealing and Genetic Algorithms, DRL-based methods have demonstrated superior performance, achieving:

- Shorter total travel distances, reducing energy consumption;
- More adaptive decision-making, allowing UAVs to autonomously recharge when necessary; and,
- Faster computational performance, making DRL-based solutions more scalable than traditional heuristics [19].

Docking Guidance

A landing system encompasses the mechanisms necessary to guide a UAV to its docking station, ensure accurate landing, and secure the UAV in place. While guidance is often managed by the UAV itself, there are situations where external assistance is required [20].

Accurate positioning during the landing process is critical for achieving precise landings. This involves maintaining relative positioning between the UAV and the docking station. The landing process is typically divided into distinct phases.

The first phase, known as the approach phase, often utilizes GNSS to navigate the UAV toward the docking station. The subsequent phase involves fine-tuning the UAV's position to ensure minimal landing errors. This phase is essential for a successful touchdown, which marks the final phase of landing. UAV are frequently guided visually using markers placed on the docking station, although other methods may also be employed. Figure 3.11 categorizes various precise positioning systems [20].

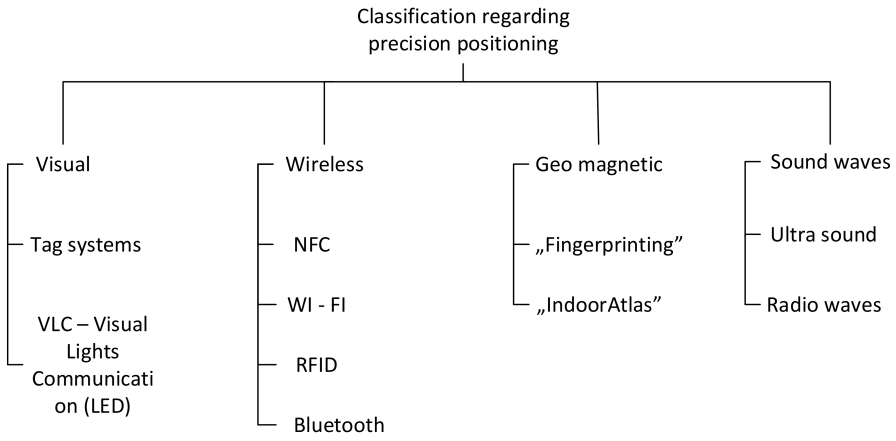


Figure 3.11: Classification of Precise Positioning Systems [20]

Common visual markers include ArUco markers, QR codes, or custom-designed markers. Alternative methods for precise guidance, such as infrared LEDs or radio waves, are particularly useful in low-visibility conditions. Numerous studies on docking stations emphasize the importance of precise positioning for UAV [20].

3.4 Docking Station Design

UAV docking stations are versatile systems that provide multiple functions for unmanned aerial vehicles. These include facilitating safe landings and takeoffs, replenishing or swapping batteries, and enabling data and payload transfers. Some advanced stations offer storage capabilities, protecting UAV from harsh weather conditions. To minimize human involvement, these processes are designed to operate autonomously. Consequently, docking stations extend a drone's operational time by eliminating the need for manual battery changes by an operator [20]. Docking stations can be classified regarding:

- Mobility (mobile and fixed);
- Charging method (two electrodes, multiple electrodes, wireless, etc.);
- Automatic battery exchange (spare batteries recharging);
- Positioning (active and passive);
- Drone storage (yes or no);
- Package delivery (storage, no storage);
- Type of landing (precision, vision-based, etc.); and,
- Type of landing platform (self-leveling).

A UAV docking station incorporates various components, including a platform for landing, a system for positioning, electronic elements, a power unit, visual aids for landing, mechanisms for recharging and swapping batteries, storage facilities for drones and items, and additional subsystems. To be effective, these docking stations for unmanned aerial vehicles must satisfy specific requirements [20].

Landing Platform

Apart from the guidance in air, the landing involves accurate positioning of the UAV upon contact with the docking station. A detailed classification of related mechanisms is provided in [21]. These mechanisms are broadly categorized as either active or passive.

Active mechanisms employ actuated systems to adjust the UAV's position, whereas passive systems rely on the design of landing pads to ensure proper alignment. Landing platforms can be classified based on several factors:

- **Positioning:** Active or passive;
- **Orientation:** Self-leveling, fixed, or tilted;
- **Power:** Charging electrodes or wireless charging; and,
- **Vision:** Incorporation of visual tags or lighting systems.

Fixed Docking Stations

This segment provides a concise overview of stationary docking stations like the one that will be developed in this project. Although these stations can be relocated, they remain fixed during operation. Some may be portable, typically smaller in size, but must be stationary when in use. A significant portion of research focuses on software solutions for landing guidance, with precise landings being a primary concern in docking station design. The visual approach, utilizing onboard cameras and markers on docking stations, is a prominent method, particularly suitable for NAV equipped with cameras, as it eliminates the need for additional sensors. This approach can be enhanced through various controllers such as Fuzzy Logic, MPC, and Deep neural networks. While some of these enhancements have minimal impact on the flight controller, others can be computationally demanding, requiring careful selection. Alternative methods for accurate landings on fixed docking stations incorporate different sensor types [20].

Fixed docking stations may include platforms with active or passive positioning systems, or simple landing pads without positioning systems that offer wireless battery charging. Wireless charging is an effective way to maintain UAV power for long routes or to charge UAV without additional equipment on the docking station. Although wireless charging is the most robust solution, it can be slow and requires precise UAV positioning for optimal performance. To address these wireless charging limitations, charging with electrodes is being explored [20].

Fixed Docking Station with Passive Positioning Example

Fixed docking stations with passive positioning mechanisms are designed to simplify the UAV landing process while ensuring accurate alignment. These systems typically rely on gravity to guide the UAV into position, eliminating the need for active mechanisms. One such example is a docking station with a conical funnel designed to accommodate all UAV legs. This design enables precise alignment as the UAV's vertical motion is transformed into horizontal positioning through interaction with the inclined walls of the funnel [21].

The conical funnel ensures that UAV legs slide into place and settle in a predefined position, making it suitable for UAV with low payload mounts. Such systems are effective for operations where simplicity and robustness are critical, particularly in stable environmental conditions. Figure 3.12 illustrates the design of a passive docking station with a conical funnel, which consists of a flat surface 21 and a set of centering funnels 22, embedded in this surface. The funnels 22 are arranged coaxially to the UAV legs and consist of conical 22A and cylindrical 22B parts. The UAV legs slide and move to a given position on the conical parts 22A and fall into the cylindrical parts 22B, where electrical contacts may be located for charging [21].

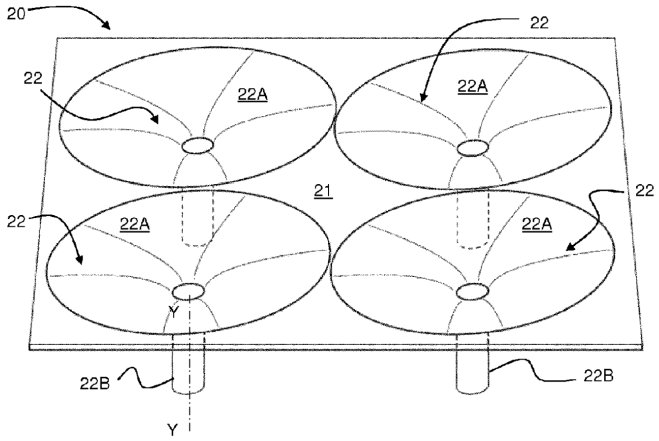


Figure 3.12: Conical funnel docking station for precise passive positioning of UAV [21]

For our project involving the docking station for the NAV, the most suitable solution is a fixed docking platform with passive positioning that aligns upon contact with the drone's legs. This approach offers several advantages: it eliminates the need for complex actuated systems, reducing mechanical complexity and overall weight. Passive positioning ensures reliability and durability by minimizing moving parts that could wear out over time. Additionally, the fixed design simplifies the landing process, allowing the NAV to align naturally with the docking station, leading to consistent and secure landings. This setup also supports efficient power transfer, particularly when integrating charging electrodes, ensuring seamless operation and reduced maintenance.

Compact Lighthouse-Based Docking Station

Recent work by Martin [22] proposes a compact docking station tailored for sub-150g UAVs operating in GNSS-denied indoor environments. The design utilizes a vertically mounted HTC Vive Lighthouse V2 sensor for high-accuracy localization (±2.5–5 cm) and integrates a small actuatable landing pad. The docking protocol leverages Bitcraze's open-source hardware and software, including the Lighthouse deck and EKF-based position estimation, to ensure robust alignment and landing.

This system successfully demonstrates autonomous precision landings using a downward-facing IR sensor array and motor-actuated supports that fold away to maintain IR visibility. Compared to other indoor positioning solutions (e.g., Loco Positioning, OptiTrack), the Lighthouse system balances portability, precision, and deployment simplicity, offering a feasible path toward foolproof indoor drone operation without requiring pilot training or extensive infrastructure.

Commercial Docking Stations

The market offers a wide array of commercial docking stations, spanning from comprehensive systems with specially designed UAV to platforms compatible with off-the-shelf UAV, including DJI drones. These products are predominantly employed in military and security applications, though some are also available for civilian use [20]. A detailed summary of selected docking station solutions is presented in the table 3.2, where + represents an implemented feature. Essential characteristics such as charging method, number of battery slots, time required for swapping, ability to exchange payloads, and the use of custom UAV are emphasized. For conciseness, storage systems and data exchange capabilities, which are common to all the featured docking stations, are excluded from the table.

Battery Management System

The primary purpose of docking stations is often to replenish the UAV's power supply. For electric UAV, this can be achieved through two main methods: recharging the battery or swapping it entirely. Most docking stations are equipped with charging mechanisms that utilize electrical contacts, typically located on the UAV's landing gear. Wireless Power Transfer (WPT) systems are also a viable alternative [20] but not fast enough for our prototype that needs to charge in less than 15 minutes.

Battery swapping involves removing the depleted battery and replacing it with a fully charged one. Ensuring the safe handling and storage of batteries is crucial during these operations to maintain operational safety [20].

Data Transfer

In certain scenarios, UAV must communicate with ground stations between flights. Docking stations equipped with communication systems enable the UAV to connect with the ground station

Table 3.2: Overview of commercially available docking station solutions [20]

Brand Name	**Setup**	**Charging Type**	**Battery Slots**	**Swap Time**	**Payload Exchange**	**Custom UAV**
Airobotics	Drone + Docking Station	Swap	/	/	+	+
Nightingale Security	Drone + Docking Station + AI Software	Contact Charging	-	-	-	+
Percepto	Drone + Docking Station	Charging	-	-	-	+
Hextronics	Battery Replacement Station	Swap	6	2 min	-	+
Skycharge Skyport	Docking Station	Contact Charging	-	-	-	-
HIVE	Docking Station	Swap	6	230 s	-	-
Dronehub	Drone + Docking Station + AI Software	Swap	/	2 min	-	+
Easy Aerial	Drone + Docking Station	Charging	-	-	-	+

for various purposes. These include assigning new flight plans, downloading logs from previous flights, or transmitting data to remote locations [20].

In conclusion, the key features of the docking station include:

- **Portable and Modular Design:** Constructed from lightweight, durable materials, the station will be easy to deploy and relocate across agricultural fields. Its modular design supports the addition of renewable energy sources, such as solar panels, for off-grid operation;
- **Efficient Charging System:** The docking station will feature a USB-C compatible charging circuit, with options for solar and battery backup to ensure functionality in remote locations; and,
- **Autonomous Docking Guidance:** Low-power beacons (infrared or RF) will guide the drone back to the docking station, minimizing power consumption during return flights.

3.5 Challenges and Constraints

As classified in Chapter 2, NAV are required to weigh less than 50 grams and smaller to fit in a square box of 15 centimeters. These constraints arise from the strict size and weight limitations inherent to their design. The reviewed literature highlights the following challenges associated with NAV:

- Limited flight endurance [23];
- High susceptibility to aerodynamic influences, including air currents, drift, and turbulence [23];
- Structural asymmetries and inconsistencies that impact performance [23]; and,
- Fragility and safety-critical operational constraints [24].

The short flight duration primarily stems from restrictions on battery weight and size, as batteries are typically used to power all onboard electronics.

Due to their fragile and safety-critical nature, NAV employ AI to mitigate these vulnerabilities. Algorithms developed using a combination of real and simulated data improve their robustness. This approach often requires equipping NAV with specialized cameras, such as a monocular camera [24] or an ultra-low-power grayscale QVGA camera [25].

To address these limitations, potential solutions include integrating energy-efficient components and leveraging AI to enhance system robustness. Strategies such as optimizing flight paths or incorporating renewable power sources will also be explored to mitigate these challenges.

3.6 Closing Remarks

The current state of the art in micro drone technology and autonomous docking illustrates significant progress toward robust, low-weight UAV systems capable of performing precise, autonomous

tasks such as pollination. Open-source platforms like Bitcraze have enabled flexible, low-cost development. Reinforcement learning and AI frameworks are increasingly applied to optimize flight control, environment interaction, and docking maneuvers.

Furthermore, recent innovations in docking station design reveal a growing focus on compact, indoor-compatible, and sensor-integrated systems. Notably, Martin [22] demonstrated a highly portable docking solution using HTC Vive's Lighthouse system, offering centimeter-scale precision without the need for GNSS or complex motion capture setups. Such work shows the potential for streamlined UAV deployment in controlled agricultural or industrial environments.

Collectively, these developments highlight both the promise and the remaining challenges of fully autonomous drone operation. Precise localization, lightweight design, and onboard processing constraints remain central concerns. However, the integration of open-source tools, advanced control strategies, and compact infrastructure solutions positions the field to meet the demands of real-world, scalable applications such as automated precision pollination.

Chapter 4

Proposed System

This chapter presents the architecture of the proposed autonomous micro-drone system for precision pollination in greenhouse environments. The system integrates a custom-built NAV with an energy-efficient docking station for recharging and future autonomous landing capabilities. The design emphasizes modularity, resource efficiency, and real-time responsiveness.

While the docking station's charging functionality has been fully implemented and tested, the autonomous docking subsystem based on computer vision remains under development. The drone's flight controller has been custom-designed and validated through manual flight using a Bluetooth Low Energy (BLE) interface and telemetry analysis. This chapter serves as a high-level overview of the entire system, providing context for the detailed subsystems described in Chapters 5 and 6.



Figure 4.1: Proposed System

4.1 System Architecture

The system comprises two main components:

- **NAV**: A lightweight quadcopter equipped with a custom flight controller, onboard sensors, and BLE communication.
- **Docking Station**: A fixed platform designed for passive mechanical alignment, contact-based charging, and vision-based autonomous landing.

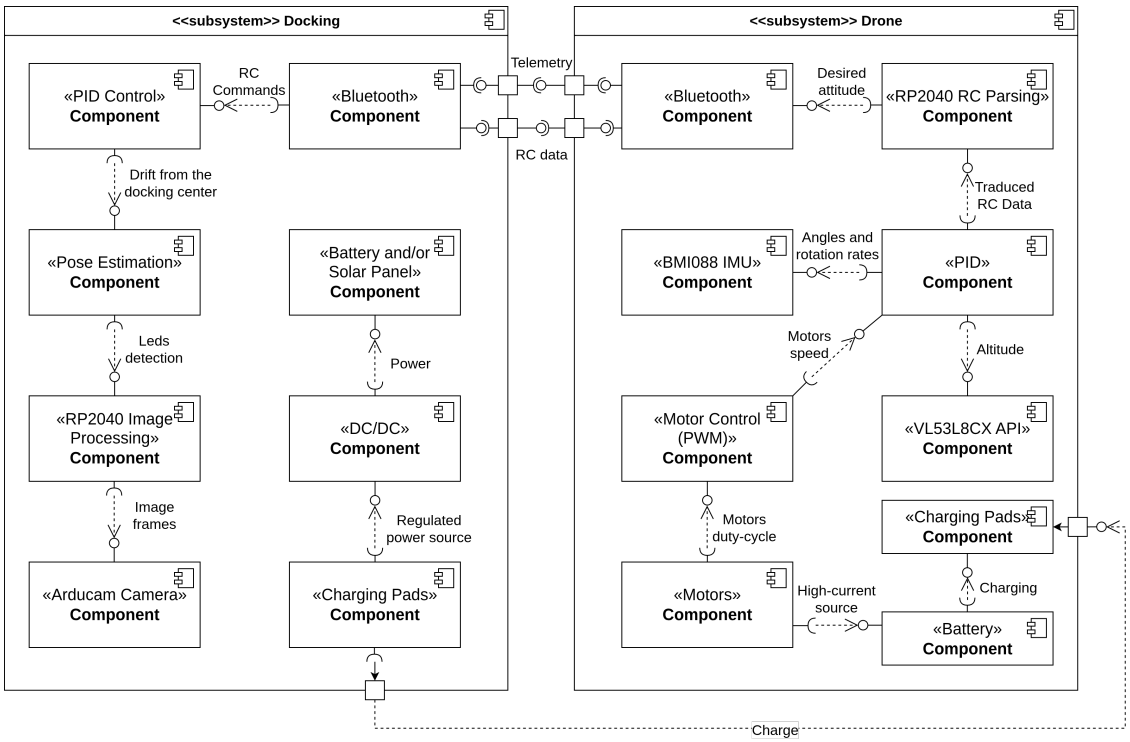


Figure 4.2: UML Component Diagram of the Docking and Drone Subsystems

The UML component diagram in Figure 4.2 illustrates the functional breakdown and interaction between the two major subsystems: the Docking Station and the Drone (NAV). Each component represents a physical or logical module responsible for a specific role within the system architecture.

Docking Station Subsystem

The docking station is responsible for autonomous landing support, power delivery, and wireless communication with the drone. Its components are as follows:

- **Arducam Camera**: Captures image frames used for detecting visual landing markers (e.g., LEDs on the drone).

- **RP2040 Image Processing**: Processes frames from the camera and extracts features necessary for visual pose estimation.
- **Pose Estimation Component**: Computes the relative position and orientation of the drone from visual cues.
- **PID Control**: Uses the estimated pose to generate corrective RC commands for guiding the drone toward a central docking position.
- **Bluetooth Component**: Handles wireless communication with the drone, transmitting control signals and receiving telemetry data.
- **Battery and/or Solar Panel**: Supplies power to the docking system, optionally sourced from renewable energy.
- **DC/DC Converter**: Regulates power to the required levels for the camera, controller, and charging system.
- **Charging Pads**: Provide contact-based energy transfer to the drone's onboard battery upon landing.

Drone (NAV) Subsystem

The drone subsystem is centered around a custom RP2040-based flight controller that handles attitude and altitude stabilization, navigation, and telemetry:

- **RP2040 RC Parsing**: Receives and interprets RC commands from the docking station over BLE.
- **Bluetooth Component**: Facilitates bi-directional communication with the docking station.
- **BMI088 IMU**: Provides real-time measurements of acceleration and angular velocity for flight stabilization.
- **PID Component**: Generates appropriate control responses based on sensor feedback and RC targets.
- **Motor Control (PWM)**: Converts PID outputs into PWM signals to regulate motor speeds.
- **Motors Component**: Propels the drone based on motor speed adjustments.
- **VL53L8CX API**: Interfaces with a LiDAR sensor used for altitude hold.
- **Charging Pads**: Receive power through contact with the docking station's charging terminals.
- **Battery Component**: Stores electrical energy for flight operations.

Subsystem Interaction

- During the docking sequence, the camera on the station detects the LEDs on the drone.
- Image data is processed to estimate the drone's position relative to the docking center.
- PID control algorithms generate corrective actions, sent as RC commands over BLE.
- The drone applies these commands via its onboard control system and adjusts its attitude accordingly.
- Once docked, charging pads establish electrical contact, and the drone begins recharging.
- Continuous telemetry feedback ensures precise final alignment and enables diagnostic monitoring.

Design Considerations

This modular architecture allows for ease of debugging, future system upgrades (e.g., adding camera-based perception on the drone), and low power operation. The use of RP2040 in both subsystems ensures hardware compatibility, development consistency, and deterministic control behavior essential for autonomous docking and flight stability.

4.2 Micro-Drone Subsystem

Hardware Components

The micro-drone was designed using a minimal but effective sensor and control suite tailored for precision tasks in constrained environments. Each hardware component was selected based on size, weight, interface compatibility, and functionality. Below is a detailed breakdown:

- **RP2040 Microcontroller**: A dual-core Arm Cortex-M0+ MCU running at up to 133 MHz, used as the central flight controller. It manages sensor data acquisition, PID control loops, BLE communication, and PWM signal generation. Its low power consumption and real-time capabilities make it ideal for resource-constrained aerial vehicles.
- **BMI088 IMU (I^2C)**: A high-performance 6-DoF Inertial Measurement Unit by Bosch Sensortec, combining a 3-axis accelerometer and a 3-axis gyroscope. It offers low drift and vibration robustness, making it suitable for flight stabilization. Communication is performed via the I^2C bus, and sensor data is fused using a Kalman filter for orientation estimation.
- **VL53L8CX Time-of-Flight (ToF) LiDAR (I^2C)**: A multi-zone, direct-ToF ranging sensor from STMicroelectronics capable of measuring distances up to 400 cm with millimeter resolution. Unlike earlier single-zone ToF sensors, the VL53L8CX supports a resolution of up

to 64 zones (8×8 array), enabling low-resolution depth mapping. In this project, it is configured in a reduced zone mode (4×4) to balance data throughput and computational efficiency. It is primarily used for altitude estimation and can support future collision avoidance tasks.

- **BC832 BLE Module (UART)**: A compact Bluetooth Low Energy module based on the Nordic nRF52832 SoC. It provides UART-based wireless communication between the drone and docking station (or a smartphone during manual testing). It transmits control commands and streams telemetry (e.g., roll, pitch, altitude) at 5 Hz.
- **4 PWM-Controlled Motors via MOSFETs**: The propulsion system consists of four brushed DC motors controlled via low-resistance MOSFET drivers. The RP2040 generates PWM signals, which are filtered and amplified to drive the motors. Duty cycles are modulated based on PID output to achieve stability across roll, pitch, and yaw axes.

Flight Controller Features

The NAV supports three flight control modes:

- **Rate Mode**: Angular velocity control using gyroscope data.
- **Angle Mode**: Attitude stabilization via a complementary Kalman filter.
- **Altitude Hold**: Velocity-based Z-axis control using LIDAR.

A custom BLE joystick application was used for tuning and manual control during flight tests.

4.3 Docking Station Subsystem

Implemented Features

- contact-based charging circuit
- 3D-printed funnel design for passive drone alignment
- BLE transmission of RC commands from docking to the drone
- Two-way BLE UART communication for command and telemetry exchange, and
- 5 Hz telemetry stream from drone to docking station with roll, pitch, velocity, motor PWM, and altitude

Planned Features

- Upward-facing Arducam for LED detection
- Pose estimation and PID-based command generation

Docking Control Logic (Planned)

```
while (true) {
    frame = capture_frame();
    led_positions = detect_leds(frame);
    pose = estimate_pose(led_positions);
    rc_cmd = compute_rc(pose, lidar_data);
    send_ble_command(rc_cmd);
}
```

4.4 Communication Flow

The next tables describe the communication flow, depending on the way the drone is controlled.

With smartphone APP:

Direction	Source	Target	Purpose
BLE UART	Smartphone App	NAV	Manual control input

Table 4.1: Communication interfaces between system components - using smartphone

With docking station:

Direction	Source	Target	Purpose
USB Serial	Python(Keyboard)	Docking Station	RC command input via USB serial
BLE UART	Docking Station	NAV	Real-time RC control over BLE
BLE UART	NAV	Docking Station	5 Hz telemetry stream
USB Serial	Docking Station	PC	CSV telemetry

Table 4.2: Communication interfaces between system components - using docking station

4.5 Implementation Status Summary

Component	Status
Custom Flight Controller	✓**Implemented**
Altitude Hold Mode	✓**Functional**
BLE Joystick Communication	✓**Functional**
IMU + LIDAR Integration	✓**Functional**
Two-way BLE UART (RC + Telemetry)	✓**Functional**
Charging Circuit	✓**Operational**
Passive Docking Funnel	✓**Built & Tested**
Upward Camera Vision System	◯ Planned
Pose Estimation Pipeline	◯ Planned
PID-Based Autonomous Docking	◯ Planned

Table 4.3: Summary of implementation status

Although the main flight and docking functionalities were successfully implemented, three features remain in the planned stage due to hardware-level limitations encountered during development.

The Upward Camera Vision System could not be completed because the RP2040 was unable to retrieve image frames from the Arducam camera, despite the camera passing both I^2C and SPI interface tests. The device responded correctly to register reads and initialization commands, which confirmed that the physical connection and basic communication protocols were functioning. However, attempts to acquire image data — either via FIFO reads or DMA transfers — consistently failed to return valid frame content.

This suggests the problem lies beyond basic bus-level connectivity, potentially involving incorrect register sequencing, timing mismatches, or incomplete driver support on the RP2040 platform. Extensive debugging and attempts to resolve the issue were made but ultimately limited by time and tool constraints. Without a working camera input, it was not possible to proceed with the implementation of the Pose Estimation Pipeline or the PID-Based Autonomous Docking routine, which both rely on real-time visual feedback.

These features are therefore postponed to a future iteration, possibly involving a co-processor with mature camera driver support or offloading image capture to a more capable platform.

4.6 Design Justifications

Due to the simplicity and controlled environment of the docking process, classical computer vision techniques were selected over machine learning. The use of color thresholding and blob detection for LED tracking is lightweight and efficient for embedded systems. PID-based control ensures predictable and interpretable drone behavior.

Additionally, a custom firmware stack was developed to provide greater flexibility and compatibility than existing open-source flight controllers, aligning better with the project's hardware and low-level integration needs.

4.7 Summary

This chapter has presented the full system architecture and outlined both implemented and conceptual components of the NAV and its docking station. While autonomous visual docking remains a future enhancement, the foundation has been laid through successful development of the custom flight controller, BLE communication, and charging hardware. The next chapters explore each subsystem in detail.

Chapter 5

NAV Hardware and Software

This chapter presents the hardware and software implementation of the NAV flight control system. A custom flight controller was developed, replacing the initial Madflight controller due to sensor incompatibilities and difficult to adapt a full software architecture and components. The final architecture was based heavily on control theory and implementation strategies from the Carbon Aeronautics Quadcopter Manual[26]. Key flight modes such as rate mode, angle mode, and velocity (altitude hold) mode are described in detail. This implementation allowed for successful manual flight using a BLE-enabled smartphone interface. The NAV is based on the RP2040 microcontroller and integrates essential sensors and communication modules to enable autonomous flight. The foundation of the hardware is the Nano Aerial Bee v1.2 (NAB) printed circuit board, which serves as the structural and electrical base for the system.

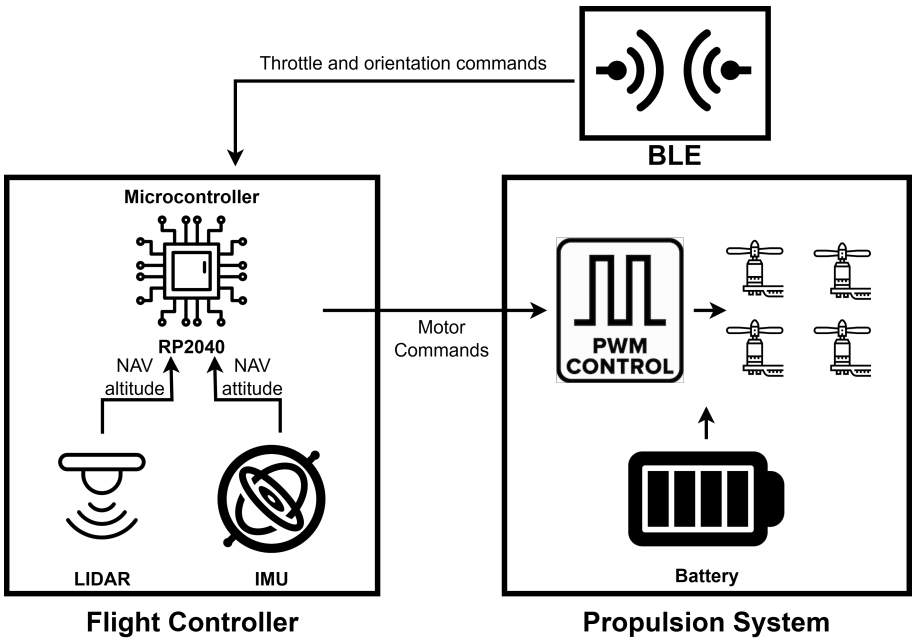


Figure 5.1: High Level Architecture

5.1 Hardware Architecture

The NAV hardware is structured into functional blocks, each responsible for a specific role in the system. The primary components are detailed below and can be observed in detail in Appendix A.

Central Processing Unit

The RP2040 microcontroller was chosen due to its dual-core architecture, lower power consumption, and reduced physical footprint, making it more suitable for NAV applications. The RP2040 enables real-time processing of flight control algorithms and sensor data. It supports up to 16 MB of external flash memory, enabling execution-in-place functionality. This allows for a more flexible memory management system compared to microcontrollers with fixed internal flash sizes.

Sensors

The NAV is equipped with multiple sensors that aid in navigation and environmental perception:

- **BMI088 (I²C):** An IMU providing 3-axis accelerometer and 3-axis gyroscope data for stabilization and navigation;
- **VL53L8CX (I²C):** A ToF sensor used for altitude measurement and obstacle detection; and,
- **BC832 (UART):** BLE module enabling wireless communication.

The IMU and ToF sensors are critical for autonomous flight, ensuring the NAV maintains stability and accurately measures distances.

Communication Interfaces

The NAV employs multiple communication protocols for data exchange:

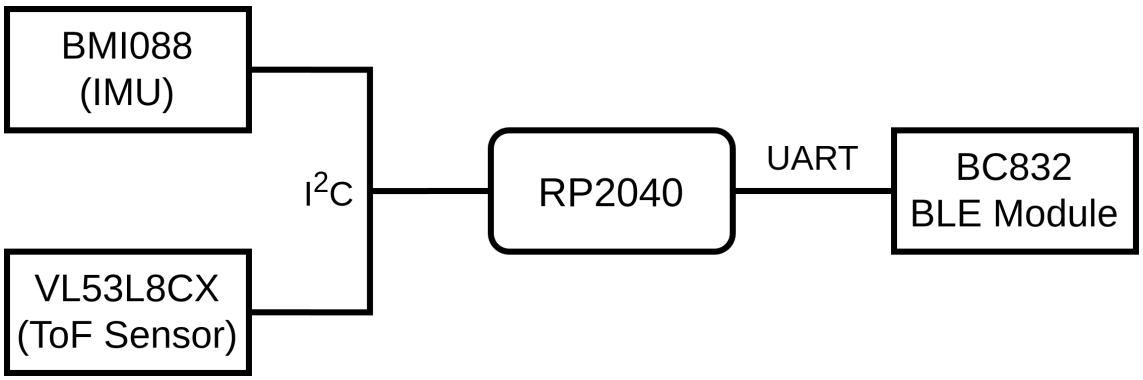


Figure 5.2: Communication interfaces used in the NAV flight controller, showing I²C and UART peripherals

- **I²C:** Used for interfacing with the BMI088 and VL53L8CX sensors;

- **UART:** Used for communication with the BC832 BLE module. A correction was made to the PCB to enable the transmission of AT commands to configure BLE communication; and,
- **SPI:** Used for high-speed data transfer with external flash memory, enabling fast execution of software modules.

The NAV's communication system was designed to be minimal and low power, yet effective, reducing unnecessary connections while ensuring all modules remain responsive during flight.

Power System

The NAV is powered by a lightweight LiPo battery, managed by a power regulation circuit that ensures stable voltage levels for all components. The BQ24075 power management IC enables simultaneous charging and operation. Instead of a low-dropout regulator, the system now incorporates the TPS25750 for USB-C Power Delivery (PD) capabilities, along with the TLV75533 and NCP612 for 5V/3.3V voltage regulation and power protection.

Power efficiency is a key concern in NAV design. The voltage regulation was analyzed considering the consumption of individual components, ensuring that peak power demand does not exceed available supply. The combination of the TPS25750, TLV75533, and NCP612 was selected to provide robust power management while ensuring efficient voltage conversion and protection.

Microcontroller Block

The heart of the NAV is the RP2040 microcontroller, responsible for executing flight control algorithms and managing sensor inputs. Figure A.1 details its schematic.

Sensorial Module

Navigation and stability are critical for the NAV's operation. The integration of IMU and ToF sensors provides essential data for autonomous flight. Figure A.2 illustrates the sensor module schematic.

Motor Control

The NAV includes four main motors for quadcopter control. The motors are controlled using MOSFET drivers, which regulate current flow based on PWM signals from the RP2040.

The MOSFETs used in motor control were analyzed to ensure they handle high-speed switching with minimal heat dissipation. To protect against back-emf, diodes are placed in parallel with each motor.

The NAV's propulsion system enables precise maneuverability. Figure A.3 presents the motor control schematic.

The quadcopter is configured in an X-layout, with motors arranged as follows:

- M1: Front-right (CCW)
- M2: Back-right (CW)
- M3: Back-left (CCW)
- M4: Front-left (CW)

This configuration ensures balanced torque and efficient yaw control through differential thrust.

Several 3D-printed components were developed to assist in the integration and testing of the NAV system. Using Autodesk Fusion 360, parts such as leg extensions and a custom support base for PID tuning were designed and fabricated. These additions facilitated safe tuning procedures and enhanced ground clearance during test flights.

5.2 Quadcopter Dynamics

A quadcopter uses four fixed-pitch propellers to generate lift and control movement in three-dimensional space. These four motors are arranged in an "X" configuration, with pairs of motors spinning in opposite directions to allow for control over roll, pitch, and yaw.

In conventional systems, motor speed is typically controlled using Electronic Speed Controllers (ESCs), which interpret standard PWM signals to drive brushless motors. However, in the custom platform developed for this project, ESCs are replaced by a direct motor control strategy using logic-level MOSFETs. PWM signals are generated by the RP2040 microcontroller and used to switch the MOSFETs, which in turn modulate the power delivered to the motors directly from the battery. This architecture simplifies the hardware and allows for precise low-level control over motor actuation, though it also increases the demand for accurate timing and power handling in software and circuit design.

To achieve stable flight, the drone must manage its motion across six degrees of freedom: translation along the x, y, and z axes, and rotation (roll, pitch, yaw) about those same axes. The dynamics of control are achieved through differential thrust:

- **Throttle (altitude):** Increasing power to all four motors uniformly generates upward lift, allowing the quadcopter to ascend. Reducing it results in descent.
- **Roll (rotation around x-axis):** Adjusting the power on left and right motors causes lateral tilting.
- **Pitch (rotation around y-axis):** Varying the front and rear motor speeds produces forward or backward tilt.
- **Yaw (rotation around z-axis):** Achieved by leveraging torque differences between clockwise and counterclockwise rotating motors.

Figure 5.3 illustrates how individual motor speeds are modulated to produce roll, pitch, and yaw using differential thrust.

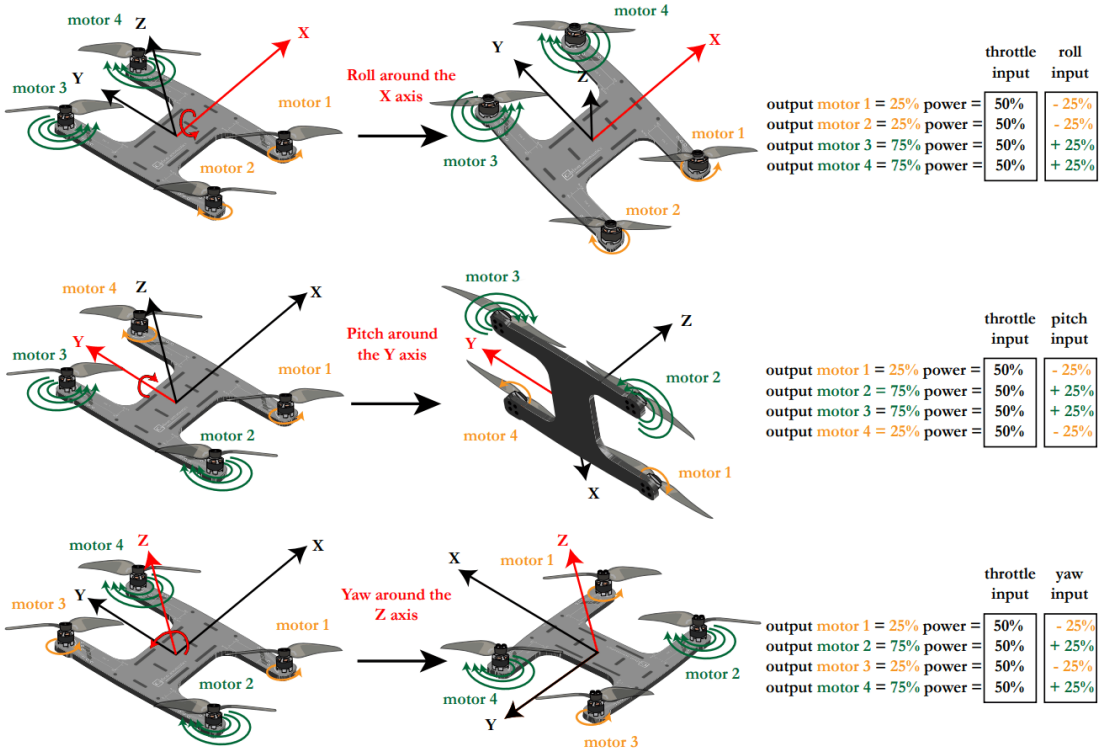


Figure 5.3: Visual representation of quadcopter control through differential thrust. Top: Roll (X-axis), Middle: Pitch (Y-axis), Bottom: Yaw (Z-axis). Each maneuver is achieved by varying motor outputs while maintaining overall throttle [26]

Each motor's thrust is modulated via a PWM signal with a duty cycle proportional to the desired output. In this system, the PWM resolution and frequency are critical for smooth and responsive control. Typically, PWM signals range from 0% to 100% duty cycle, where motor output begins at a certain minimum threshold due to static friction and the electrical characteristics of the motor and FETs. For precise attitude control, the throttle range is often constrained to reserve headroom for applying differential thrust during aggressive maneuvers.

Pilot commands received over Bluetooth are translated into desired angular velocity targets. These setpoints are compared with real-time sensor feedback—primarily from a gyroscope and accelerometer—and a PID control loop computes the required PWM duty cycle adjustments for each motor. This feedback mechanism compensates for external disturbances and non-idealities in the drone's construction, such as uneven mass distribution or aerodynamic asymmetries.

This direct-drive control model allows for low-latency actuation and deep integration between the control algorithm and motor response, but it also requires careful consideration of current handling, thermal dissipation, and power stability, especially when scaling thrust for aggressive flight profiles.

5.3 Software Structure

The drone flight software is built in a modular way using a custom firmware file, `flight.cpp`, written for the RP2040 microcontroller. It is based on the Pico SDK and uses the `rp2040_agrolib` library for sensor interfacing and motor control.

The RP2040 features dual-core support, which is leveraged to split tasks efficiently:

- **Core 0** handles time-sensitive flight control, sensor fusion, and motor actuation.
- **Core 1** handles UART reception from the BLE module and command parsing.

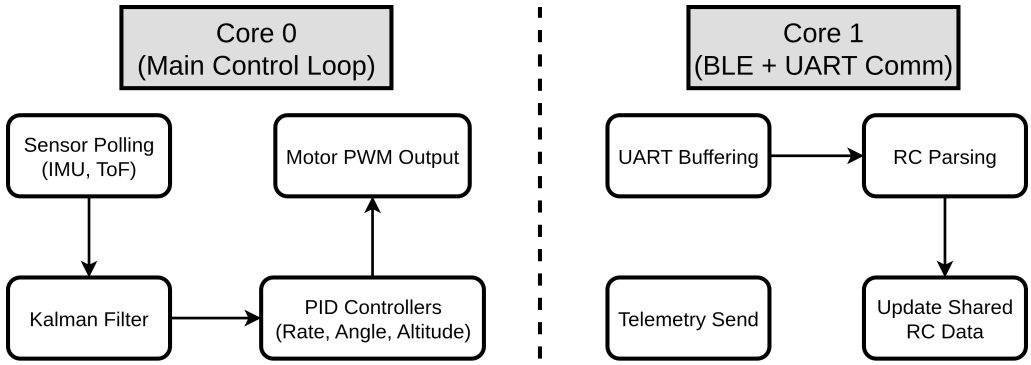


Figure 5.4: Task allocation between Core 0 and Core 1 in the RP2040 microcontroller for drone control

This task separation improves control stability by preventing sensor polling and PID calculations from being interrupted by UART communication or BLE latency.

5.4 Integration with RP2040_agrolib

The firmware relies on a modular and reusable hardware abstraction library named RP2040_agrolib, developed and maintained by the TRIBE Laboratory. This internal GitLab-hosted repository provides lightweight drivers for various sensors and peripherals commonly used in embedded robotics and agricultural automation.

By isolating low-level device interactions into well-defined modules, the library simplifies application development and improves maintainability. Each hardware component is handled through a dedicated API, ensuring that the control logic remains portable and decoupled from vendor-specific implementations.

The key modules used in this project include:

- **rp_agrolib_i2c**: Wrapper for I^2C communication used by sensors like BMI088 (IMU) and VL53L8CX (LiDAR).
- **rp_agrolib_bmi088**: Interface for configuring and reading acceleration and angular velocity from the IMU.

- **rp_agrolib_vl53l8cx**: Handles initialization, calibration, and range measurement using the ToF LiDAR.
- **rp_agrolib_uart**: Provides UART communication routines, used here for BLE-based telemetry and RC control.
- **rp_agrolib_motors**: Manages PWM signal generation to control the speed of each of the four motors.

This structured approach promotes code reuse across different RP2040-based projects within the lab and ensures faster development cycles, especially when integrating common sensor modules.

5.5 BLE UART Communication

Bluetooth Low Energy (BLE) is used to transmit real-time flight control commands from a smartphone app or docking to the drone. The BLE module communicates with the RP2040 over a UART interface and uses a lightweight protocol for sending control values.

BLE Control Interface

Control signals are transmitted via a custom Android joystick app, shown in figure 5.5, that sends commands in the format:

```
RC,roll,pitch,throttle,yaw\n
```

These values are centered around 1500 (neutral position) with a throttle base of: 0 for angle mode and 1500 for velocity mode. A parsing module reads the messages, validates them, and updates the controller input. A deadband is applied near the center of each axis to prevent unintentional movement. This structure mimics standard RC receiver output and simplifies integration with the flight control system.

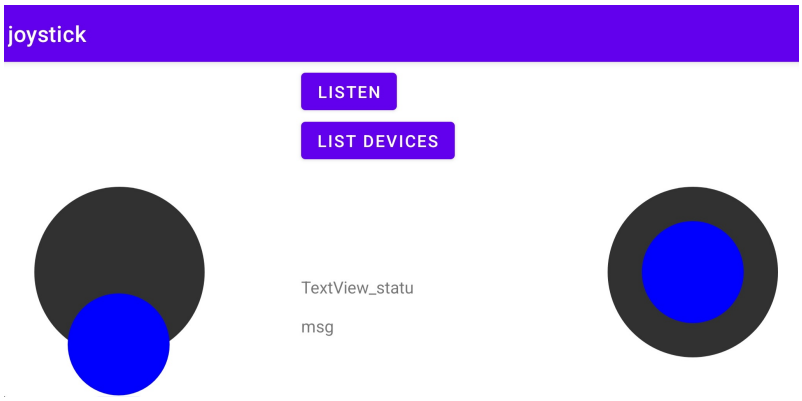


Figure 5.5: Bluetooth joystick app used to control the NAV

Command Handling

The parsing routine (`parse_rc_command`), running on Core 1, handles error checking and value validation. It extracts the four control values from the message and stores them in a thread-safe structure, shared with Core 0.

Once parsed, the control values (roll, pitch, throttle, yaw) are used to update the setpoints for the PID controllers. A deadband is applied around center values (±10 for roll/pitch, ±200 for yaw) to prevent unintended drift due to joystick noise. If no valid data is received within 500 ms, default safe values are restored. This ensures that outdated or dropped BLE connections do not result in unintended commands.

This approach tightly couples BLE communication with the control layer, ensuring low latency and immediate response to user input.

5.6 Custom IMU Driver: `rp_agrolib_bmi088`

One of the contributions of this work is the development of a custom driver for the Bosch BMI088 IMU. The driver, implemented in the `rp_agrolib_bmi088` module, enables efficient and reliable communication between the BMI088 and the RP2040 microcontroller via I^2C. It is fully integrated with the `rp2040_agrolib` framework and optimized for low-latency sensor acquisition.

Library Structure

The `rp_agrolib_bmi088` module provides the following functionality:

- Register-level macros for sensor configuration and data decoding;
- I^2C communication routines using `rp_agrolib_i2c`;
- Initialization of the accelerometer and gyroscope with configurable ranges and bandwidths; and,
- Functions for reading raw or scaled acceleration and angular velocity data.

Ranging Flow

A typical setup and data acquisition loop using the library is shown in Figure 5.6.

This abstraction allows seamless integration of IMU data into the flight control loop, maintaining clean separation between hardware drivers and control algorithms.

Role in Flight Control

The BMI088 driver is called at every control loop iteration (250 Hz) to fetch acceleration and gyroscope data. These measurements are filtered using a Kalman filter to estimate the drone's

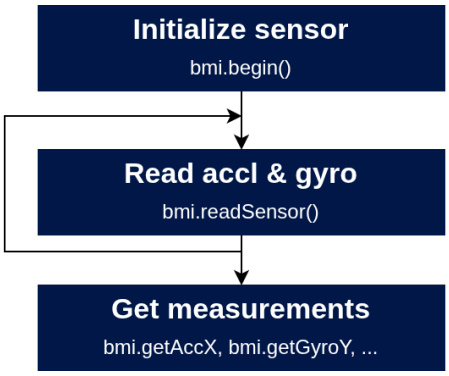


Figure 5.6: Ranging flow using BMI088

attitude in real time. The driver ensures consistent performance and includes error handling for initialization failures.

The low-latency design of the `rp_agrolib_bmi088` module contributes directly to the responsiveness and stability of the quadcopter during aggressive maneuvers and in the presence of environmental disturbances.

5.7 Rate Mode

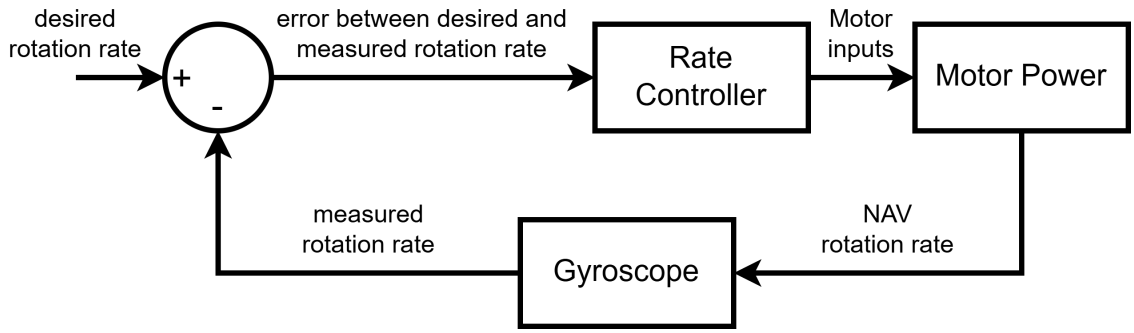


Figure 5.7: Feedback control block diagram for rate mode

Flying and stabilizing a quadcopter manually is nearly impossible due to the high update frequency required. Instead, I implemented an automated control loop that updates motor commands every 0.004 seconds (250 Hz). In rate mode, the drone stabilizes using angular velocity data from the gyroscope.

The inner loop of the control architecture is the rate mode. This is the most basic level of stabilization, in which the drone responds directly to angular velocity commands.

Each of the roll, pitch, and yaw axes is controlled by a PID loop based on gyroscope feedback from the IMU. The desired rotation rates are generated directly by higher control layers or user input.

Control Principle

The control structure for rate mode compares the desired and measured rotation rates and minimizes the error using a PID controller. Initially, only a P controller is considered:

$$\begin{aligned} \text{Error}(k) &= \text{DesiredRate}(k) - \text{Rate}(k) \\ \text{Input}_{\text{motor}}(k) &= P \cdot \text{Error}(k) \end{aligned}$$

This setup responds quickly but may produce steady-state error and overshoot. To address this, an integral term is added:

$$\begin{aligned} \text{Iterm}(k) &= \text{Iterm}(k-1) + I \cdot \frac{\text{Error}(k) + \text{Error}(k-1)}{2} \cdot T_s \\ \text{Input}_{\text{motor}}(k) &= P \cdot \text{Error}(k) + \text{Iterm}(k) \end{aligned}$$

Further improvements are achieved by adding a derivative term, which anticipates the rate of error change:

$$\begin{aligned} \text{Dterm}(k) &= D \cdot \frac{\text{Error}(k) - \text{Error}(k-1)}{T_s} \\ \text{Input}_{\text{motor}}(k) &= P \cdot \text{Error}(k) + \text{Iterm}(k) + \text{Dterm}(k) \end{aligned}$$

This PID loop is executed for each axis—roll, pitch, and yaw—at 250 Hz.

As the figure below shows, the PID controller allows the quadcopter to approach the desired rotation rate quite fast with a small overshoot and settling time.

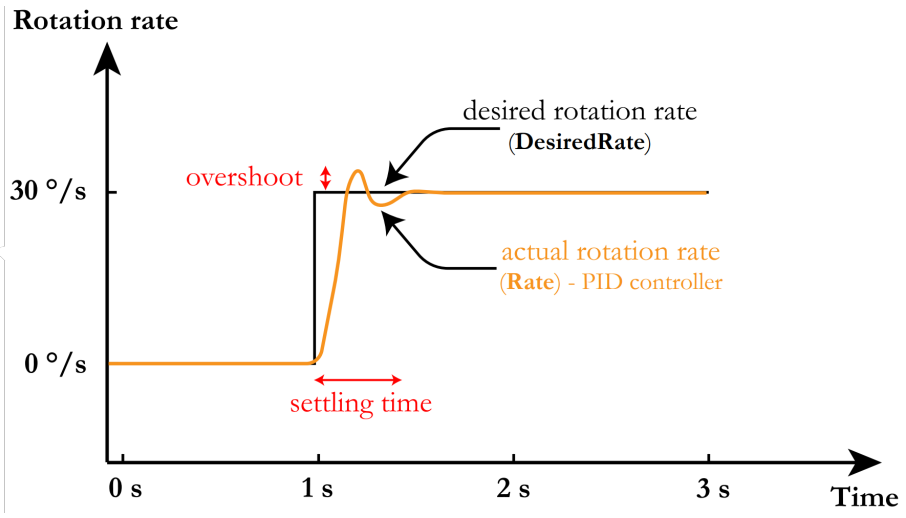


Figure 5.8: PID Response[26]

This mode is essential for executing fast and responsive movements, and forms the base for more complex control strategies.

In figure 5.9 the full control loop with equations is visualized. Start each loop with obtaining the commands from receiver, corresponding to the position of the joysticks on the smartphone app or PC keyboard. Transform these receiver values to the desired roll, pitch and yaw rates and the value for the throttle. Next, obtain the actual roll, pitch and yaw rates of the quadcopter from your gyroscope. These are subtracted from the desired rotation rates to obtain the error between both. Now we have the necessary information for the PID equations, which gives the motor input values for roll, pitch and yaw. Finally, introduce these input values in the motor power commands. With the first iteration now complete, wait until 0.004 seconds have passed to start the next iteration.

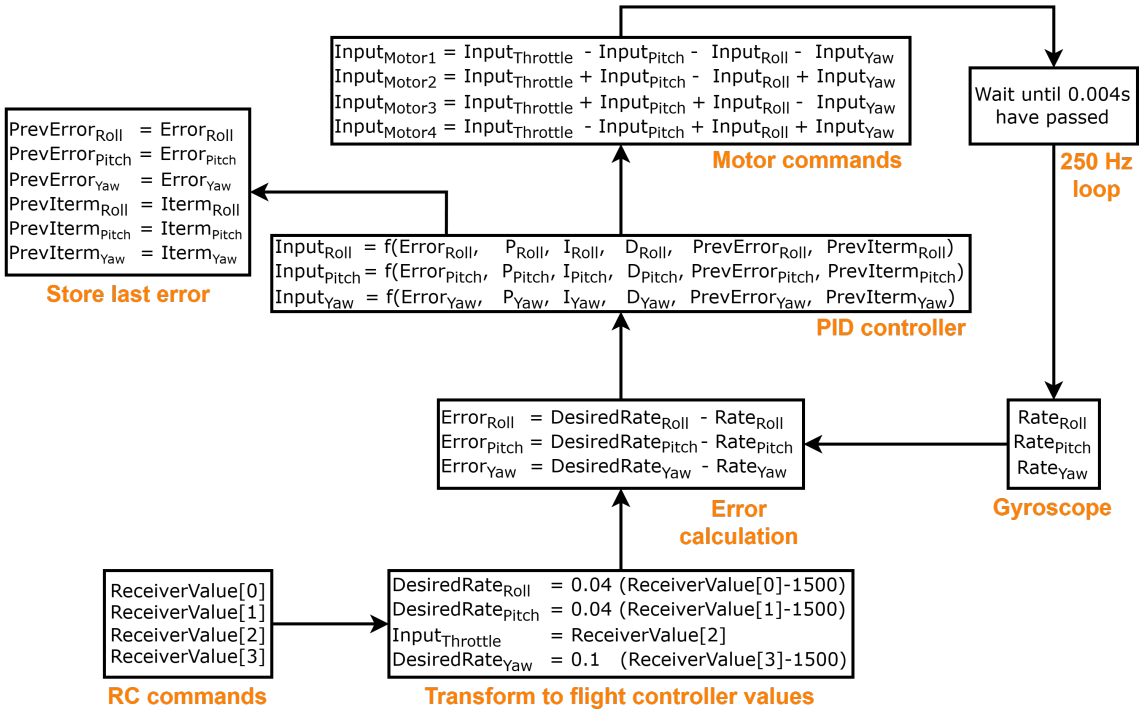


Figure 5.9: Rate control loop

PID Values

Based on tuning guidelines in the Carbon Aeronautics Manual[26], the following values were achieved to a stable flight:

- Roll/Pitch: P = 1, I = 3, D = 0.05
- Yaw: P = 45, I = 10, D = 0.01

Notice that the values for the roll and pitch rates are equal; this is evident since the quadcopter is almost symmetric in both directions. However, for the yaw axis, the control dynamics differs due to the indirect nature of yaw torque generation, which relies on counter-rotating motor pairs

rather than direct thrust imbalance; as a result, higher P and I values are typically required to achieve adequate responsiveness and stability.

The PID tuning was performed manually through iterative test flights, beginning with all gains initialized to zero. The tuning followed a classical approach:

1. **Proportional Gain (P):** Starting from zero, the proportional gain was gradually increased until the drone began to oscillate slightly, indicating the onset of overshoot. This provided an estimate of the maximum useful P value before the system became unstable.

2. **Derivative Gain (D):** A small derivative term was then introduced to dampen the oscillations caused by the proportional gain. The D value was tuned just enough to eliminate the shaking without introducing excessive noise or lag.

3. **Integral Gain (I):** Finally, the integral term was increased to eliminate steady-state drift, especially in hover. The value was tuned conservatively to avoid the accumulation of large error over time, which could lead to integral windup.

This process was repeated independently for roll, pitch, and yaw axes, resulting in the stable gain values reported above.

5.8 Angle Mode

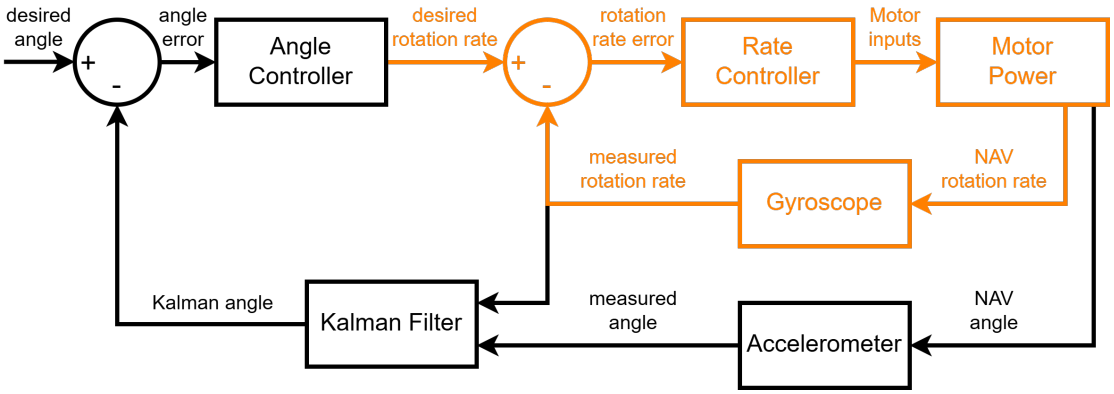


Figure 5.10: Feedback control block diagram for stabilize mode

Flying a quadcopter using only rotation rates is difficult—when you release the stick, rotation stops but the angle stays the same. Stabilize mode improves pilot control by ensuring the drone returns to level when sticks are centered.

To ease pilot control, an outer loop PID controller is added to manage orientation. Desired angles are received from the joystick, and actual angles are computed using a Kalman filter that fuses accelerometer and gyroscope data.

This mode allows the drone to self-level, returning to a horizontal orientation when stick inputs are released. It is especially useful for beginners and for precision tasks such as hover and slow maneuvers.

As shown in Figure 5.11, the smartphone interface allows intuitive control of the drone by mapping joystick input to desired pitch, roll, yaw, and throttle values.

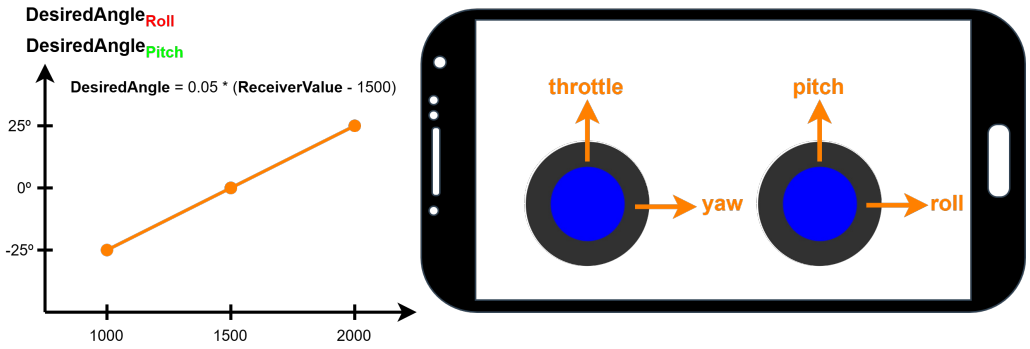


Figure 5.11: Smartphone interface used to send desired angles via BLE. Left stick controls throttle and yaw; right stick controls pitch and roll. These inputs are used as targets in the outer-loop angle controller

Cascaded Controller

This mode adds a P controller for angle stabilization in front of the rate PID controller, forming a cascaded structure:

- Outer loop: Angle error $\rightarrow$ Desired Rate
- Inner loop: Rate error $\rightarrow$ Motor Input

The angle controller requires only a proportional term:

$$\begin{aligned} \text{Error}_{\text{angle}}(k) &= \text{DesiredAngle}(k) - \text{Angle}(k) \\ \text{DesiredRate}(k) &= P_{\text{angle}} \cdot \text{Error}_{\text{angle}}(k) \end{aligned}$$

These desired rates are then used by the inner PID loop from rate mode.

Angle Estimation via Kalman Filter

The angle of the drone is computed by fusing gyroscope and accelerometer data using a Kalman filter:

To estimate the quadcopter's orientation, we begin by predicting the current angle using the previously estimated angle and the angular velocity measured by the gyroscope:

$$\text{Angle}_{\text{kalman}}(k) = \text{Angle}_{\text{kalman}}(k-1) + T_s \cdot \text{Rate}(k) \tag{5.1}$$

where T_s is the control loop period (0.004 s in our implementation), and $\text{Rate}(k)$ is the gyroscope reading at iteration k.

This prediction is inherently uncertain due to sensor noise and integration drift. Therefore, the uncertainty of the predicted angle is computed as:

$$\text{Uncertainty}(k) = \text{Uncertainty}(k-1) + T_s^2 \cdot \sigma^2 \tag{5.2}$$

with $\sigma = 4^\circ/\text{s}$ representing the standard deviation of the gyroscope noise.

To improve the estimate, we introduce the Kalman gain, which determines the relative weighting between the predicted angle and the angle measured by the accelerometer. It is computed as:

$$\text{Gain}_{\text{kalman}} = \frac{\text{Uncertainty}(k)}{\text{Uncertainty}(k) + \sigma_{acc}^2} \tag{5.3}$$

where $\sigma_{acc} = 3^\circ$ is the standard deviation of the accelerometer noise.

A high Kalman gain gives a large importance to the measurement (e.g. the accelerometer), while a low Kalman gain gives a larger importance to the prediction (e.g. the integration of the rotation rate).

Using this gain, the angle estimate is corrected by blending the prediction with the accelerometer measurement:

$$\text{Angle}_{\text{kalman}}(k) = \text{Angle}_{\text{kalman}}(k) + \text{Gain}_{\text{kalman}} \cdot (\text{Angle}_{acc} - \text{Angle}_{\text{kalman}}(k)) \tag{5.4}$$

Over time, as the filter becomes more confident in the gyroscope's long-term stability, the Kalman gain naturally shifts more weight toward the gyroscope-based prediction and less toward the accelerometer, which is subject to short-term noise but provides drift-free measurements.

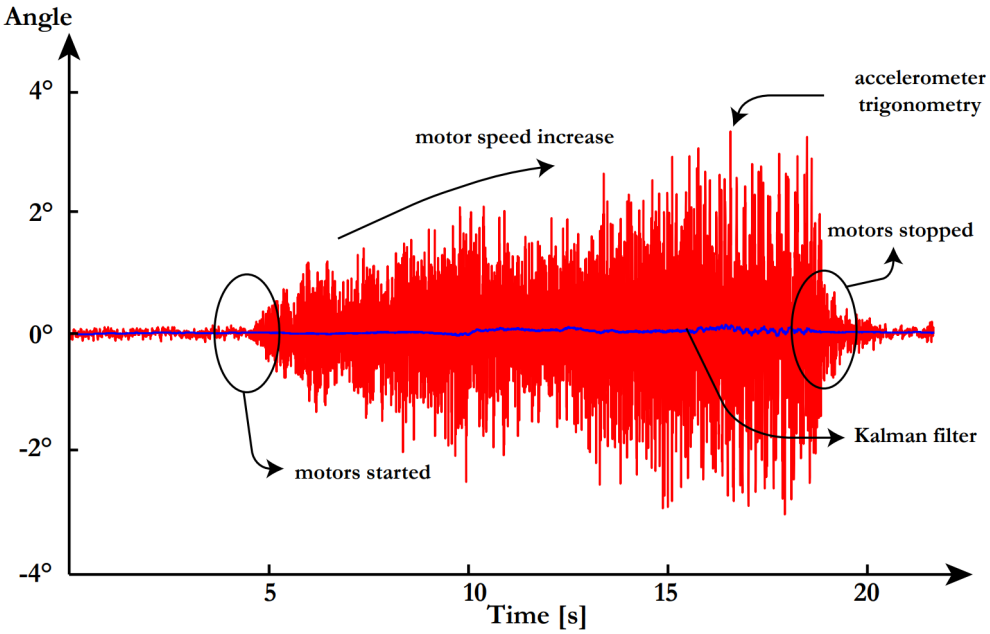


Figure 5.12: Comparison between raw angle estimation from the accelerometer (red) and the Kalman filter output (blue) while the quadcopter remained stationary on the ground. Motor activity is indicated along the timeline[26]

Figure 5.12 illustrates the difference between the angle measurements obtained directly from the accelerometer and those produced by the Kalman filter. The data was collected while the quadcopter remained static on the ground, both before and after the motors were activated. As shown, the accelerometer-derived angles exhibit high-frequency noise, with fluctuations reaching up to $\pm 3^\circ$ due to vibrations induced by the motors. In contrast, the Kalman filter output remains smooth and stable throughout, demonstrating its ability to reject sensor noise and provide a more reliable angle estimate for use in closed-loop flight control.

To further mitigate noise in the measurements, digital low-pass filters were enabled on both the accelerometer and gyroscope. This helped suppress high-frequency disturbances at the sensor level, improving the quality of the raw data feeding into the Kalman filter and enhancing the overall stability of the estimated orientation.

5.9 Custom LiDAR Driver: `rp_agrolib_vl53l8cx`

A significant contribution of this work is the implementation of the `rp_agrolib_vl53l8cx` driver for the VL53L8CX ToF sensor. This driver was adapted from the STMicroelectronics official API and tailored specifically to the RP2040 platform using a custom I^2C backend. The integration ensures reliable distance measurements suitable for altitude control in lightweight quadcopters.

Driver Architecture

The driver builds upon the original vendor-provided API, though I implemented the platform abstraction layer to interface with the RP2040's hardware via:

- `i2c_read_register()` and `i2c_write_register()` for transactional data exchange (from `platform.c`);
- Implemented functions: `VL53L8CX_RdByte`, `VL53L8CX_WrByte`, `VL53L8CX_RdMulti`, `VL53L8CX_WrMulti`; and,
- Timing utilities and platform-specific memory access methods.

This allows the driver to communicate efficiently with the LIDAR sensor, boot its firmware, configure its resolution (4x4 or 8x8), and extract ranging data at up to 60 Hz.

Ranging Flow

The diagram below demonstrates the basic use of the `rp_agrolib_vl53l8cx` driver for initializing the sensor and retrieving distance measurements in a loop.

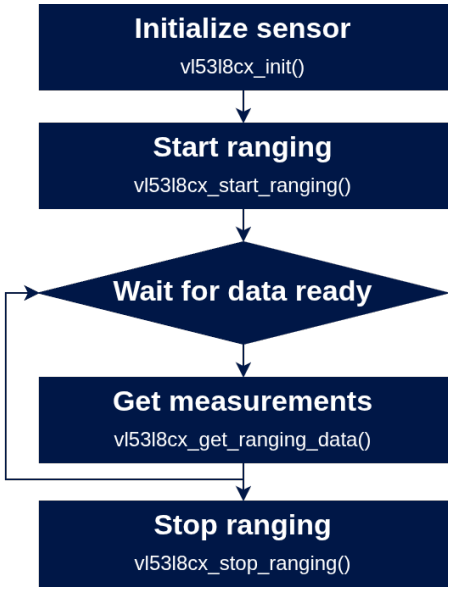


Figure 5.13: Ranging flow using VL53L8CX. Adapted from [27]

Integration with Altitude Hold

The VL53L8CX sensor provides direct and accurate altitude measurements in the short-range domain, making it ideal for indoor hovering and autonomous landing. Its readings are used to compute vertical velocity via finite difference methods and are fed into a PID controller that modulates motor thrust in altitude mode.

The driver's custom platform implementation ensures low-latency I^2C communication, with optimized error handling, timeout recovery, and firmware upload for each session. This enables reliable integration into a 250 Hz flight control loop.

5.10 Altitude Mode

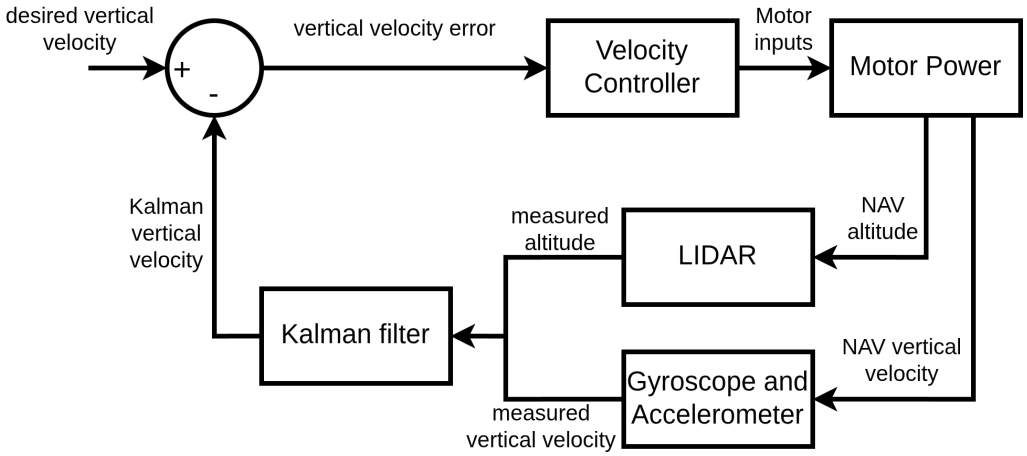


Figure 5.14: Feedback control block diagram for velocity mode (altitude hold)

The altitude hold mode (also referred to as velocity mode) is designed to regulate the drone's height automatically by controlling its vertical velocity. Instead of commanding motor thrust directly via throttle, this mode allows the pilot to specify a desired vertical speed, and the controller adjusts motor output accordingly to maintain or change altitude.

Control Logic

The control approach is based on a PID loop that targets vertical velocity. Unlike direct altitude control, velocity-based feedback avoids integrator windup and allows for smoother transitions, particularly useful during takeoff and hover.

Altitude measurements are acquired from the VL53L8CX ToF sensor and fused with vertical acceleration from IMU using a two-dimensional Kalman filter, which estimates both altitude and vertical velocity. The control system uses the filtered vertical velocity as feedback in the following structure:

- The pilot's throttle input is converted into a desired vertical speed.
- The PID controller compares the desired vertical speed to the estimated $\text{Velocity}_{\text{kalman}}(k)$.
- The controller output adjusts motor power as a correction over the stored throttle baseline at the moment of activation.

This approach ensures smooth control, even if LIDAR updates are delayed or noisy, by relying on a prediction-corrected state rather than raw sensor differences.

Two-Dimensional Kalman Filter for Vertical Velocity Estimation

To accurately control vertical motion during altitude hold, the system must estimate not only the altitude, but also the vertical velocity. As in angle estimation, a Kalman filter is used to combine data from multiple sensors. In this case, we build a two-dimensional Kalman filter to fuse:

- Vertical acceleration from the IMU, and
- Altitude measurements from the VL53L8CX LIDAR.

This fusion allows the estimation of a state vector S that contains both altitude and vertical velocity:

$$S = \begin{bmatrix} \text{Altitude}_{\text{kalman}} \\ \text{Velocity}_{\text{kalman}} \end{bmatrix} \tag{5.5}$$

Prediction Step

We begin with motion equations derived from Newtonian physics. The vertical velocity is the integral of vertical acceleration, and altitude is the integral of velocity:

$$\text{Velocity}_{\text{kalman}}(k) = \text{Velocity}_{\text{kalman}}(k-1) + T_s \cdot \text{AccZ}_{\text{inertial}}(k) \tag{5.6}$$

$$\text{Altitude}_{\text{kalman}}(k) = \text{Altitude}_{\text{kalman}}(k-1) + T_s \cdot \text{Velocity}_{\text{kalman}}(k-1) + 0.5 \cdot T_s^2 \cdot \text{AccZ}_{\text{inertial}}(k) \tag{5.7}$$

This system can be expressed in matrix form:

$$S(k) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} S(k-1) + \begin{bmatrix} 0.5 \cdot T_s^2 \\ T_s \end{bmatrix} \text{AccZ}_{\text{inertial}}(k) \tag{5.8}$$

Let:

$$F = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} 0.5 \cdot T_s^2 \\ T_s \end{bmatrix} \tag{5.9}$$

The initial state and uncertainty are set as:

$$S(k=0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad P(k=0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.10}$$

We assume a standard deviation of $\sigma = 1000\,\text{cm/s}^2$ for the IMU acceleration. The process noise covariance matrix Q is given by:

$$Q = G \cdot G^\top \cdot \sigma^2 \tag{5.11}$$

The predicted uncertainty of the state is:

$$P(k) = F \cdot P(k-1) \cdot F^{\top} + Q \tag{5.12}$$

Measurement Update (Correction Step)

LIDAR provides a direct measurement of altitude. We define the observation model as:

$$M(k) = \text{Altitude}_{\text{lidar}}(k) \tag{5.13}$$

The observation matrix H selects altitude from the state vector:

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.14}$$

Assuming a standard deviation of $\sigma_{\text{lidar}} = 1\,\text{cm}$, the measurement uncertainty is:

$$R = \sigma_{\text{lidar}}^2 = 1 \tag{5.15}$$

With the observation matrix H, calculate the intermediate matrix:

$$L(k) = H \cdot P(k) \cdot H^{\top} + R \tag{5.16}$$

Now we compute the Kalman gain using:

$$K(k) = P(k) \cdot H^{\top} \cdot L(k)^{-1} \tag{5.17}$$

Using the Kalman gain, the updated state becomes:

$$S(k) = S(k) + K(k) \cdot (M(k) - H \cdot S(k)) \tag{5.18}$$

The uncertainty matrix is updated as:

$$P(k) = (I - K(k) \cdot H) \cdot P(k) \tag{5.19}$$

where I is a 2x2 identity matrix:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.20}$$

This two-dimensional Kalman filter provides a robust estimate of both altitude and vertical velocity. It allows the controller to operate effectively even when LIDAR updates are intermittent or noisy. The predicted velocity serves as feedback in the vertical PID loop of the altitude hold mode, enabling smoother and more responsive throttle adjustments.

Despite being designed to operate with LIDAR, this filter structure is identical to that used with barometric sensors, and future upgrades to fusion with additional sensors (e.g., barometers or optical flow) can be done with minimal modification.

The control loop runs at 250Hz, providing high-frequency motor updates. However, altitude and vertical velocity updates from the Kalman filter occur only when new LIDAR data is available—typically at a slower and variable rate-around—60Hz is the max LIDAR measurement rate. This asynchronous behavior ensures the filter uses the most recent sensor data without blocking the remaining flight control.

The Kalman filter is implemented using manual matrix operations for all steps (prediction, innovation, and update). The time step Ts is updated dynamically at each filter call using the actual time difference between LIDAR updates. This allows the system to handle irregular sensor update intervals gracefully.

To improve estimation accuracy, raw vertical acceleration from the IMU is rotated into the inertial frame using real-time roll and pitch angles like in the equation below. Additionally, a Z-axis acceleration bias (AccZBias) is computed at startup and subtracted to remove gravity and sensor offset. This corrected inertial acceleration is then used as the control input in the Kalman filter's prediction step.

$$a_{z,\text{inertial}} = -\sin(\theta)\cdot a_x + \cos(\theta)\cdot\sin(\phi)\cdot a_y + \cos(\phi)\cdot\cos(\theta)\cdot a_z \tag{5.21}$$

where:

- $a_{z,\text{inertial}}$ is the vertical acceleration in the inertial world frame,
- a_x, a_y, a_z are the raw accelerometer readings in the drone's body frame (forward, left, up),
- θ is the pitch angle,
- ϕ is the roll angle.

Limitations and Issues

Although initial flight tests revealed non-functional behavior in altitude hold mode, the root cause has since been identified and resolved. The issue was not related to I2C timing or software limitations, but rather to voltage instability: when all four motors were active, the 3.3V rail experienced voltage spikes due to insufficient decoupling and back EMF from motors, causing the LIDAR to become unresponsive. This was corrected by adding higher-value capacitors after the 3.3V regulator and increasing PWM frequency, which stabilized the voltage and restored reliable LIDAR operation. With consistent ranging restored, the Kalman-based vertical controller now receives regular updates and functions as expected.

The main remaining challenge for robust altitude control is the high-frequency noise present in the inertial Z-acceleration ($a_{z,\text{inertial}}$). This noise propagates through the Kalman filter and affects the accuracy of vertical velocity estimation, which in turn affects control response.

5.11 LEDs Behavior

To provide the operator with visual feedback during the drone initialization and connection process, a simple LED signaling protocol was implemented. This removes ambiguity about the system's current state.

When the drone is powered on, it enters a calibration phase during which the two red LEDs remain solidly lit. After calibration, the red LEDs begin blinking, indicating that the drone is actively searching for a BLE connection. Once the BLE link is established and the drone is fully ready for flight, the red LEDs turn off and the green LEDs are turned on continuously, signaling a ready state.

During flight, both red and green LEDs blink simultaneously at a frequency of 1 Hz, providing a heartbeat-like indicator that the drone is armed and active. This intuitive LED-based signaling allows users to quickly interpret system readiness and state without needing telemetry access or visual inspection of logs.

5.12 Flight Results and Mode Evaluation

Flight Time and Setup

Flight time tests were conducted and, with the finalized hardware configuration, including structural legs and power connectors, average flight time was between 4 and 5 minutes for angle mode and slightly less for velocity mode.

Setup from power-on to liftoff typically takes under 30 seconds, including Bluetooth pairing, sensor checks, and throttle lock.

Rate Mode Evaluation

Initial tests were conducted in rate mode, where pilot inputs directly map to angular velocity targets. While functional, this mode was extremely difficult to control due to the high update frequency and lack of self-stabilization. Minor disturbances or slight overcorrections easily led to loss of control. Manual hovering was practically impossible.

Angle Mode Evaluation

Switching to angle mode dramatically improved flight stability. This mode introduces an outer-loop controller that returns the drone to level orientation when the sticks are released, making it much easier to control for both steady hover and gentle maneuvers. Tuning of PID values yielded a responsive yet stable system, capable of maintaining attitude with minimal pilot correction.

Altitude Mode Evaluation

With the LIDAR issue resolved through improved voltage regulation, altitude hold mode is now operational. The controller runs at 250Hz, with altitude and vertical velocity estimated asynchronously via a two-dimensional Kalman filter that fuses LIDAR data with inertial acceleration.

The vertical velocity PID controller uses empirically tuned gains:

- P = 6
- I = 3
- D = 0.01

Throttle input is translated into a desired climb or descent rate, and the system adjusts motor output accordingly. In flight, it responds smoothly to vertical commands and maintains a stable hover with little oscillations when the desired velocity is near zero.

Some oscillations persist due to noise in the inertial Z-acceleration ($a_{z,\text{inertial}}$), which can affect velocity estimation and long-term stability. Nonetheless, the mode performs reliably in controlled indoor tests and provides a solid foundation for future improvements, including better IMU filtering or additional sensor fusion.

Responsiveness and Propeller Testing

During flight testing, two distinct sets of propellers were evaluated: a larger, high-torque set and a smaller, lightweight set. Initially, the larger propellers, although theoretically more efficient in terms of thrust per watt, introduced noticeable oscillations during flight. This instability was primarily due to their higher rotational inertia, which limited the motors' ability to respond quickly to control inputs, but also due to bad PID calibration.

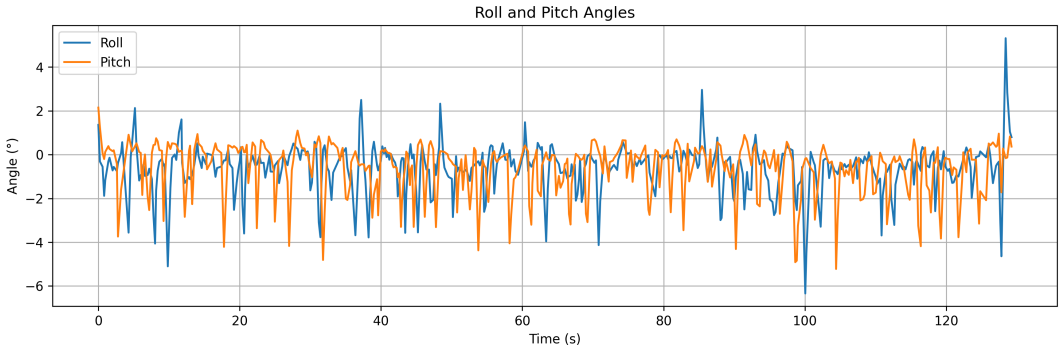
In contrast, the smaller propellers provided a faster dynamic response due to their reduced inertia. This enabled tighter coupling with the control loop and resulted in smoother, more stable flight. Despite delivering slightly less thrust, the enhanced control responsiveness made them the preferred option for the early flight prototypes. Additionally, the reduced oscillations minimized the need for aggressive filtering or extensive PID tuning.

However, with the integration of additional payload—such as structural legs and charging cables—the total mass of the quadcopter increased. This added weight required higher sustained thrust to maintain hover, which pushed the smaller propellers near their performance limits. In response, the PID control parameters were re-tuned from scratch to better match the dynamics of the system. After tuning, the larger propellers exhibited significantly improved behavior, offering both adequate thrust and stable flight. As a result, the final prototype uses the larger propellers, which now deliver both sufficient lift and reliable flight performance under the updated mass constraints.

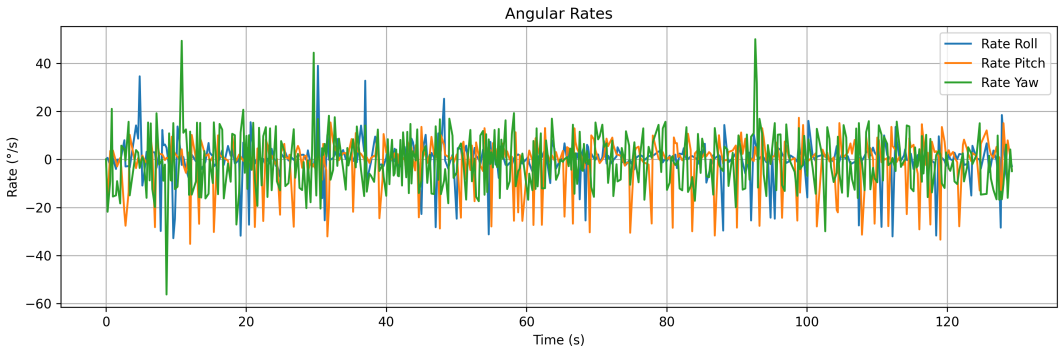
In the end, with smaller propellers, control response was fast but limited in lift. After tuning the PID gains, larger propellers provided improved thrust while maintaining stable flight. Overall control delay is imperceptible to the pilot.

5.13 Telemetry Visualization and Analysis

The integration of a two-way BLE communication pipeline allowed live telemetry capture at 5 Hz during manual and semi-autonomous flight. The telemetry includes roll, pitch, motor PWM, altitude, and velocity, which were recorded and processed using a custom Python script. The resulting plots enable detailed debugging and PID fine tuning.



(a) Roll and Pitch angles during flight



(b) Angular rate responses (Roll, Pitch, Yaw)

Figure 5.15: Attitude telemetry plots during manual control

Figure 5.15 presents the evolution of the drone's attitude during a manual flight session, including roll and pitch angles (Figure 5.15a) and angular rates for all three axes (Figure 5.15b). The roll and pitch angles demonstrate smooth transitions and remain within reasonable bounds, confirming the proper operation of the outer-loop attitude controller. The angular rate plots show continuous oscillations between approximately $+10$ deg/s and -10 deg/s on the roll and pitch axes — a common effect during hover, resulting from minor corrective actions by the inner rate loop. The yaw rate displays slightly larger and more irregular oscillations, likely due to the drone's higher sensitivity to external torque disturbances or asymmetric motor response. Despite these fluctuations, the system maintains stability, and no sustained drift or overshoot is observed. This confirms that the PID gains for angular velocity control are effectively tuned for responsive yet stable behavior.

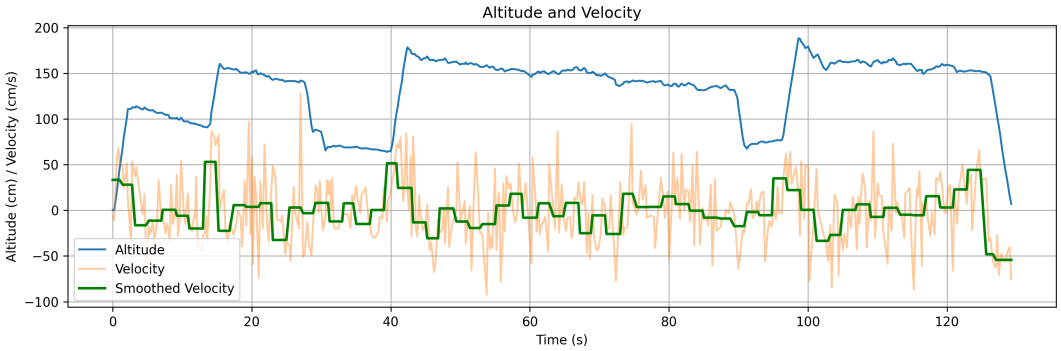


Figure 5.16: Altitude and vertical velocity over time during manual flight

Figure 5.16 shows the drone's vertical behavior during a manual flight session. The altitude curve remains relatively stable over time, with a slow downward drift of a few centimeters every 20 seconds, which is acceptable and fixable with some PID adjust. The vertical velocity signal, estimated using the accInertialZ component from the IMU, exhibits significant high-frequency noise due to inherent accelerometer variability. Nevertheless, the overall trend remains interpretable: sharp increases to approximately $+50\,\text{cm/s}$ during ascent and drops to $-50\,\text{cm/s}$ during descent clearly reflect pilot throttle inputs. These transitions confirm that the system accurately detects and responds to vertical motion, despite the noise in velocity data.

5.14 Challenges, Debugging, and Solutions

During the development and testing of the NAV flight controller, several hardware and software challenges were encountered and progressively resolved through systematic debugging and iteration.

Hardware Challenges and Fixes

- **IMU Orientation:** During initial flight attempts, the drone consistently flipped at takeoff. The root cause was traced to the upside-down mounting of the BMI088 IMU. Software-side axis inversion was implemented to compensate for the orientation, resolving the issue.

- **BLE Module Configuration (BC832):** The BLE module failed to respond to AT commands until a hardware bug on the custom PCB was identified—Pin 9 had not been pulled HIGH. After this was corrected, AT command configuration and stable BLE communication were successfully achieved.

- **Battery Shutdown Under Load:** The original battery could not supply enough current for the motors, leading to abrupt power loss. This was confirmed through testing and resolved by switching to a Crazyflie battery with a higher discharge rating, which restored reliable operation under load.

- **LIDAR blocking:** The root cause of LIDAR data dropout was identified as voltage spikes on the 3.3V rail when all motors were simultaneously active. These transients were severe enough to cause the VL53L8CX sensor to lock up or stop ranging. To resolve the issue, larger decoupling capacitors were added after the 3.3V regulator. This stabilized the power supply under load, enabling continuous and reliable operation of the LIDAR sensor during flight.

Software Challenges and Debugging

- **BMI088 Driver Adaptation:** The BMI088 sensor driver was adapted from an Arduino-based implementation to use the custom `rp_agrolib_i2c` interface. A peculiar issue emerged where the sensor remained unresponsive after resets unless an initial dummy read was performed. This behavior required careful debugging and delayed progress.

- **VL53L8CX LIDAR Integration:** Integration of the ST VL53L8CX time-of-flight sensor required customizing the manufacturer's C API for the RP2040 platform. The most difficult part was developing new low-level I^2C functions in `platform.c/.h` that respected the LIDAR's communication timing requirements. Although functional, earlier the sensor delivered measurements at a slower-than-expected rate due to miss configuration of LIDAR ranging rate that was 1Hz by default, limiting its use for real-time altitude control.

- **BLE Communication Latency:** Early BLE communication tests were made sending single commands via BLE app, which is not ideal when trying to tune PID parameters. A 3D-printed ball joint test fixture 5.17 was used for tuning while the drone was restrained. However, real tuning only became effective once fully wireless, unrestrained flight testing was possible.



Figure 5.17: Drone socket ball joint

- **PID Tuning:** Tuning the cascaded PID controllers was a gradual process. Without direct flight feedback, early gains were derived from simulation. Once flight testing became feasible, a classical tuning method was applied. Gains were incrementally adjusted in-flight, starting from zero, to achieve the stable performance seen in angle mode today.

Summary

Table 5.1 summarizes the results for each implemented flight mode.

Mode	Status	Description and Observations
Rate Mode	Working (Not Usable)	High-frequency manual control using angular velocity inputs. Unstable and overly sensitive to RC input. Telemetry shows oscillations up to $\pm 20^\circ$/s on yaw and $\pm 10^\circ$/s on roll/pitch.
Angle Mode	Working	Stable and responsive cascaded PID control. Allows reliable manual maneuvers and hovering. Telemetry confirms angular oscillations limited to $\pm 1^\circ$ with consistent roll/pitch tracking.
Altitude Mode	Working	Kalman-based altitude hold now works reliably after addressing LIDAR dropout. Vertical drift is minimal (20 cm/min). Some high-frequency noise in vertical velocity from accelerometer ($accInertialZ$), but clear velocity trend is visible during ascents/descents (± 50 cm/s).

Table 5.1: Summary of flight mode evaluations with telemetry-based observations

With quantitative telemetry data now available through the BLE communication link between the drone and the docking station, it became possible to go beyond purely qualitative evaluation methods. Although the original smartphone BLE interface was restricted to one-way command transmission—preventing any form of telemetry reception—the system was significantly improved through the development of a two-way BLE communication pipeline. This new implementation not only enables real-time control input from the docking station to the drone but also supports continuous telemetry streaming at 5 Hz from the drone back to the docking station.

The received telemetry data, which includes parameters such as roll, pitch, angular velocities, motor outputs, altitude, and vertical velocity, is timestamped and logged via USB using a TinyUSB-based implementation on the RP2040. Logging is initiated and managed via serial commands from a host PC, allowing seamless start/stop control and integration with post-processing scripts.

As a result, detailed plots and data-driven analysis have been incorporated into this document, providing objective insight into system behavior, PID performance, sensor noise characteristics, and flight mode stability. This transition from visual-only to quantitative evaluation greatly enhances the reliability and reproducibility of test results and enables future benchmarking of autonomous behaviors.

Chapter 6

Docking Station

The docking station is a crucial element of the autonomous precision pollination system, serving as the physical and electrical interface for recharging and monitoring the micro drone. It enables autonomous landing through a combination of visual guidance and passive mechanical alignment, ensuring reliable operation and extended autonomy.

6.1 Mechanical Design



Figure 6.1: Docking Station design

The docking station comprises a 3D-printed modular structure, developed with adaptability and ease of prototyping in mind. The main components include:

- **Top cover with funnels:** Four conical funnels guide the legs of the descending drone to assist passive alignment. A central cutout accommodates an upward-facing camera for visual tracking.
- **Base structure:** Supports the TL.HEART and TL.HEART#BLC PCBs, provides mechanical support during drone charging, and houses all internal electronics.

- **Arducam adapter mounts:** Modular holders for quick switching between different Arducam modules (HM0360, HM01B0, SPI 2MP PLUS).
- **Copper charging surfaces:** Embedded in the base of each funnel to connect with the drone's copper contact legs for charging.

Drone Adaptation for Docking

The drone was modified to accommodate the docking procedure:

- **Extended 3D-printed legs:** Designed to increase clearance and ensure contact with the charging pads;
- **Copper pads:** Affixed to the base of each leg post-printing to enable power transfer upon contact with the station; and,
- **Leds facing down:** The leds of the drone were facing up, but my integration with the docking needs them emitting light towards the ground where is the camera of the station.



Figure 6.2: Underside view of the drone

All mechanical components of the docking station, including the enclosure, landing supports, and camera adapters, were designed using Autodesk Fusion 360. The camera adapters ensured

correct positioning and field of view alignment for drone detection. All parts were manufactured with PLA filament, enabling fast prototyping and physical iteration throughout the design process.

6.2 Hardware Architecture

The docking station uses a custom-designed PCB stack composed of the TL.HEART and the TL.HEART#BLC board, the last of which was developed in this research:

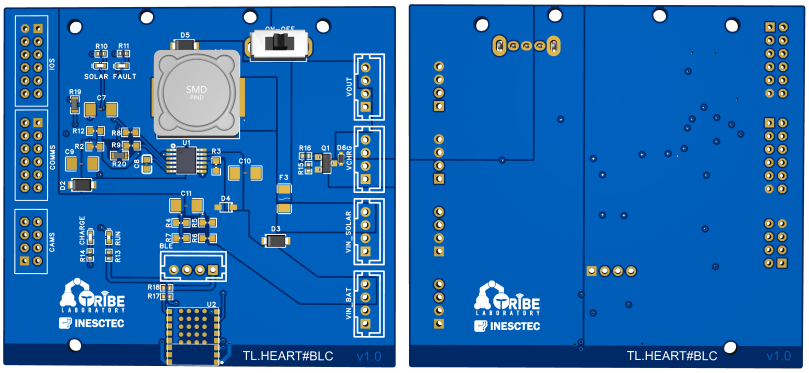


Figure 6.3: Docking Station PCB design

TL.HEART#BLC Highlights

- **Camera interfaces:** SPI and I^2C connections for external modules;
- **Power and charging:** LT3652 battery management with solar panel input support;
- **BLE communication:** Nordic BC832 module for wireless control/feedback from the drone; and,
- **Charging control:** MOSFET-controlled high-current line to copper pads.

The TL.HEART#BLC has now been integrated into the hardware development portfolio of the laboratory and serves as a reusable platform for sensor integration and mobile power systems. This custom PCB used in the docking station was designed using EasyEDA. The detailed schematics are availabe in Appendix B.

6.3 Custom BLE Driver: `rp_agrolib_bc832`

One of the key contributions of this work is the development of a custom driver for the Fanstel BC832 module. The driver enables robust wireless communication between the drone and docking station via BLE. It provides a high-level abstraction over the BC832's AT command interface, optimized for reliable data transmission and connection management in the autonomous precision agriculture system.

Library Structure

The `rp_agrolib_bc832` module provides comprehensive functionality for BLE communication:

- **AT command abstraction:** Complete implementation of BC832 AT command set with automatic mode switching between AT and data modes;
- **Connection management:** Automated connection establishment, monitoring, and recovery for both central and peripheral roles;
- **Message handling:** Buffered data processing with callback-based message reception and transmission confirmation;
- **Device configuration:** Functions for setting device names, RF power, advertising parameters, and scanning configurations; and,
- **Autoconnect functionality:** Intelligent peripheral discovery using manufacturer data filtering for secure device pairing.

Communication Architecture

The library implements a dual-mode communication strategy where the docking station operates as a BLE central device and the drone functions as a peripheral. The central device uses autoconnect functionality with manufacturer data filtering to automatically discover and connect to the designated drone, ensuring secure and reliable communication links.

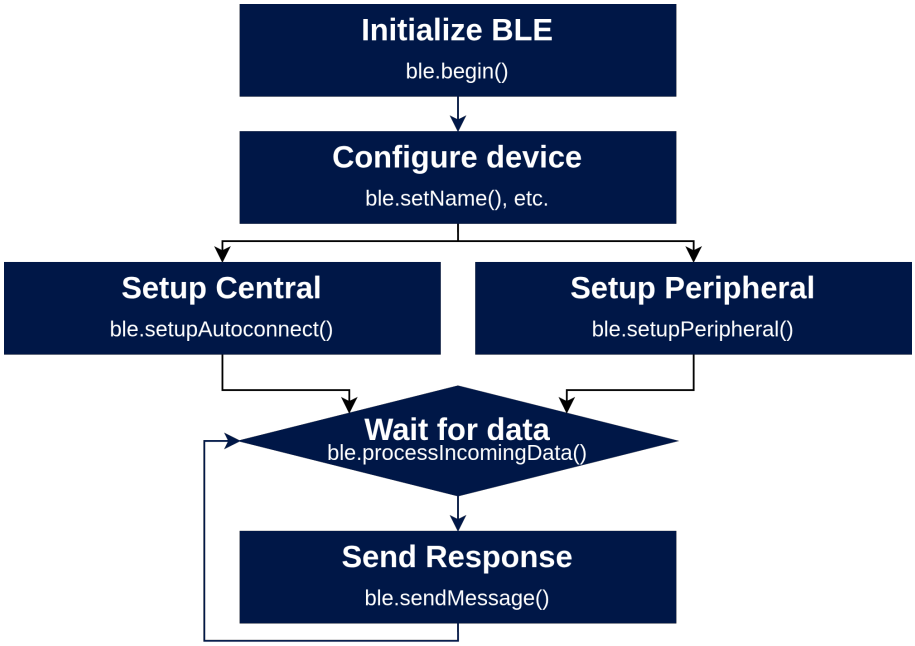


Figure 6.4: BLE communication flow

Key Features

- **Automatic mode management:** Seamless switching between AT command mode for configuration and data mode for message transmission;
- **Connection status monitoring:** Real-time tracking of connection state with automatic reconnection capabilities;
- **Message buffering:** Efficient handling of incoming data with proper message extraction and processing;
- **Transmission confirmation:** Reliable data delivery with SEND acknowledgment handling; and,
- **Error recovery:** Robust error handling with module reset and reconfiguration capabilities.

Integration with Control System

The BLE driver integrates seamlessly with the overall system architecture, enabling real-time transmission of RC commands (roll, pitch, yaw, throttle) from docking station to drone.

This abstraction provides a reliable wireless communication backbone for the autonomous docking system, maintaining clean separation between the BLE hardware interface and higher-level control algorithms.

The library has been successfully tested with continuous data exchange rates suitable for real-time control applications, demonstrating stable connections and reliable message delivery in the precision agriculture environment.

Two-Way BLE Communication and Telemetry Logging

The docking station now implements full-duplex communication using a UART-based BLE interface. The BC832 module on the drone connects to the docking station via BLE, allowing for real-time transmission of RC commands and reception of telemetry data at 5 Hz.

The received telemetry packets are timestamped, cleaned, and transmitted as formatted CSV data. A custom firmware implementation was developed for the RP2040, ensuring reliable data capture. Logging can be toggled from the PC with special keywords like `l` for log and `o` for status, via a Python script interface.

6.4 Docking Workflow

The docking procedure follows these steps:

1. The drone descends toward the docking station using visual cues from downward-facing LEDs.

2. The camera on the docking station detects LED positions, estimates the drone's pose, and computes relative corrections.
3. Real-time RC commands (roll, pitch, yaw, throttle) are generated by a PID controller and sent to the drone via BLE.
4. The funnel design passively aligns the drone legs in the final landing phase.
5. Upon landing, the copper legs contact the charging pads, initiating a controlled charging cycle.

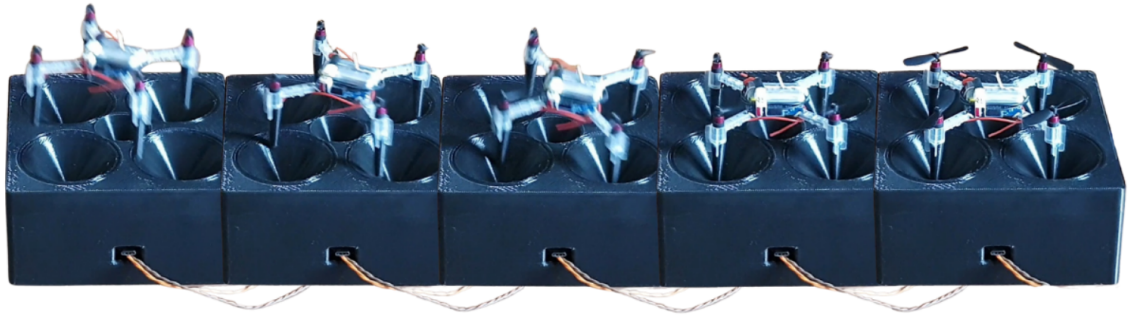


Figure 6.5: Manual land using smartphone APP Video

6.5 Vision-Based Docking Implementation and Challenges

As part of the docking station development, a complete software stack was implemented to enable autonomous landing of the drone using vision-based pose estimation. Two independent processing pipelines were developed for different camera modules:

- `docking_spi.cpp`: Designed for the Arducam SPI 2MP RGB sensor, using color thresholding and LED pair-based pose estimation.
- `docking.cpp`: Developed for the HM0360 grayscale camera via I^2C and parallel interface, implementing geometric LED detection and triangle-based orientation estimation.

Both implementations include the following modules:

- Real-time LED detection algorithms;
- Pose estimation routines for position and yaw;
- PID controllers for roll, pitch, and yaw correction; and,
- BLE communication for transmitting RC commands to the drone.

Implementation Status

Although the software infrastructure was completed and compiled successfully, the entire vision-based docking system remains untested in real-world conditions due to unresolved camera input issues. Specifically:

- Frame acquisition from both SPI and I^2C cameras fails, despite passing hardware initialization steps.
- Pose estimation and PID controllers were not validated with live input or tested on the drone.

Key Challenges

- Camera frame synchronization signals (VSYNC, HREF, PCLK) appear unreliable or misconfigured, preventing successful frame capture;
- Absence of valid image data blocks downstream modules such as pose estimation and control feedback.

Future Work

- Use of logic analyzer or oscilloscope to validate camera signaling at the hardware level;
- Isolated testing of LED detection using mock or prerecorded image data; and,
- Investigation of alternative camera modules with more stable image acquisition support.

While not operational at the time of writing, the developed codebase provides a solid foundation for future testing and integration of a fully autonomous vision-guided docking system.

Chapter 7

Conclusion and Future Work

Pollination is a vital biological process underpinning agricultural productivity and global food security. The global decline of natural pollinator populations—due to factors such as pesticide use, habitat loss, and climate change—necessitates the development of alternative, scalable pollination methods. This dissertation proposed and partially implemented an autonomous micro-drone system designed for precision pollination in greenhouse environments. The system consists of two main components: a custom-built Nano Aerial Vehicle (NAV) flight controller and a docking station designed for contact-based charging and passive alignment. Together, these components contribute to a scalable, low-power, and autonomous solution capable of supporting precision agriculture.

The work completed during this project led to the successful development of a fully functional NAV platform, including a custom flight controller based on the RP2040 microcontroller. The controller supports multiple flight modes (rate, angle, and altitude hold) and integrates BLE-based remote command reception. Extensive testing demonstrated that the drone could achieve stable manual flight using a smartphone joystick application or PC + docking as visible in demonstration videos: Google Drive folder.

On the docking station side, a robust passive alignment structure and contact-based charging circuit were designed, built, and validated. Although the autonomous docking logic using visual pose estimation and real-time control was not completed, the architecture for this functionality was developed and integrated into the system concept. The docking station was designed to support upward-facing camera feedback and BLE-guided correction loops, laying the groundwork for future autonomous landing functionality.

Despite constraints related to time, hardware availability, and system complexity, this work has demonstrated the viability of autonomous micro drones as a future alternative for precision pollination. The project delivers a modular and extensible platform on which future research and development can build.

System Achievements and Scalability Foundations

The implemented system establishes a robust technical foundation for scalable autonomous pollination:

- **Miniaturized Platform:** The drone prototype meets critical size (15 cm $\times$ 15 cm $\times$ 10 cm) and weight ($\leq$ 50 g) constraints, making it suitable for confined agricultural environments;
- **Modular Architecture:** The design allows for component upgrades (e.g., camera modules, controllers) without system redesign;
- **Cost-Effective Implementation:** The total system was built within the budget constraints of €250 for the drone and €200 for the station, supporting economical fleet scaling;
- **Passive Docking Mechanics:** The docking station uses a passive alignment funnel, providing reliable alignment without complex actuators and reducing maintenance in scaled deployments; and,
- **Renewable Energy Readiness:** The PCB of the docking station supports connection with external solar panels and batteries, enabling off-grid operation across distributed sites.

Future Work: Evolutionary Pathways

The following extensions build naturally on the validated foundation, positioning each as a logical and achievable enhancement:

Near-Term Enhancements

- **Vision-Based Docking:** Complete the vision pipeline for autonomous docking, including LED tracking, pose estimation, and BLE-guided control;
- **Flower Detection and Pollination Mechanism:** Integrate a forward-facing camera or sensor and develop perception algorithms for flower detection, potentially leveraging reinforcement learning in later stages; and,
- **Multi-Agent Coordination:** Explore the deployment of multiple NAVs operating in a shared environment, leveraging centralized or distributed scheduling algorithms.

Strategic Scaling Opportunities

- **Autonomous Navigation Expansion:** Leverage existing BLE and altitude-hold capabilities to implement SLAM for a 50 m operational radius, enabling larger greenhouse coverage.
- **Solar Integration Validation:** Field-test the implemented solar-ready PCB under varying light conditions to optimize off-grid deployment.

Concluding Technical Positioning

The work presented in this dissertation delivers a complete functional nucleus for autonomous micro-drone pollination systems. The NAV platform solves core challenges of micro drone stability and control, while the docking station provides reliable energy replenishment infrastructure. Communication protocols establish the connectivity backbone for scalable operations.

These elements form a scalable architectural framework where future work represents capability expansions rather than fundamental reengineering. The system's modularity and cost efficiency position it for incremental enhancement through the outlined pathways, supporting the vision of scalable, environmentally conscious agricultural automation.

With the current software and hardware stack fully implemented — including two-way BLE communication, RC override and USB telemetry logging — the final step toward full autonomy consists of integrating the vision module. Specifically, this involves equipping the docking station with an upward-facing camera to detect the drone's LED pattern, estimate relative pose, and use that information to control drone attitude via PID. This would allow autonomous docking using onboard feedback, replacing or complementing manual commands from the PC keyboard.

References

[1] H. T. Ngo, Y. O. Estrada, G. Design, *et al.*, *The Assessment Report on Pollinators, Pollination and Food Production of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services*. ipbes, 2017. [Online]. Available: www.ipbes.net.

[2] H. Ritchie, "How much of the world's food production is dependent on pollinators?" *Our World in Data*, 2021. [Online]. Available: https://ourworldindata.org/pollinator-dependence.

[3] ResearchGate Contributors, *A schematic showing how the horizon scanning issues for pollinators map onto existing knowledge*, Accessed: 2024-10-29. [Online]. Available: https://www.researchgate.net/figure/A-schematic-showing-how-the-horizon-scanning-issues-for-pollinators-map-onto-existing_fig1_305999958.

[4] T. Hiraguri, H. Shimizu, T. Kimura, *et al.*, "Autonomous drone-based pollination system using ai classifier to replace bees for greenhouse tomato cultivation," *IEEE Access*, vol. 11, pp. 99 352–99 364, 2023, ISSN: 21693536. DOI: 10.1109/ACCESS.2023.3312151.

[5] M. Hassanalian and A. Abdelkefi, *Classifications, applications, and design challenges of drones: A review*, 2017. DOI: 10.1016/j.paerosci.2017.04.003.

[6] I. A. Showers, "Development of an autonomous airborne docking and undocking system for micro drones to a mother drone," M.S. thesis, Cape Peninsula University of Technology, 2019. [Online]. Available: https://etd.cput.ac.za/bitstream/20.500.11838/3161/1/Showers_Inyeni_211052833.pdf.

[7] A. N. Wilson, A. Kumar, A. Jha, and L. R. Cenkeramaddi, *Embedded sensors, communication technologies, computing platforms and machine learning for uavs: A review*, 2022. DOI: 10.1109/JSEN.2021.3139124.

[8] FlexPCB Team, *SPI, I2C, and UART Communication Protocols and Uses*, Accessed: 2025-03-16, 2025. [Online]. Available: https://flexpcb.org/spi-i2c-uart-communication-protocols-and-uses/.

[9] M. Zemlyanikin, A. Smorkalov, T. Khanova, A. Petrovicheva, G. Serebryakov, and X. Ai, *512kib ram is enough! live camera face recognition dnn on mcu*. [Online]. Available: https://github.com/xperience-ai/gap_.

[10] MultiWii Community, *MultiWii: An Open-Source Flight Controller Software*, Accessed: 2025-01-01. [Online]. Available: http://www.multiwii.com/.

[11] AutoQuad Project, *AutoQuad: High-Performance Open-Source Flight Control Software*, Accessed: 2025-01-01. [Online]. Available: http://autoquad.org/.

[12] LibrePilot Project, *LibrePilot: Open-Source Flight Control Software for UAVs*, Accessed: 2025-01-01. [Online]. Available: https://www.librepilot.org/site/index.html.

[13] ArduPilot Development Team, *ArduPilot: Versatile Open-Source Autopilot Software*, Accessed: 2025-01-01. [Online]. Available: https://ardupilot.org/.

[14] Bitcraze AB, *Crazyflie 2.1: Modular Nano Drone Platform*, Accessed: 2025-01-01. [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1-plus/.

[15] MadFlight Project, *MadFlight: Advanced Open-Source Flight Control Platform*, Accessed: 2025-01-01. [Online]. Available: https://madflight.com/.

[16] LuminousBees Team, *ArduBee: The World's Smallest Open-Source Autonomous Drone*, Accessed: 2025-01-01. [Online]. Available: https://www.kickstarter.com/projects/luminousbees/ardubee.

[17] Y. Matsuo, Y. LeCun, M. Sahani, *et al.*, "Deep learning, reinforcement learning, and world models," *Neural Networks*, vol. 152, pp. 267–275, Aug. 2022, ISSN: 18792782. DOI: 10.1016/j.neunet.2022.03.037.

[18] A. M. Ali, A. Gupta, and H. A. Hashim, "Deep reinforcement learning for sim-to-real policy transfer of vtol-uavs offshore docking operations," *Applied Soft Computing*, vol. 162, Sep. 2024, ISSN: 15684946. DOI: 10.1016/j.asoc.2024.111843.

[19] M. Fan, Y. Wu, T. Liao, *et al.*, "Deep reinforcement learning for uav routing in the presence of multiple charging stations," *IEEE Transactions on Vehicular Technology*, vol. 72, pp. 5732–5746, 5 May 2023, ISSN: 19399359. DOI: 10.1109/TVT.2022.3232607.

[20] C. G. Grlj, N. Krznar, and M. Pranjić, *A decade of uav docking stations: A brief overview of mobile and fixed landing platforms*, 2022. DOI: 10.3390/drones6010017.

[21] M. Galimov, R. Fedorenko, and A. Klimchik, *Uav positioning mechanisms in landing stations: Classification and engineering design review*, 2020. DOI: 10.3390/s20133648.

[22] T. Martin, J. R. Blanco, J. B. Mouret, and T. Raharijaona, "Compact docking station for sub-150g uav indoor precise landing," in *2024 International Conference on Unmanned Aircraft Systems, ICUAS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 1267–1274, ISBN: 9798350357882. DOI: 10.1109/ICUAS60882.2024.10557124.

[23] A. Braithwaite, T. Alhinai, M. Haas-Heger, E. McFarlane, and M. Kovač, "Tensile web construction and perching with nano aerial vehicles," in Cham: Springer Proceedings in Advanced Robotics, 2018, pp. 71–88, ISBN: 978-3-319-51532-8. DOI: 10.1007/978-3-319-51532-8_5.

[24] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6008–6014. DOI: 10.1109/ICRA.2019.8793735.

[25] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, 2019. DOI: 10.1109/JIOT.2019.2917066.

[26] C. Aeronautics, *Carbon aeronautics quadcopter build and programming manual*. Carbon Aeronautics, 2023. [Online]. Available: https://github.com/CarbonAeronautics.

[27] S. NV, "Introduction a guide for using the vl53l8cx, low-power, high-performance time-of-flight multizone ranging sensor um3109 user manual," ST, Tech. Rep., Oct. 2024, pp. 1–18. [Online]. Available: https://www.st.com/resource/en/user_manual/um3109-a-guide-for-using-the-vl53l8cx-lowpower-highperformance-timeofflight-multizone-ranging-sensor-stmicroelectronics.pdf.

Appendix A

NAB Schematics

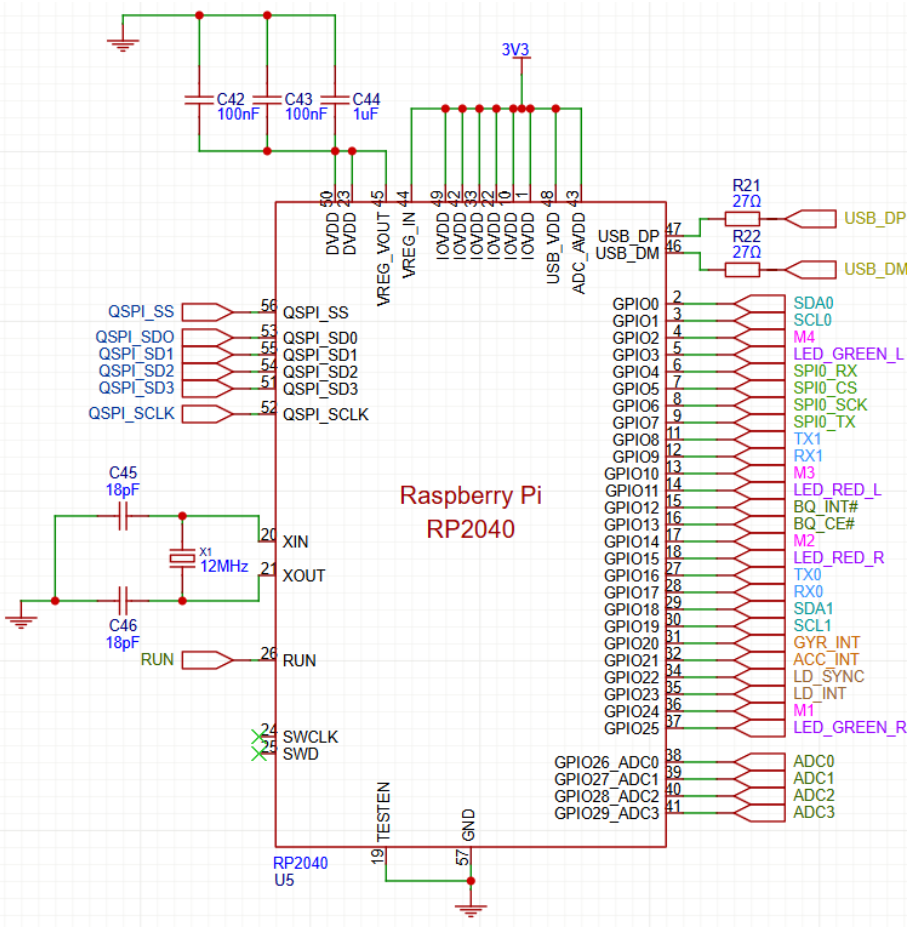


Figure A.1: Schematic of the microcontroller block

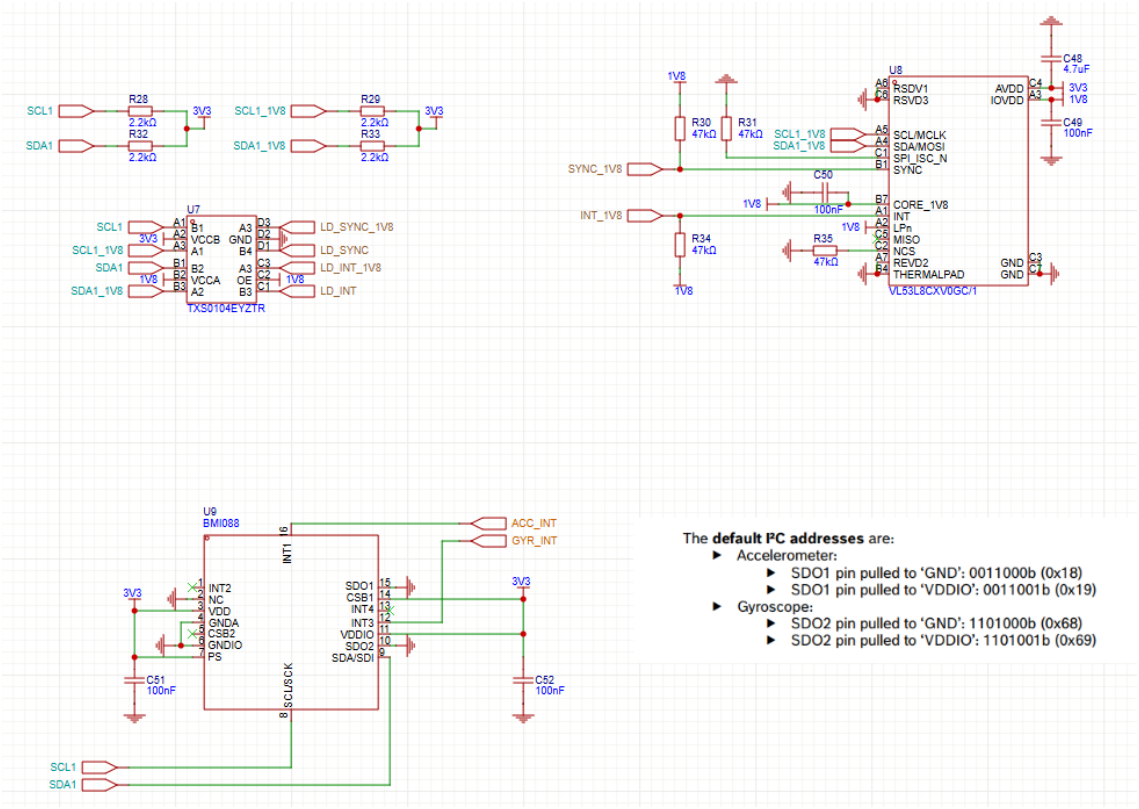


Figure A.2: Schematic of the sensorial module block

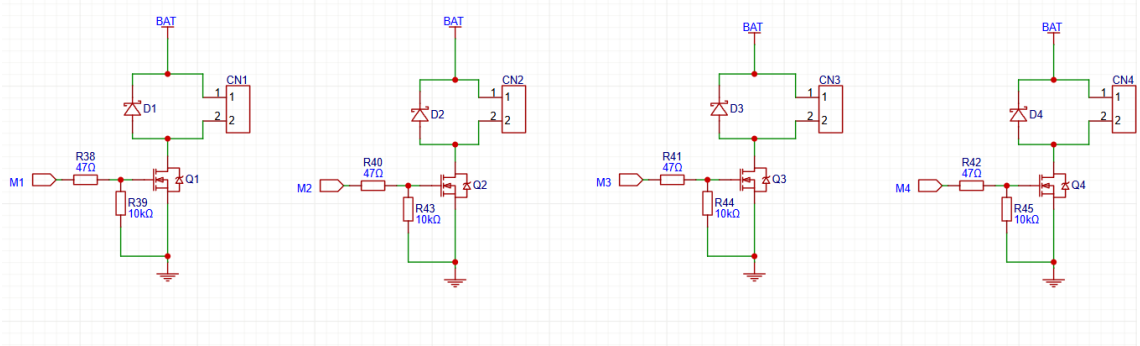


Figure A.3: Schematic of the motor control block

Appendix B

TL.HEART#BLC Schematics

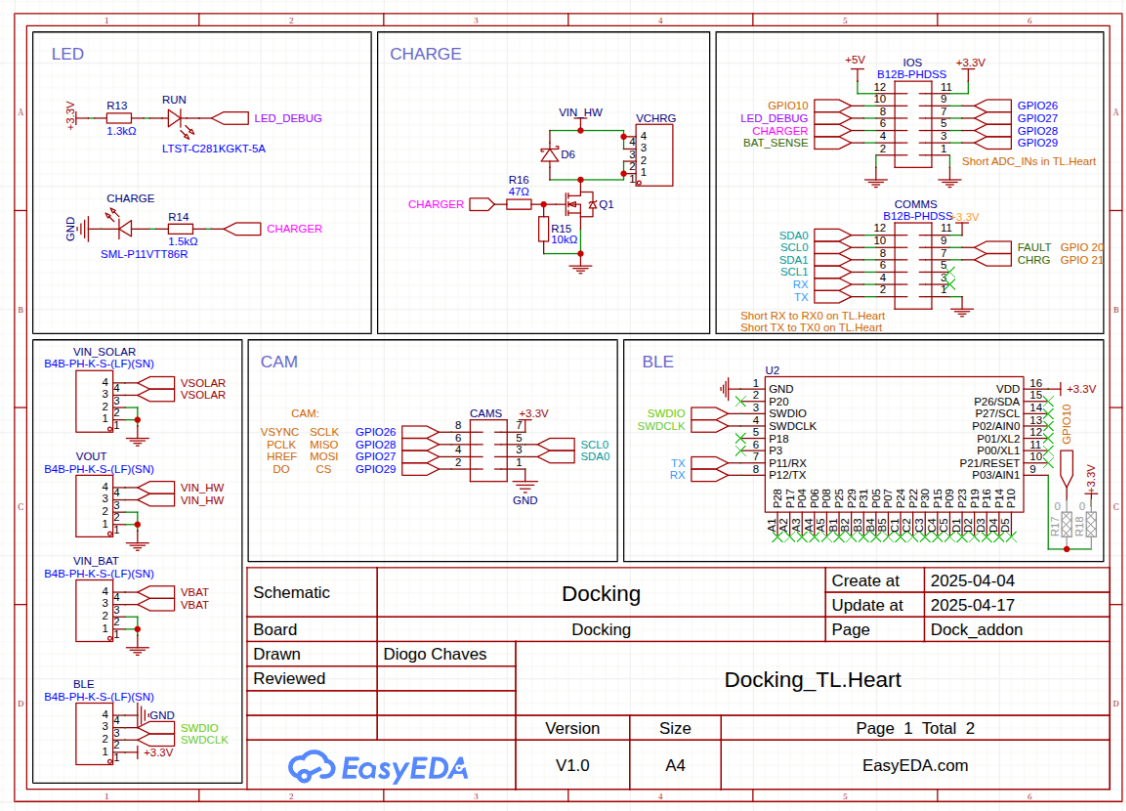


Figure B.1: Interfaces module schematic

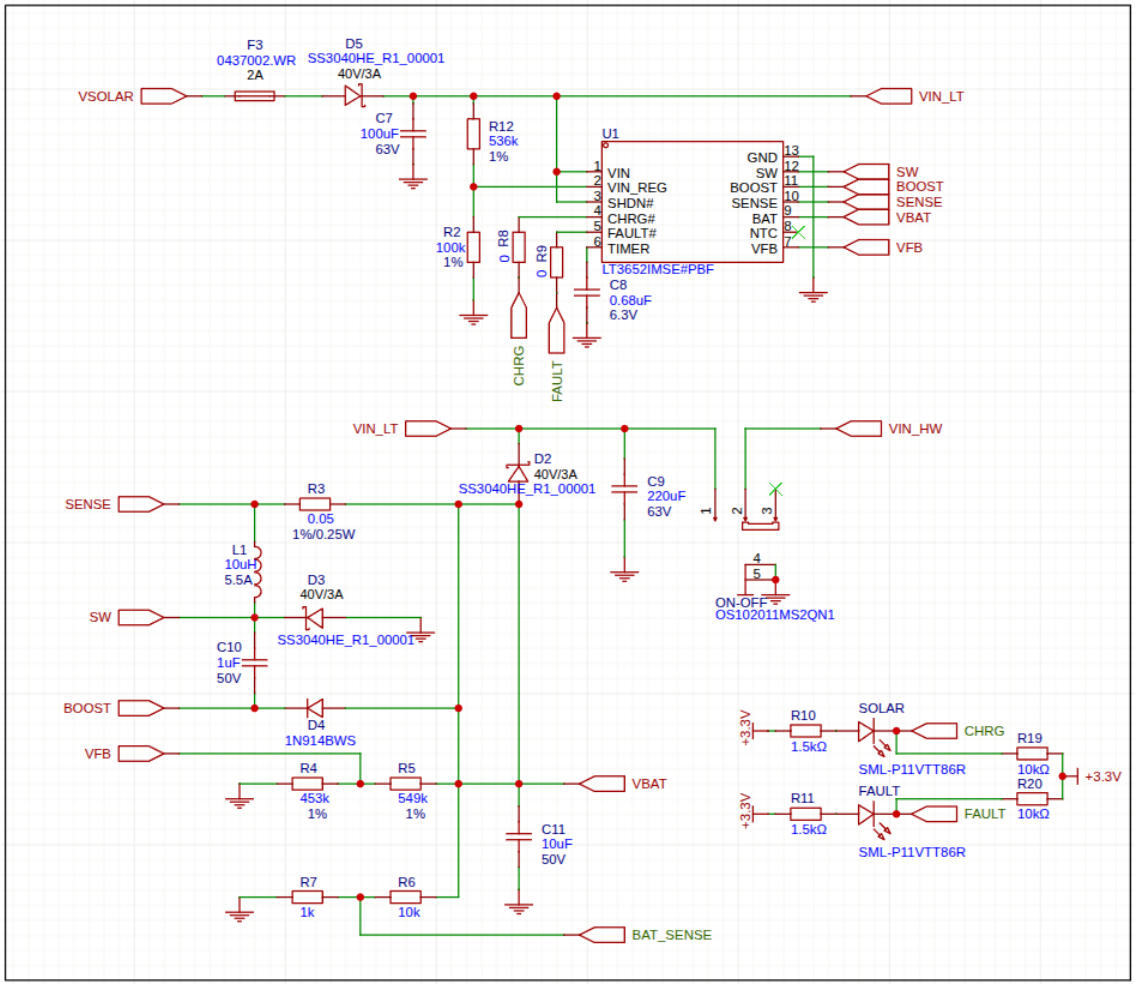


Figure B.2: Solar module schematic