ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A PRACTICAL IMPLEMENTATION OF NAVIGATION AND OBSTACLE AVOIDANCE FOR QUADCOPTERS

M.Sc. THESIS

Onur YILDIRIM

Department of Mechatronics Engineering

Mechatronics Engineering Programme

OCTOBER 2022

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A PRACTICAL IMPLEMENTATION OF NAVIGATION AND OBSTACLE AVOIDANCE FOR QUADCOPTERS

M.Sc. THESIS

Onur YILDIRIM
(518191046)

Department of Mechatronics Engineering

Mechatronics Engineering Programme

Thesis Advisor: Prof. Dr. Ovsanna Seta ESTRADA

OCTOBER 2022

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DÖRT PERVANELİ HELİKOPTERLER İÇİN BİR ENGELDEN KAÇINMA VE SEYRÜSEFER UYGULAMASI

YÜKSEK LİSANS TEZİ

Onur YILDIRIM
(518191046)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ovsanna Seta ESTRADA

EKİM 2022

Onur YILDIRIM, a M.Sc. student of ITU Graduate School student ID 518191046 successfully defended the thesis entitled "A PRACTICAL IMPLEMENTATION OF NAVIGATION AND OBSTACLE AVOIDANCE FOR QUADCOPTERS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Ovsanna Seta ESTRADA** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Metin GÖKAŞAN** Istanbul Technical University	.....
	Asst. Prof. Levent UCUN Yıldız Technical University	.....
		.....

Date of Submission : **4 October 2022**
Date of Defense : **21 October 2022**

To my family,

FOREWORD

First of all, I would like to thank Prof. Dr. Ovsanna Seta ESTRADA for her guidance and support from the beginning to the end of the project.

I would like to thank Prof. Dr. Metin GÖKAŞAN for his support and directions.

I also would like to express my appreciation to my friend Aykut ÖZDEMİR for his support and help.

I would like to thank my family for their support and love. I dedicate this work to them.

October 2022

Onur YILDIRIM
Mechatronics Engineer

TABLE OF CONTENTS

ABBREVIATIONS

BLDC : Brushless Direct Current
MEMS : Micro Electro Mechanical Systems
LiPo : Lithium Polymer Battery
DOF : Degrees of Freedom
RC : Radio Controlled
ESC : Electronic Speed Controller
IMU : Inertial Measurement Unit
GPS : Global Positioning System
GNSS : Global Navigation Satellite System
PWM : Pulse Width Modulation
MCU : Microcontroller
ARM : Advanced RISC Machine
FPU : Floating Point Unit
DMA : Direct Memory Access
SPI : Serial Peripheral Interface
I2C : Inter-Integrated Circuit
UART : Universal Asynchronous Receiver Transmitter
HAL : Hardware Abstraction Layer
UAV : Unmanned Aerial Vehicle
GCS : Ground Control Station
PF : Probabilistic Foam
SLAM : Simultaneous Localization and Mapping
PC : Personal Computer
PID : Proportional Integral Derivative
PD : Proportional Derivative
ANN : Artificial Neural Network
DCM : Direction Cosine Matrix
KF : Kalman Filter
EKF : Extended Kalman Filter
UKF : Unscented Kalman Filter
RRT : Rapidly Exploring Random Trees
PRM : Probabilistic Roadmap Methods
FMT : Fast Marching Trees
BIT : Batch Informed Trees
GPF : Goal-biased PF
HPF : Heuristic-guided PF
AHPF : Angle and Heuristic Biased Probabilistic Foam
PPM : Pulse Position Modulation
NED : North East Down
CPP : Coverage Path Planner

SYMBOLS

$\mathbf{R_x}$	: Roll Rotation Matrix
$\mathbf{R_y}$	: Pitch Rotation Matrix
$\mathbf{R_z}$	: Yaw Rotation Matrix
$\mathbf{K_p}$	: Proportional Gain
$\mathbf{K_i}$	: Integral Gain
$\mathbf{K_d}$	: Derivative Gain
ϕ	: Roll Angle
θ	: Pitch Angle
ψ	: Yaw Angle

LIST OF TABLES

Page

LIST OF FIGURES

A PRACTICAL IMPLEMENTATION OF NAVIGATION AND OBSTACLE AVOIDANCE FOR QUADCOPTERS

SUMMARY

Robotics is a multidisciplinary science field which includes the design, production and usage of robots. Robotic systems consist of robots and other devices and systems used with robots. Nowadays, robots are primarily used in the military, agriculture, food packaging and production sector. The robots that are used in these sectors increase productivity and automate the process, as well as work in areas that can be harmful to humans. The ability to be used in many areas and the surveillance capability of Unmanned Aerial Vehicles (UAV) increases its popularity, therefore increasing its usage in these sectors. UAVs can be defined as a type of aircraft with a thrust system, can be operated with a remote controller, and has a payload capacity. UAVs can be divided into two main categories: fixed-wing types and multi-rotor UAVs. Quadcopters can be described as multi-rotor UAVs with vertical takeoff capability. These UAVs use four actuators and navigate in a 3D coordinate system. These vehicles have six degrees of freedom which, three of them represents x,y and z axes and other three are rotation around these axes.

The advancements in rare earth magnets made it possible to develop very lightweight but powerful BLDC motors. Advancements in battery technology enabled high power density batteries. With the help of MEMS technology, lightweight and fast sensors have become accessible, leading to quadcopters becoming a prevalent research topic. A Quadcopter consists of motors, ESCs, battery and autopilot hardware with sensors and a microcontroller to run flight control and navigation software. Flight control software runs on an embedded microcontroller and calculates the motor output by processing the sensor data and the reference input. In teleoperated systems, these inputs are generated by the operator via Ground Control Stations (GCS). In autonomous systems, high-level navigation software gives reference inputs to the flight controller, and flight control software gives sensor readings and quadcopter state to high-level navigation software. The software is designed in a way that operator can always take over the control of the aircraft in all flight modes in case of any failure or controller problem, ensuring the safety of the development platform.

In this thesis, a quadcopter is built, and low-level and high-level software is developed. First, the necessary hardware, electronic modules and sensors for a quadcopter setup are determined. After building the development platform, sensor drivers and calibration software are developed for the NUCLEO-H7A3 development board. The gyroscope sensor is calibrated to get more accurate angular velocity readings. The accelerometer is calibrated and filtered to remove misalignment errors from the sensor data. The magnetometer sensor is calibrated to eliminate hard and soft iron effects. These calibration stages are crucial for the filtering algorithm. A robust quadcopter

attitude estimation is achieved by fusing calibrated GPS, accelerometer, barometer and gyroscope readings.

In order to control the quadcopter for attitude, position and velocity tracking, a cascaded PID controller structure is used. The attitude control of the quadcopter is achieved using six cascaded PID controllers, and the position control of the quadcopter is performed using seven cascaded PID controllers. In total, thirteen PID controllers are used in the flight control system. The controller performance is tested on the platform with different references.

Online planning and replanning methods are commonly used in unknown or dynamic environments to ensure path safety. As high-level navigation and obstacle avoidance software, a novel local path replanning algorithm is proposed. The probabilistic Foam (PF) idea is adapted for reactive planning tasks. Probabilistic Foam infrastructure is selected for several reasons. Firstly, these types of planners are lightweight, and secondly, they ensure collision-free path generation. But they don't guarantee path length optimality. In this work, the latest achievement on the PF idea was further improved and adapted to local planning tasks. It is proved that this algorithm produces shorter paths and more lightweight than alternatives by conducting qualitative and quantitative tests. This algorithm is also tested as a local planner on coverage path tracking and 2D obstacle avoidance tasks designed for a quadcopter in a realistic simulation environment.

DÖRT PERVANELİ HELİKOPTERLER İÇİN BİR ENGELDEN KAÇINMA VE SEYRÜSEFER UYGULAMASI

ÖZET

Batarya teknolojilerindeki ilerlemeler, güç ağırlık oranı yüksek fırçasız doğru akım motorlarının üretilebilmesi ve ilerleyen sensör teknolojileri sayesinde İnsansız Hava Araçları (İHA) günümüzde tarım, savunma sanayi, yangın söndürme, gözetleme, ilk yardım ve faydalı yük taşıma gibi birçok alanda kullanılmaktadır. İHA'ların en önemli özellikleri yüksek manevra kabiliyeti, küçük boyutları ve otonom olarak görev yapabilmeleri olarak sıralanabilir. İHA'lar sabit kanatlı ve döner kanatlı olarak iki ana kategoride incelenebilir. Sabit kanatlı İHA'lar uzun menzil ve yük taşıma kabiliyetleri nedeniyle tercih edilirken döner kanatlı İHA'lar dikey kalkış iniş yeteneği, havada asılı kalabilmesi ve boyutları nedeniyle tercih edilmektedir.

Dört pervaneli helikopterler de döner kanatlı kategorisine giren İHA tiplerindendir. Birçok şirketin çeşitli uygulamalar için geliştirmiş olduğu ürünler günümüzde kolayca ulaşılabilmektedir. Ayrıca savunma sanayisinde gözetim, savunma ve saldırı görevleri için kullanımları yaygınlaşmıştır. Bu tarz araçlar uzaktan kontrollü şekilde veya otonom olarak kullanılabilmektedir. Uzaktan kontrol edilebilen sistemlerde operatör, Yer Kontrol İstasyonu (YKİ) üzerinden kablosuz şekilde veri alışverişi yardımıyla kontrolü gerçekleştirilebilmektedir. Fakat bu sistemlerde veri alışverişinde kesinti olması durumlarında bu araçların kaza ve kırımla karşılaşma olasılığı yüksek bir durumdur. Ayrıca bu araçların savunma sistemlerince sinyal karıştırma yöntemleriyle şaşırtılması da bu araçlar için tehlike arz eden durumlardandır. Bu yüzden bu tarz cihazların otonom karar verme, sensör gürültüleri ve bozuculara dayanıklı tasarımı önem arz etmektedir.

Ayrıca dört pervaneli helikopter otonom görev sırasında bilinmeyen bir güzergahta uçuş yaparken önüne çıkan engellere çarpabilmekte ve görevini tamamlayamadan kırıma uğrayabilmektedir. Bu durumları önlemek amacıyla aracın çevresel farkındalığını arttırmak için ultrasonik engel sensorleri, iki boyutlu lidarlar ve kameralar kullanılabilmektedir. Bu sensörlerden gelen veriler seyrüsefer yazılımında işlenerek görev sırasında bir engelle karşılaşılması durumunda hava aracının durması, engelin etrafından dolaşması ya da kullanıcıyı bilgilendirip manuel kontrolü ele alması sağlanabilmektedir.

Bu çalışmada bir dört pervaneli helikopter donanımı ve alt ve üst seviye yazılımı geliştirilmiştir. Öncelikle donanım olarak kullanılacak motor, sürücü pervane ve gövde seçimi yapılmıştır. Bu parçalar birleştirildikten sonra mikroişlemci seçilmiş ve gerekli olan ataletsel ölçüm birimi, manyetometre, barometre sensörleri ile donanım oluşturulmuştur. Bilgisayar ile haberleşme ve veri aktarımı için radyo alıcı verici ve bluetooth modül kullanılmıştır. Manuel kontrol için radyo kontrollü uzaktan kumanda kullanılmıştır.

Uçuş kontrol yazılımında çeşitli uçuş modları bulunmaktadır. Bunlar, kumanda ile açı kontrollü manuel uçuş, kumanda ile açısal hız kontrollü manuel uçuş, kumanda ile dikey hız kontrolü ve açı kontrollü uçuş, kumanda ile GPS yatay ve dikey konum kontrollü uçuş, otonom kalkış, otonom eve dönüş, otonom acil iniş ve otonom hedef takibi olarak sıralanabilir.

Yazılımda öncelikle mikroişlemci saat ve çevresel birim ayarları yapılmıştır. Daha sonrasında sensör sürücü yazılımları oluşturulmuştur. Kullanılan sensör türüne göre kalibrasyon algoritmaları eklenmiştir. Dönüölçer sensörü açısal hız bilgisi vermektedir. Hareketli durumda sensör yanıtı düzgün olarak alınabilmektedir ancak bu sensör sabit halde iken çıkışında hareket varmış gibi çıktı vermekte ve kalıcı durum hatasına sebep olmaktadır. Bu durum araç her uçuş öncesi iki saniye boyunca sensör verilerinin toplanıp kalibre edilmesi ile giderilmiştir.

İvmeölçer sensörü doğrusal ivmeyi ölçmektedir. Bu sensör eksenleri arasında aynı uyarılara farklı cevaplar vermektedir. Bu durum her üç eksende yer çekimi ivmesini pozitif ve negatif olarak ölçerek elde edilen veriler sayesinde kalibrasyon uygulanarak çözülmesi önerilmiştir.

Manyetometre sensörü manyetik alan ölçümü yapmaktadır. Bu sensör de ivmeölçer gibi eksen kaçıklığı problemi yaşamaktadır. Eksen kaçıklığı problemine yakında bulunan ferromanyetik nesneler ve akım taşıyan kablolar neden olmaktadır. Bu nedenle manyetometre sensörü kullanılacağı ortama montajlandıktan sonra kalibre edilmelidir. Kalibrasyon, sensörü tüm eksenleri etrafında her yönde çevirerek sensör verilerinin toplanması ve toplanan verilerin kalibrasyon algoritmasında kullanılması ile gerçekleştirilmiştir.

Sensörlerden gelen veriler doğrudan yazılımda kullanılamamaktadır. Bu nedenle sensör füzyonu algoritmaları oluşturulmus ve veri doğruluğu iyileştirilerek hatalar azaltılmıştır. Konum verisi GPS ve ivmeölçer verilerinin füzyonu ile iyileştirmiş ve güncelleme hızı yükseltilmiştir. Dikey hız verisi barometre ve ivmeölçer füzyonu ile hızlandırılmış ve güvenilirliği arttırılmıştır. Dönüölçer ve ivmeölçer verileri füzyonu ile açı bilgisi hesaplanması iyileştirilmiştir.

Dört pervaneli helikopterin kontrolü amacıyla PID kontrolcüler basamaklı olarak kullanılmıştır. Standart PID kontrolcü yapısı iyileştirilmiş ve sayısallaştırılarak yazılıma entegre edilmiştir. Tutum kontrolü için altı, pozisyon ve hız kontrolü için yedi olmak üzere toplamda on üç PID kontrolcü kullanılarak dört pervaneli helikopterin tutum ve pozisyon kontrolü gerçekleştirilmiştir.

Üst seviye navigasyon yazılımında ise quatrotor ile yol planlama ve engelden kaçınma yöntemleri üzerine çalışmalar yapılmıştır. Yol planlama yöntemleri bir noktadan hedef noktaya ulaşabilecek yolların ve ziyaret edilmesi gereken noktaların planlanmasına verilen addır. Kapsayıcı yol planlayıcılar İHA'lar ile tarımsal ilaçlama için uygun yöntemlerdir. Bu yöntemler verilen yol aralığında (örn. İHA ile ilaçlamada ilaçlama bölgesi genişliği) belirli bir alanda ziyaret edilmemiş alanın kalmaması için yollar planlamak için kullanılırlar. Bu yöntemler genelde sarmal şeklinde belirli aralıkta yollar oluşturur ve bu yollar İHA'lar tarafından takip edilir. Buradaki en büyük sorun ise ortamın detaylı haritasının elde edilmesinin zor ve masraflı olmasıdır. Tarama alanının içerisinde bulunan engeller (ev, ağaç, direk vb.) İHA'lar için risk teşkil

etmektedir. Bu tarz durumlarda engelden kaçınma ve lokal planlama yöntemleri sıklıkla kullanılmaktadır.

İHA'larda lokal planlama ve engelden kaçınma yöntemleri ile engellerden belirli bir mesafede kaçınmak mümkün olmaktadır. Fakat bu görevi gerçekleştirmek için algılayıcılara ihtiyaç duymaktadırlar. Maliyetlerinin ve ağırlıklarının düşük olması sebebiyle iki boyutlu lidar algılayıcıları hava ve kara araçlarında sıklıkla kullanılmaktadır. Bu çalışmada engelden kaçınma ve yol takibi icrası için yol planlama çalışmalarından ilham alınarak yeni bir engelden kaçınma yöntemi önerilmiştir. Bu yeni önerilen yöntem, olasılıksal köpük yöntemi (PF) tabanlı çalışmaların lokal planlamaya uyarlanması ile gerçekleştirilmiştir. Yöntemin hız, efektiflik ve engelden kaçınma performansı diğer yöntemlerle karşılaştırılmıştır. Bu karşılaştırmalarda ortalama planlanan yolların uzunluğu, hesaplama süresi ve örnekleme efektifliği açısından avantajlı olduğu kanıtlanmıştır. Bu bulgular ışığında Kapsayıcı yol planlayıcı kullanarak oluşturulan yolların üzerinde simülasyon ortamında engeller yerleştirilerek yöntemin sonuçları gözlemlenmiştir. Unreal fizik motoru üzerinde yapılan farklı engel yoğunluklarına sahip dört farklı ortamda yapılan testlerde yöntemin engelden kaçınma ve yolu takip etme davranışları incelenmiştir. Bu bilgiler ışığında önerilen AHPF yönteminin efektif, güvenli yol üretme ve yol takip görevlerinde kullanılabilecek potansiyeli olduğu kanıtlanmıştır. Hesaplama süresinin oldukça kısa olması da bu yöntemimin gömülü sistemler üzerinde çalışmasını mümkün kılmaktadır. Sonraki çalışmalarda bu engelden kaçınma yönteminin gömülü bir sistem üzerinde denenmesi planlanmaktadır.

1. INTRODUCTION

Robotics is a multidisciplinary science field which includes the design, production and usage of robots. Robotic systems consist of robots and other devices and systems used with robots. Nowadays, robots are primarily used in the military, agriculture, food packaging and production sector. The robots that are used in these sectors increase productivity and automate the process, as well as work in areas that can be harmful to humans. The ability to be used in many areas and the surveillance capability of Unmanned Aerial Vehicles (UAV) increases its popularity, therefore increasing its usage in these sectors. UAVs can be defined as a type of aircraft with a thrust system, can be operated with a remote controller, and has a payload capacity. UAVs can be divided into two main categories: fixed-wing types and multi-rotor UAVs. Quadcopters can be described as multi-rotor UAVs with vertical takeoff capability. These UAVs use four actuators and navigate in a 3D coordinate system. These vehicles have six degrees of freedom which, three of them represents x,y and z axes and other three are rotation around these axes.

The advancements in rare earth magnets made it possible to develop very lightweight but powerful BLDC motors. Advancements in battery technology enabled high power density batteries. With the help of MEMS technology, lightweight and fast sensors have become accessible, leading to quadcopters becoming a prevalent research topic. A Quadcopter consists of motors, ESCs, battery and autopilot hardware with sensors and a microcontroller to run flight control and navigation software. Flight control software runs on an embedded microcontroller and calculates the motor output by processing the sensor data and the reference input. In teleoperated systems, these inputs are generated by the operator via ground control stations (GCS). In autonomous systems, high-level navigation software gives reference inputs to the flight controller, and flight control software gives sensor readings and quadcopter state to high-level navigation software.

Navigation can be divided into four main sections: mapping, localization, path planning and reactive navigation. Mapping and localization are often combined in simultaneous localization and mapping (SLAM) studies. These works try to create an environment map while localizing the robots in this map. Path planning studies focused on creating an obstacle-free global plan between initial and goal configurations. These methods require map information to plan global trajectories. Local planners are needed when the map is not static or unknown. These methods use lidars, depth sensors or cameras to detect nearby obstacles and navigate the robot into obstacle-free regions. Because most navigation algorithms demand computational power, these types of stacks often run on PCs. This navigation software generates control input reference for flight control software. Figure 1.1 roughly demonstrates the communication between the flight controller and the high-level navigation stack.

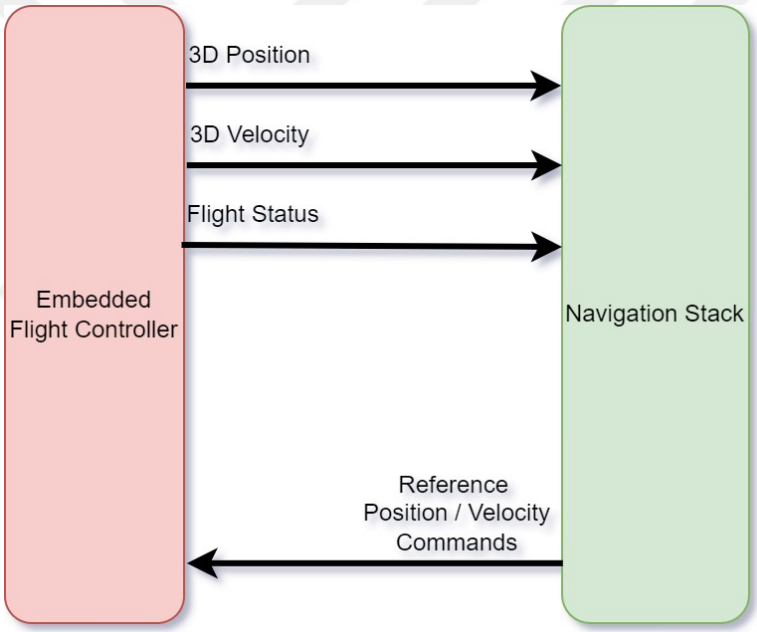

Figure 1.1 : Flight controller and navigation software communication.

In the following sections, the contributions and related work in the area are introduced. The development platform is introduced in Section 2. Developed embedded flight control software is explained in Section 3. High-level navigation software is introduced in Section 4. Experimental results are given in Section 5. In the last section, the results and future works are discussed.

1.1 Literature Review

Quadcopters are a suitable platform for researchers that works in the application and verification of various control theory problems. Control strategies in the literature can be divided into three groups: linear, nonlinear and AI-based control. Prior studies in linear control are applied PD and PID controllers [1] [2] [3] for reference tracking for a quadcopter. In [4], authors compared PID controller with LQ-based controller tracking performance. Attitude and position PID control results on linearized model further investigated in [5]. PID controllers are commonly used for their simplicity and robust performance.

Because quadcopters have nonlinear dynamics and linearized model-based control restricts quadcopter motion capabilities, nonlinear controllers are proposed. One of the popular methods in this area is sliding mode controllers. In [6,7], sliding mode controller is implemented for a quadcopter path tracking task. In recent years, backstepping [8] and adaptive control [9] methods are most preferred methods for nonlinear quadcopter control. Even though these methods have superior performance on quadcopter control tasks, they have the following drawbacks. The backstepping method suffers from low energy efficiency and requires an exact quadcopter model for robust performance, and the adaptive control method has complex adaptation rules to work properly.

In more recent studies, artificial intelligence methods are used to control quadcopters. A fuzzy logic type controller is proposed in [10] to control quadcopter. Because fuzzy tuning requires an expert, in this work, fuzzy parameters are found by trial and error. Similarly to this study, the work in [11] uses an artificial neural network to improve the performance of a PID controller. With the help of ANNs, control performance on different payloads surpassed the native PID controller performance.

In order to work with controllers, quadcopter attitude and position should be measured correctly. In quadcopters, Global Positioning System (GPS) receivers are commonly used to get aircraft position in the world frame. Inertial Measurement Units (IMUs) are often used to estimate the attitude of the aircraft. Quadcopter orientation can be described in three forms: Quaternion, Euler and Direction Cosine Matrix (DCM).

Quaternion representation stands out since it does not have a singularity problem and linear angle dependency. Pioneer studies in altitude estimation use Kalman Filter (KF) [12] based methods. Improved Kalman Filter methods used in altitude estimation of quadcopters on Extended Kalman Filter-based (EKF) [13] and Unscented Kalman Filter-based (UKF) [14] studies. In [15], the authors proposed a GPS and attitude fusion algorithm for a quadcopter inertial navigation system. Kalman Filter can be used to calculate the attitude and position information from the sensor data. Since it is a computationally expensive method for embedded systems, a different approach is proposed in this project.

High-level motion planning includes simultaneous mapping and localization (SLAM), path planning and obstacle avoidance tasks. Since SLAM studies are out of scope, readers are directed to [16,17] for further information.

Path planning is a well-studied research area, and a broad range of algorithms are proposed in previous studies. Most influential works in the continuous domain are rapidly exploring random trees (RRT) [18] and probabilistic roadmap methods (PRM) [19]. These sampling-based path planning methods are probabilistically complete and further improved with respect to speed [20,21], sampling technique [22] and kinematically feasible trajectories [23]. Later on efficient planner alternatives like Fast Marching Trees (FMT) [24] and Batch Informed Trees (BIT) [25] are proposed. Recently, Probabilistic Foam (PF) based path planning method in [26] stands out for its simplicity and safety. This method is further improved in the Goal-biased PF (GPF) [27] and Heuristic-guided PF (HPF) [28] studies. Even though these improvements improved the efficiency of PF, all methods lack path length optimality because of their bubble expansion strategy.

There is a continuous effort on dynamic path planning and real-time planning in the robotics research field. Conventional path planners demand high computational power because of collision-checking procedures. Besides, as in RRT^x [29] and $RRT^\#$ [30] methods, memorizing and pruning path segments with respect to dynamic obstacles are costly operations. Instead of planning in the whole area, CL-RRT [31] re-plans local trajectories that obey the robot kinematics. Planning area restriction reduces the

computational cost and saves memory for real-time applications. Sampling is also a key factor in path length optimality. In Informed RRT* paper [22], authors proposed a sampling restriction technique to shorten paths faster than random sampling. Even though RRT-based planners produce near-optimal paths, their real-time applications demand high computational power.

1.2 Purpose of Thesis

The thesis aims to develop and implement an embedded flight control software and proposes a novel local path replanner algorithm for tracking a trajectory while avoiding unknown obstacles in the environment using low-cost 2D lidar. First, flight control software is developed with the following features:

- Manual flight with angle and throttle references using RC transmitter
- Manual flight with angular rate and throttle references using RC transmitter
- Manual flight with angle and vertical speed references using RC transmitter
- Manual flight with horizontal and vertical speed references using RC transmitter
- Autonomous take-off
- Autonomous return to home
- Autonomous waypoint tracking
- Autonomous flight with horizontal and vertical speed references using GCS

First, the necessary hardware, electronic modules and sensors for a quadcopter setup are determined. After building the development platform, sensor drivers and calibration software are developed for the NUCLEO-H7A3 development board. The gyroscope sensor is calibrated to get more accurate angular velocity readings. The accelerometer is calibrated and filtered to remove misalignment errors from the sensor data. The magnetometer sensor is calibrated to eliminate hard and soft iron effects. These calibration stages are crucial for the filtering algorithm. A robust quadcopter

attitude estimation is achieved by fusing calibrated GPS, accelerometer, barometer and gyroscope readings.

In order to control the quadcopter for attitude, position and velocity tracking, a cascaded PID controller structure is used. The attitude control of the quadcopter is achieved using six cascaded PID controllers, and the position control of the quadcopter is performed using seven cascaded PID controllers. In total, thirteen PID controllers are used in the flight control system. The controller performance is tested on the platform with different references.

Quadcopters need navigation software for high-level navigation tasks. This study focuses on path planning, path tracking and local obstacle avoidance tasks. A coverage path planner is implemented to generate coarse trajectories for a quadcopter. Test scenarios are created in which unknown obstacles are present on the trajectories. This work applies local re-planning and smart sampling ideas to Probabilistic Foam methods to reduce computation load and increase efficiency. The proposed local planner is tasked with both tracking these paths and avoiding the obstacles. The main contributions are:

- It is the first application that PF types methods are used for local re-planning purposes.
- The proposed Angle and Heuristic biased Probabilistic Foam (AHPF) method surpassed the alternatives in execution speed and path length metrics.
- Collision checking procedures are less computationally expensive and easy to implement than RRT-based methods.
- Proposed method is very lightweight to run in real-time on embedded devices.

2. DEVELOPMENT PLATFORM

This chapter includes the details of the hardware. Platform parts can be divided into two groups. These are the thrust system and control board. The thrust system includes motors, propellers, electronic speed controllers and a LiPo battery. The control board includes a microcontroller, 6 DOF IMU, 3-axis magnetometer, GPS receiver, barometer, 433MHz telemetry module, RC receiver, Bluetooth module and 2D lidar. System diagram and connections can be seen in Figure 2.1. The control board and the software can be used on a different frame, motor and battery setups after rate and angle PID controllers are tuned for the new hardware.

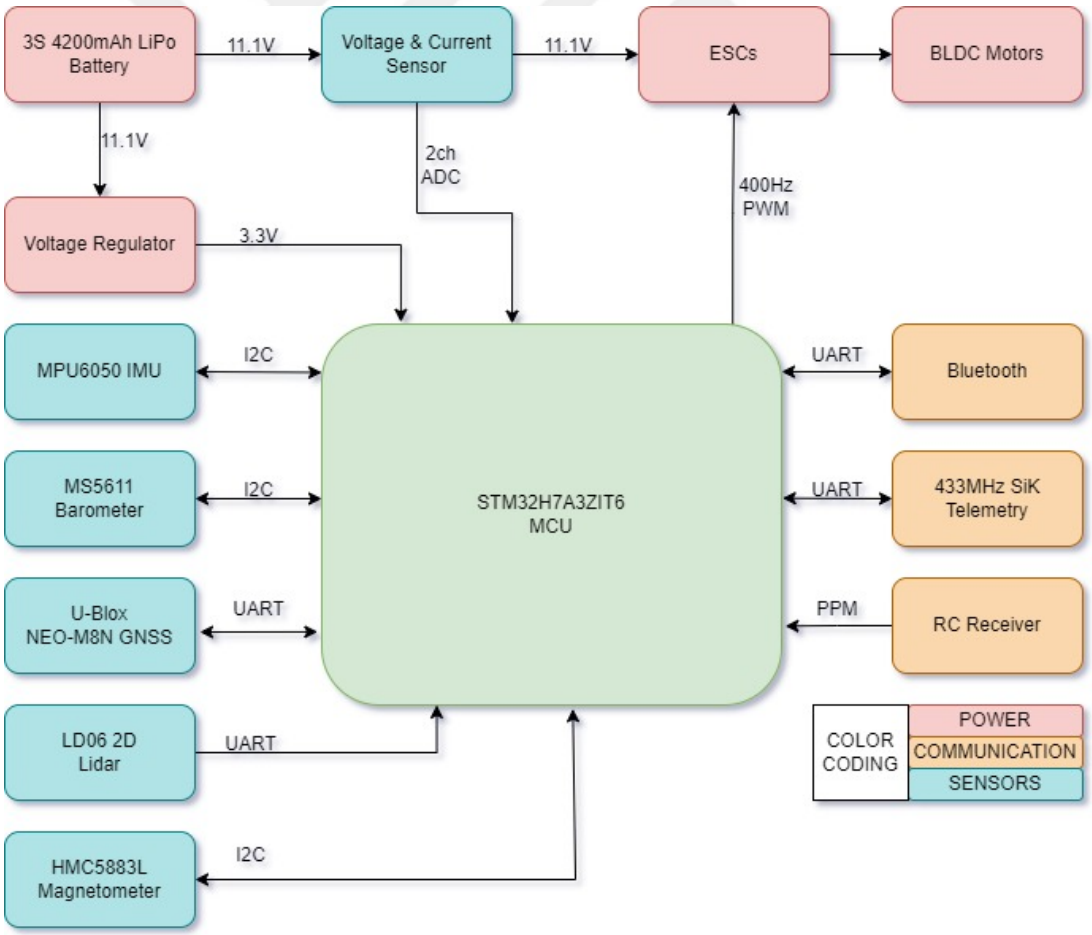

Figure 2.1 : Hardware diagram.

2.1 Thrust System

The thrust system consists of a LiPo battery and electronic speed controllers (ESC). LiPo battery powers the motors, ESCs and control board components. ESC uses a special PWM signal to drive the motors. In the standard PWM signal, the duty cycle changes from 0% to 100% and controls the motor speed. However, in this type of PWM signal, no additional information is carried from the microcontroller to ESC. For instance, 0% PWM could mean ESC connected to the MCU and MCU requests zero rpm or the PWM cable is broken, and the signal is not reached to the ESC. To differentiate this, ESC uses repeated PWM pulses ranging from 1ms to 2ms at 400Hz. 1ms means the control signal is present and requesting 0% PWM, and 2ms means 100% PWM. Furthermore, command frequency can be easily adapted from as low as 25Hz to as much as 400Hz. In this way, a simple digital communication is created between ESC and the MCU.

The control board consists of a microcontroller that runs the flight control software, sensors and communication peripherals. The microcontroller processes the data from the multiple sensors, and the user input runs the controller, determines the PWM values for each motor, and drives the motors using the ESCs. The quadcopter can be operated using an RC transmitter for manual control or over a telemetry link from a ground control station. Since there is a lot of magnetic disturbance emanating from the ESCs and the motors, sensors are placed further away from thrust system components.

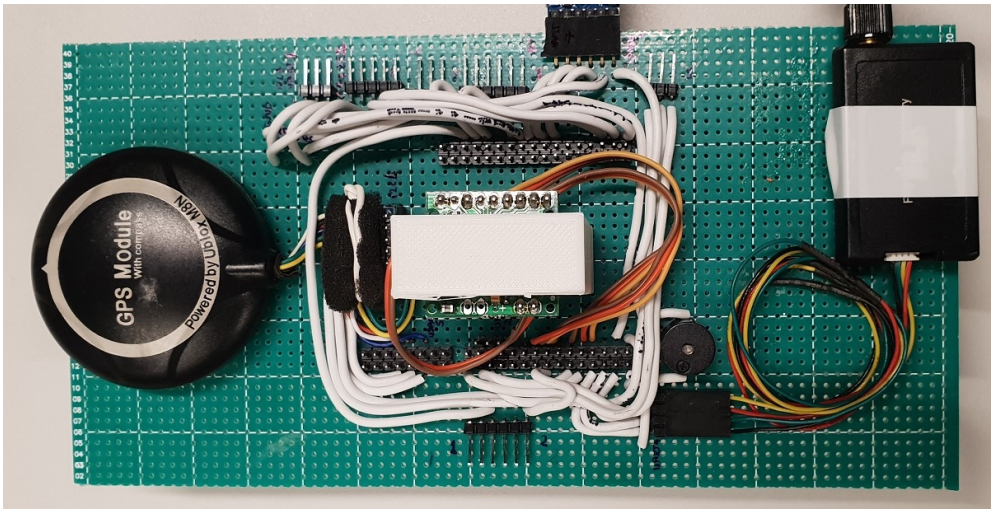

Figure 2.2 : Daughterboard.

Table 2.1 : Control board components.

No	Component Name	Technical Specifications
1	MPU6050 (IMU)	- 3 axis gyroscope - 3 axis accelerometer - 16 bit resolution - I2C interface
2	MS5611 (Barometer)	- 10cm resolution - 24bit ADC - I2C interface
3	HMC5883L (Magnetometer)	- 3 axis magnetometer - 1.3 - 8.0 Gauss measurement range - I2C interface
4	U-BLOX NEO-M8N (GPS)	- 3 GNSS reception (GPS, Galileo, Glonass) - 72channel receiver - 10Hz data output rate - UART interface
5	HC-05 (Bluetooth)	- V2.0 EDR - 1Mbps bandwidth - 10 meter range - UART interface
6	FrSky D8R-II (RC Receiver)	- 8Ch receiver - PPM support - Telemetry support
7	SiK 433Mhz Radio (Telemetry)	- Receiver sensitivity: -121dBm - Transmit power: 20dBm - Transparent serial link - Up to 250kbps air data rate - UART interface

2.2 Control Board

The control board is designed as a custom daughterboard for the development board NUCLEO-H7A3. The development board has a STM32H7A3 microprocessor and an ST-LINK V3 onboard debugger. The daughterboard is made to be compatible with all Nucleo144 boards so that different MCUs can be tested on the same sensor platform. The daughterboard can be seen in Figure 2.2. Control board components and their technical specifications can be seen in Table 2.1

The microcontroller on the control board is responsible for all the communication, sensor fusion and control software. It should be selected based on clock speed, peripheral options, pin count and memory size. Considering all the abovementioned specifications, STM327A3ZIT6 is selected for its ARM cortex M7 core, FPU unit,

which accelerates float operations and DMA feature which speeds up the peripherals without CPU load. Important features of this MCU [32] is listed below.

- 32-bit Arm core with double-precision FPU
- 2 Mbytes of Flash memory, 1.4 Mbytes of RAM
- 280Mhz, 144 5 V-tolerant I/O
- 5 DMA controllers
- 5× USART/5x UARTs
- 6× SPI, 4× I2C

The flight control software uses sensor values and user inputs to calculate necessary motor outputs. The sensors in the flight control system and their features can be listed as follows.

- **MPU6050 IMU:** This sensor is used on the system to calculate the orientation angles of the quadcopter. It is also used for estimating the short-term position of the quadcopter. It integrates a three-axis accelerometer and a three-axis gyroscope in a single package with the help of MEMS technology. The sensor communicates with the MCU using an I2C bus. The outputs are 400Hz for both the gyroscope and accelerometer.
- **MS5611 Barometer:** This sensor is used in the system to calculate the altitude and vertical velocity of the quadcopter. It has a resolution of 10cm and 24bit ADC. The sensor communicates with the MCU using I2C bus. The data output rate is 50Hz.
- **HMC5883L Magnetometer:** This sensor is used in the system to calculate the heading of the quadcopter. The quadcopter must know its heading so that it can navigate in the air. The sensor communicates with the MCU using an I2C bus. The data output rate is 50Hz.

- **U-Blox NEO-M8N GPS:** This sensor is used on the system to determine its 3D position in the air. It enables the quadcopter to naviagate itself and track a trajectory requested by the user or the path planner. It has 2-meter accuracy and 10 Hz data output rate. It communicates with the MCU using the UART bus.

- **HC-05 Bluetooth:** This module is used on the system to acquire real-time data from the sensors and the controller outputs. With this module, controller inputs can be sent with 400Hz data rate, and the sensor data can be gathered. It has 1Mbps bandwidth and communicates with the MCU using the UART bus.

- **FrSky D8R-II RC Receiver:** This module is used on the system to give manual control commands and switch between autonomous and manual mode when necessary. It has eight separate control channels and telemetry functionality. It uses PPM modulation over a single wire to transmit all eight channels to the MCU.

- **SiK 433Mhz Radio Telemetry:** This module is used on the system to communicate with the ground control station. The MCU sends its telemetry and heartbeat data to the ground control station for logging and connection checking. Mission and control commands can be sent from the ground control station over this link. The module communicates with the MCU using the UART bus. The data rate is 5Hz.

3. EMBEDDED FLIGHT CONTROL SOFTWARE

This chapter goes into detail about the flight control software. Flight control software runs on an embedded microcontroller and calculates the motor output by processing the sensor data and the user input. The developed software is written using STM32 CubeIDE integrated development environment for the chosen STM32H7A3ZIT6 MCU. Code is written in C++ language and uses STM32 HAL peripheral driver packages. HAL drivers are generated using STM32CubeMX Code generator software. A real-time scheduler algorithm is used for getting most of the performance of the MCU, running tasks without delay and monitoring task and MCU load status. A real-time scheduler is a deterministic system. It is designed for real-time applications and multitasking because it guarantees the execution time under a limit and the fixed calling frequency of the tasks. In addition, it also utilizes the MCU more efficiently. A real-time scheduler algorithm is created from scratch. Control algorithms utilize matrix operations, trigonometric conversions and computationally heavy calculations. CMSIS-DSP software package is used To access the MCU's special hardware accelerators. The flight control system's different flight modes can be seen below.

- Manual flight with angle and throttle references using RC transmitter
- Manual flight with angular rate and throttle references using RC transmitter
- Manual flight with angle and vertical speed references using RC transmitter
- Manual flight with horizontal and vertical speed references using RC transmitter
- Autonomous take-off
- Autonomous return to home
- Autonomous waypoint tracking
- Autonomous flight with horizontal and vertical speed references using GCS

3.1 Software Configuration

STM32CubeMX software is used to configure MCU clock speed, peripheral settings and pin assignments. It has a graphical user interface and lets users configure peripherals and pin assignments for all the STM32 microcontroller families.

3.2 Clock Settings

STM32H7A3ZIT6 microcontroller clock frequency can be set to any frequency between 8MHz and 280MHz using its PLL module and internal or external oscillator as their clock source. In order to use the microcontroller's full speed, the core frequency is set to 280MHz using PLL with an 8MHz external oscillator. PLL configuration of DIVM1, DIVN1, and DIVP1 is set as 1, 70, and 2, respectively. Selected MCU has different clock frequencies for different peripherals and domains. It is listed below.

- Core frequency: 280MHz
- APB1 peripheral clock frequency: 140MHz
- APB1 timer clock frequency: 280MHz
- APB2 peripheral clock frequency: 140MHz
- APB2 timer clock frequency: 280MHz
- AHB1,2 peripheral clock frequency: 280MHz

3.3 Timer Configuration

There is a total of 14 timer peripherals in the STM32H7A3ZIT6 microcontroller. TIM1, TIM2, TIM6 and TIM14 are used in flight control software. TIM1 is used for PWM generation for ESCs. TIM2 is used in the MCU performance tracking software to keep track of the task elapsed times and frequencies. TIM6 is a 400Hz periodic timer tick for the real-time scheduler software. TIM14 is used for reading the RC receiver's PPM output to get the user commands from the RC transmitter. Detailed timer settings can be found in Table 3.1

Table 3.1 : Timer module settings.

Module	Pin	Frequency	Period	Connected to
Timer1 Ch 1	PE9	400Hz	2.5ms	Esc1
Timer1 Ch 2	PE11	400Hz	2.5ms	Esc2
Timer1 Ch 3	PE13	400Hz	2.5ms	Esc3
Timer1 Ch 4	PE14	400Hz	2.5ms	Esc4
Timer2	-	1MHz	1us	-
Timer6	-	400Hz	2.5ms	-
Timer14	PF9	1MHz	1us	RC Receiver

There are four motors in the development platform, but more can be easily added. The TIM2 module is reserved for different platforms and more than four motors. TIM1 is used for sending commands to ESCs at a 400Hz rate. TIM2 is used for measuring the elapsed time of the tasks in the software; it has 1us resolution for accurate tracking and frequency measurement. TIM6 is used for periodic timer ticks for the real-time scheduler algorithm. A flag is set in every timer interrupt so that software can run the tasks with predetermined frequencies. TIM14 module is configurated for input capture function. It decodes the RC receiver's PPM signal to 8 separate channel information.

3.4 Peripheral Module's Configuration

MCU can communicate with external modules and sensors using its peripherals. I2C2, I2C4, USART3, UART4, UART5 and UART7 peripherals are used in the project. I2C2 module is used for communicating the MS5611 barometer and HMC6883L magnetometer, the I2C4 module is used for communicating the MPU6050 IMU, USART3 is used for sending debug information to the PC terminal, UART4 is used for GCS telemetry communication, UART5 is used for acquiring the GPS data from U-Blox module, and UART7 is used for transmitting real-time sensor and control data over PC to analyze and logging purposes. Detailed peripheral module configuration can be seen in Table 3.2

I2C4 module is used for communicating the MPU6050 IMU sensor. Since MCU reads the data at a high frequency, only the IMU is connected to this module. Sensor driver software initializes the sensor for 400Hz read frequency. I2C2 module is used for communicating both MS5611 and HMC5883L sensors. Both sensors can communicate at 400kHz speed. USART3 module is used to communicate with the

Table 3.2 : Peripheral module settings.

Module Name	Pin Name	Speed	Connected Module
I2C_2 - SCL	PB10	400kHz	MS5611 - HMC5883L
I2C_2 - SDA	PB11	400kHz	MS5611 - HMC5883L
I2C_4 - SCL	PF14	400kHz	MPU6050 - SCL
I2C_4 - SDA	PF15	400kHz	MPU6050 - SDA
USART3 - TX	PD8	230.4kBaud	USB Terminal - RX
USART3 - RX	PD9	230.4kBaud	USB Terminal - TX
UART4 - TX	PD1	57.6kBaud	SiK Telemetry - RX
UART4 - RX	PD0	57.6kBaud	SiK Telemetry - TX
UART5 - TX	PC12	115.2kBaud	U-Blox GPS - RX
UART5 - RX	PD2	115.2kBaud	U-Blox GPS - TX
UART7 - TX	PE8	1Mbaud	Bluetooth - RX
UART7 - RX	PE7	1Mbaud	Bluetooth - TX

Table 3.3 : ADC module settings.

Module Name	Pin Name	Resolution	Connected Module
ADC 1 - IN10	PC0	16bits	Voltage Sensor
ADC 1 - IN15	PA3	16bits	Current Sensor

PC and print the debug information to the terminal. UART4 module is used for communicating with the ground control station over a wireless telemetry link. UART5 module is used for getting the GPS data from the U-Blox GNSS module. UART7 module sends real-time sensor data to PC and receives high-level control commands.

3.5 ADC Configuration

With the ADC peripheral of the STM32H7A3ZIT6, analog sensor outputs can be converted to digital values for processing in the software. Voltage & current sensors give analog output in the 0V - 3.3V range. For measuring the battery's voltage and current draw, 2 ADC channels are activated and connected to sensor outputs. ADC configuration settings can be seen in Table 3.3

3.6 Sensor Calibrations

The flight control system's performance is linked directly to the sensor data quality and controller robustness. That is why sensor calibration is important to increase the controller's performance. All the sensors are calibrated in the factory before shipping. However, temperature and soldering stress can cause a shift in the calibration values.

In addition, some sensors, i.e. magnetometer, need to be calibrated on the platform to account for metal objects. There are a couple of different calibrations for different sensors. The gyroscope has a steady-state error, accelerometer and magnetometer have misalignment errors. All these errors can be eliminated with calibration.

Gyroscope calibration can be performed by reading the sensor output values while the platform is not moving. Gyroscope calibration is performed every time before starting the motors. The values for all three axes are summed separately for two seconds and are divided by the total number of samples to get the calibration offset value. Then these values are subtracted from the readings to get the calibrated sensor data.

Accelerometer calibration is performed to remove misalignment errors. This means that different axes on the sensor give different readings to the same input. This calibration can be performed using the earth's gravity.

For each axis of the accelerometer, positive and negative gravity measurements are gathered. Calibration can be performed with the values read from the sensors. First, the scale ratio is calculated for each axis. Then bias is calculated. With every read operation, these values are used to calculate the calibrated data. Bias calculation formula is given in 3.1, scale calculation formula is given in 3.2 and getting calibrated output formula is given in 3.3

$$B(i) = (M(i)_{max} + M(i)_{min})/(2 * g) \tag{3.1}$$

$$S(i) = (M(i)_{max} - M(i)_{min})/(2 * g) \tag{3.2}$$

$$C(i) = (R(i) - B(i)) * S(i) \tag{3.3}$$

The magnetometer is also prone to misalignment errors. In order magnetometer to work correctly, it must be calibrated in the final environment where it will be working. In an ideal environment, a magnetometer measures the earth's magnetic field vector. And the measurement output graph would be a sphere. Magnetometers have two kinds of disturbances, hard iron and soft iron. Hard iron disturbance shifts the origin of the measurement sphere. Ferromagnetic materials close to the magnetometer can cause these kinds of disturbances. Soft iron disturbance changes the sphere's shape to an

ellipsoid. Calibration is performed by rotating the platform around all its axes and recording the output values. After all the values are acquired, Bias and scale factors can be calculated using 3.1, and 3.2, and calibrated output value can be calculated using equation 3.3

3.7 Scheduler

The microcontroller used in this project can only process a single part of the code at a time. A scheduler algorithm is implemented so that multiple tasks can be run at the same time. This algorithm efficiently creates time slots for the tasks that need to run at different frequencies. It enables the software to run in real time. Processor load and individual tasks elapsed times can be monitored, and software can be optimized for strict time requirements.

In the software, Round Robin scheduler [33] is used. It uses a periodic time tick to work. A counter is used for running tasks at a different frequency. In the system, a 400Hz timer is used for the time tick. With every timer interrupt, the counter is increased, and modulo operation is used for running tasks at different time intervals. For instance, 1Hz task runs when the counter is an exact multiple of 400. For 400Hz, 200Hz and 100Hz tasks, this can be seen in Figure 3.1

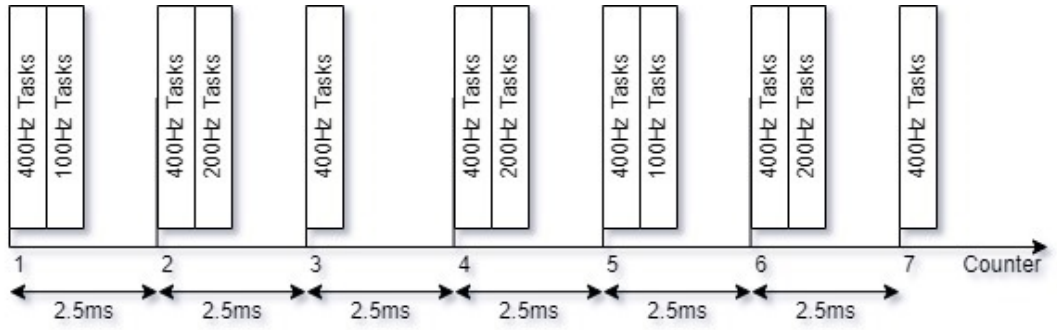

Figure 3.1 : Round Robin scheduler.

3.8 Software Flow

Software flow consists of two parts. First, the initialization part is executed. MCU peripherals are configured in this part, and sensor initializations are performed. Initialization sequence can be seen in Figure 3.2. First, MCU PLL settings are

performed, and the MCU clock frequency is set to 280MHz. Then, timer, UART, I2C and ADC modules are configured. After this step, the real-time scheduler is initialized, and the initialization part inside the scheduler is performed. This includes sensor initialization, user input initialization and controller initialization. If everything works without error, the software proceeds to run the real-time scheduler.

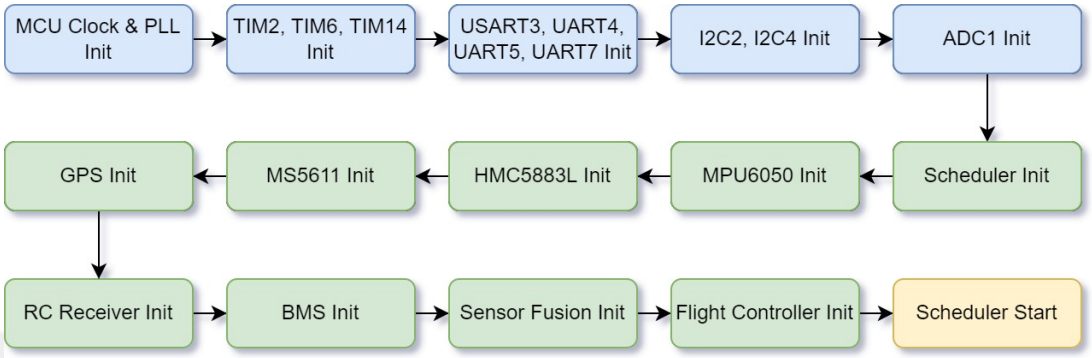

Figure 3.2 : Software initialize structure.

After the initialization is complete, the real-time scheduler starts, and the flight controller state machine algorithm runs. The state machine processes user inputs and sensor values and drives the motors according to the active flight modes. State machine flow can be seen in Figure 3.3 First, IMU data and status registers are read. The driver checks whether the sensor reports any problem or not. If sensor registers indicate that the data is reliable, then values are saved. Second, magnetometer and barometer data are read. Then, GPS position and velocity data are read from the sensor. GPS sensor also gives additional data for the reliability of the received position and velocity values. After that, battery voltage and current data are read. Current data is also used to calculate the estimated energy consumption and remaining flight time. Then, RC receiver and GCS user commands are read. These commands set the flight mode and reference values. Also, flight control software continuously checks the connectivity between the quadcopter and the RC transmitter and GCS. In case the GCS communication is lost for more than 10 seconds while in the air, the quadcopter stops its mission and starts to return to the takeoff point. If communication is regained, the user can stop the quadcopter from returning home and continue to the mission or let it continue returning.

After all the sensor data is read, the sensor fusion algorithm runs and calculates the necessary position, attitude, and velocity values. Then, according to the active flight mode, necessary PID controllers are run, and the motor output values are calculated. PID controller outputs are fed to the motor mixer software. This software converts the requested thrust values and sends them to the motors depending on the motor's position. The state machine runs the entire time the quadcopter is powered.

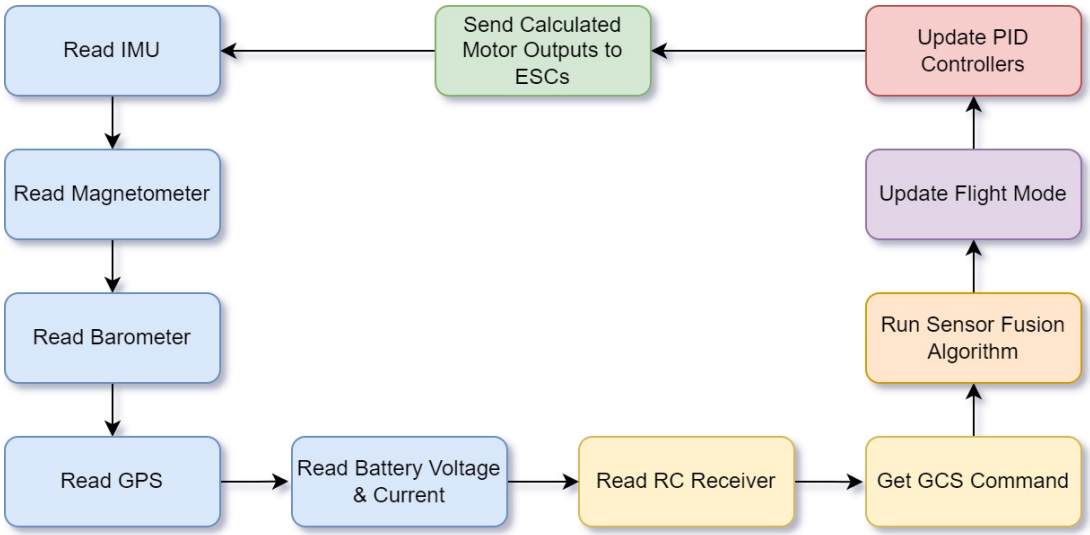

Figure 3.3 : Scheduler flow diagram.

Real-time sensor data is always transmitted to the PC for logging and real-time monitoring over Bluetooth communication. Since Bluetooth's communication range is relatively limited, low-frequency sensor data and flight status information are also transmitted over a telemetry link to the GCS for longer-distance missions. MCU performance tracking software monitors all the tasks and reports if a task is taking longer time than it is allowed. If this happens while the quadcopter is in the air, this triggers emergency land flight mode and tries to recover the quadcopter before something happens.

3.9 Sensor Fusion

Quadcopter flight control software uses a feedback control system that needs real-time sensor data to control the aircraft. Therefore, accurate position and attitude data are necessary to control the quadcopter fully. These values can be acquired using different types of sensors, including a barometer, IMU, magnetometer, and GPS. These sensors'

Table 3.4 : Sensors and their outputs.

Sensor Name	**Output**
Accelerometer	Linear Acceleration
Gyroscope	Angular Velocity
Magnetometer	Magnetic Field Strength
Barometer	Pressure Altitude
GPS	NED Position, NED Velocity

output values can be seen in Table 3.4 With the advancements in sensor technology, nowadays, these types of sensor sizes are quite small and lightweight. Thus, they can be used on a quadcopter easily.

State-of-the-art navigation solutions with accurate and precise position and attitude data can be found. However, they are not suitable for quadcopters because of their size, weight and price. Thus, lightweight and less accurate sensors are used with sensor fusion algorithms and filters to get accurate position and attitude data.

In recent years, MEMS-based sensors have been preferred in position and attitude estimation algorithms because of their small size, low-cost, high-frequency outputs. They are often used in IMUs, but they are prone to measurement errors and noises inherent in MEMS systems. MEMS-based sensors give successful results in attitude estimation but poor results in position estimates. GNSS receivers are used frequently in the open areas to acquire position and velocity data. GNSS receivers usually have output rates between 1Hz and 10Hz. This rate is slow for a dynamic system.

With the evaluation of the current state-of-the-art measurement systems according to their weight and price point, it is clear that the usage of a single system is not feasible. This problem is usually solved by fusing the data of IMU and GNSS systems. In this way, GNSS data can be used for increasing long-term accuracy, and IMU data can be used for short-term dynamic corrections. As a result of this fusion, high-frequency position and attitude data can be acquired. In this project, different sensor outputs are used in a sensor fusion algorithm to calculate the attitude and position data. Detailed information can be seen in Table 3.5

Kalman filter can be used to calculate the attitude and position information from the sensor data. Since it is a computationally expensive method for embedded systems,

Table 3.5 : Attitude and position information sources.

Data	**Used Sensor(s)**
Roll Angle	Accelerometer & Gyroscope
Pitch Angle	Accelerometer & Gyroscope
Yaw Angle	Magnetometer & Gyroscope
Roll Angular Velocity	Gyroscope
Pitch Angular Velocity	Gyroscope
Yaw Angular Velocity	Gyroscope
North Position	GNSS & Accelerometer
Earth Position	GNSS & Accelerometer
Altitude	GNSS & Barometer & Accelerometer
North Velocity	GNSS & Accelerometer
East Velocity	GNSS & Accelerometer
Vertical Speed	GNSS & Barometer & Accelerometer
Vertical Acceleration	Accelerometer

a different approach is followed in this project. Angular velocity information can be acquired from gyroscopes. It can be integrated to get the angle of an axis. However, since the gyroscope is a relative sensor, it does not know the initial position; thus, the absolute angle value cannot be calculated without external information. The gyroscope sensor also has a steady state drift problem, which means the sensor still reports angular velocity even while it stands still. Integrating this value causes drift in the angle calculations.

An accelerometer, on the other hand, is an absolute sensor. Because it continuously measures the earth's gravity vector. We can get the angle values for the roll and pitch axis from the accelerometer. However, accelerometer output is very noisy and dynamic measurements prone to errors in calculating the angle. Accelerometer and gyroscope can be used together to eliminate the errors in the measurement and get accurate attitude data. Since the accelerometer can only give angle information for the roll and pitch axis, the yaw axis still needs absolute information to calculate the heading. Magnetometer data is used with a gyroscope to get accurate heading information. The sensor fusion algorithm flow can be seen in Figure 3.4. Since the magnetometer measures the earth's magnetic field, the result is a 3D vector. This means it can also be used in roll & pitch estimation. However, in the tests done on the

platform, magnetometer readings proved more unreliable than accelerometers. Thus, magnetometer readings were used only in heading calculation.

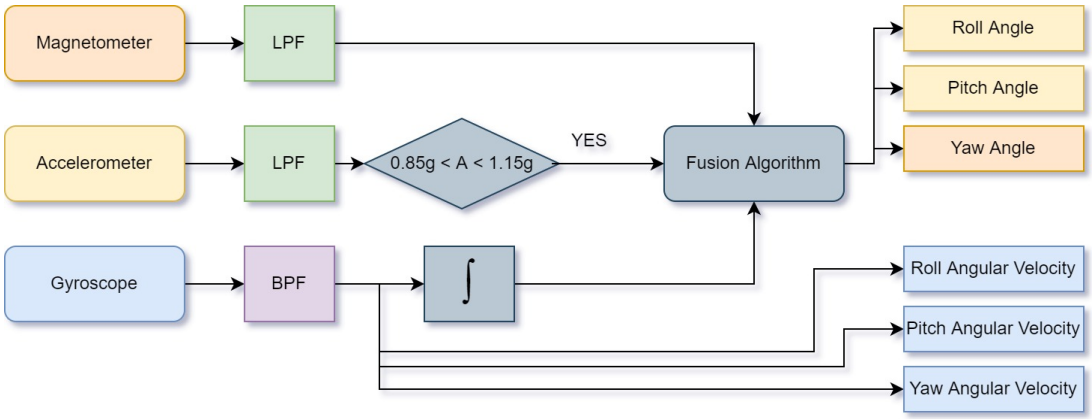

Figure 3.4 : IMU & magnetometer fusion algorithm.

Vertical position and speed values can be acquired from the accelerometer, barometer and GPS. Barometer values are reliable but have delays due to the filtering process. In order to measure vertical speed values with less delay, accelerometer values are integrated and fused with barometer speed values. GPS and barometer data are used for altitude data. Filtered accelerometer data is used for vertical acceleration. The sensor fusion algorithm for Vertical position, speed and acceleration can be seen in Figure 3.5.

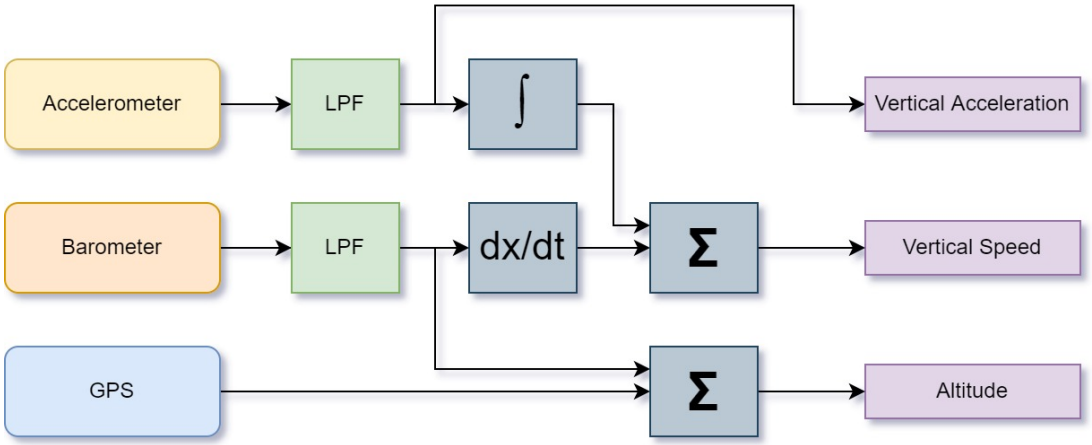

Figure 3.5 : GPS & barometer & accelerometer fusion algorithm.

Accelerometer data outputs are with respect to the quadcopter's body axes. In order to be able to be used in the controller, the values should be converted to the NED axes. To do this, a rotation matrix is used. For each axis conversion, there is a dedicated rotation matrix. Gravity vector multiplied by $\mathbf{R}_Z$ then $\mathbf{R}_Y$ then $\mathbf{R}_X$ in this order. The result is

the vector with respect to the NED frame. The formula can be seen in 3.4. After this calculation, the result of the operation always will be with respect to the earth frame.

GPS data output rate is 5Hz, but position controllers run at 200Hz. Only GPS data has not had enough speed to work reliably. Integrating the accelerometer data with respect to the NED frame can also be used for position and velocity estimates. However, since accelerometer data is very noisy, integrating the data quickly accumulates errors and will be unreliable. To solve this problem, a new method is applied. Linear accelerometer data is integrated into the GPS velocity data until the next GPS velocity data is available. When the subsequent GPS data is available, old data is discarded, and accelerometer data is integrated into the new GPS data. The algorithm can be seen in Figure 3.6. With the help of the sensor fusion algorithm, the position controller runs at 200Hz.

$$\mathbf{R}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$
$$\mathbf{R}_Y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.4}$$
$$\mathbf{R}_Z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

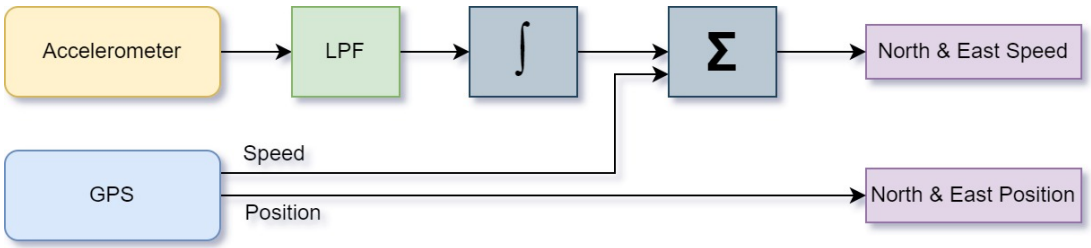

Figure 3.6 : GPS & linear accelerometer fusion algorithm.

All sensor fusion algorithms are designed to work in failsafe mode. For instance, if the GPS sensor cannot give new data, accelerometer fusion continues, and flight mode changes to emergency land. This helps recover the quadcopter and land without a crash in case of sensor failure. The same applies to the vertical position and speed references.

If the barometer fails, the fusion algorithm works with GPS and accelerometer data and tries to land the quadcopter as soon as possible.

3.10 PID Control

PID controller is a combination of three fundamental control systems, which also form its name: Proportional controller, integral controller and derivative controller. PID controller, one of the linear feedback control methods, is widely used because it is simple and easily understood. The structure of the PID controller can be seen in Figure 3.7. The controller simply compares the system output with the reference point and calculates the error. Then this error value is separately fed to the proportional, integral and derivative parts, and the results are added together to calculate the controller output. Equation 3.5 represents time domain PID controller and equation 3.6 represents frequency domain PID controller. The constants $\mathbf{K}_p$, $\mathbf{K}_i$ and $\mathbf{K}_d$ in the equations are proportional, integral, and derivative gains, respectively. **e** denotes controller error, **s** and $\mathbf{1/s}$ denotes derivative and integral in the frequency domain, respectively.

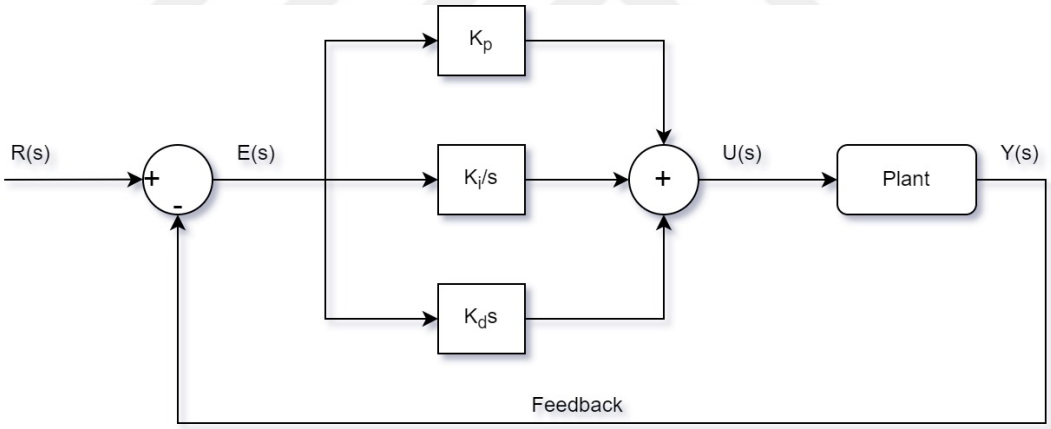

Figure 3.7 : PID controller block diagram.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{3.5}$$

$$U(s) = E(s)(K_p + \frac{K_i}{s} + K_d s) \tag{3.6}$$

Controller performance and stability are directly connected to the PID controller gains. Multiple methods exist to determine the coefficients, but manual tuning is generally

preferred. In this method, coefficients are found by trial and error until the system meets the stability criteria. This is done by changing a gain at a time and observing the system's response. This can be performed in simulation as well as on the actual platform. Besides, there are methods like Ziegler-Nichols, which uses open and closed loop step response to determine the PID values analytically. In this project, PID controller coefficients are determined by manual tuning on a quadcopter connected to a custom-made test platform.

Standard PID controller has some drawbacks. Integrators can saturate with big errors and can lead controllers to instability. This can be prevented by adding limits to the integrator so that it does not accumulate further. Also, the derivative part is prone to noise and sharp input changes. These transient states cause derivate terms to lose control of the system. This can be prevented by adding a low pass filter to the derivative input, and instead of taking the derivative of error, taking the derivative of measurement prevents instability when sharp input changes are applied. This primarily affects manual piloted control of the quadcopter. Trajectories generated from path planning algorithms are not affected by this situation because they are inherently smooth. In addition, a slope limit is added to the reference input, so the system input is filtered. This prohibits the saturation of the controller and increases stability. With the proposed changes, the final PID controller can be seen in Figure 3.8

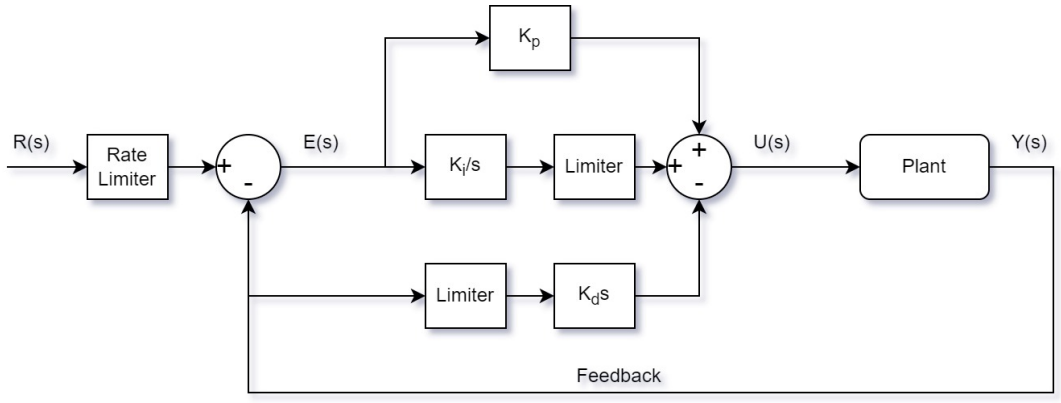

Figure 3.8 : Updated PID controller block diagram.

3.11 Controller Structure

Controller structure can be classified into two groups: position controller and attitude controller. Position control covers the control of the position and velocity of the

quadcopter with respect to NED coordinates. Attitude control covers control of the orientation of the quadcopter in the body frame. In this scope, the flight control system consists of cascade PID controllers, position and velocity controllers in the outer-loop, and attitude controllers in the inner loop. The detailed structure of the controller system can be seen in Figure 3.9

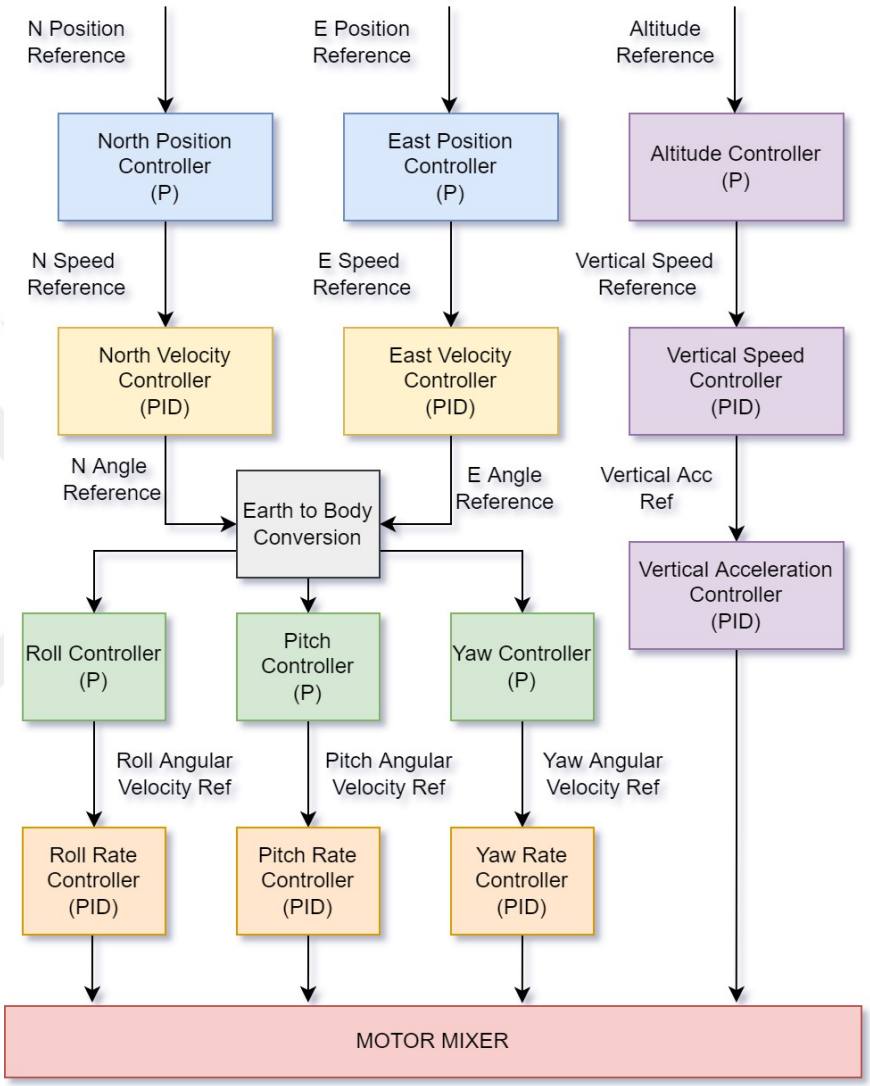

Figure 3.9 : Controller structure block diagram.

North and East position controllers are in the outer loop. These controllers take GPS coordinates as their reference and are responsible for going to a coordinate or holding the current position of the quadcopter. The outputs of these controllers are fed into North and East velocity PID controllers. These controllers are responsible for controlling the linear speed of the quadcopter. Since the quadcopter can move in any direction, N & E angle reference values must be converted from NED frame to

body frame. Earth to body conversion block calculates the frame transfer and converts the requested angle values to roll, pitch and yaw references. Roll, pitch, and yaw rate controllers are responsible for the quadcopter's angular velocity controls.

The altitude controller runs independently from the other controllers. It takes the altitude reference point and generates the necessary velocity and acceleration set points. All the calculated values are fed to the motor mixer algorithm. The motor mixer algorithm converts this request to suitable motor commands and sends commands to the ESCs.

4. HIGH LEVEL NAVIGATION SOFTWARE

Online planning and replanning methods are commonly used in unknown or dynamic environments to ensure path safety. In this chapter, a novel local path replanning algorithm is proposed. The probabilistic Foam (PF) idea is adapted for reactive planning tasks. Probabilistic Foam infrastructure is selected for several reasons. Firstly, these types of planners are lightweight, and secondly, they ensure collision-free path generation. However, they do not guarantee path length optimality. In this work, the latest achievement on the PF idea was further improved and adapted to local planning tasks.

This chapter is organized as follows. In the first section, the local planning problem is defined. The Heuristic Guided Probabilistic Foam method is introduced in the following section. In the last section, the proposed method is explained.

4.1 Problem Definition

In this thesis, the Heuristic-guided Probabilistic Foam (HPF) path planner is adapted for real-time path re-planning. This local planner is designed to follow the global path and avoid obstacles. Assume C is the n-dimensional workspace of the robot, and C_o and C_f represent occupied and free regions, respectively. Thus, the union of the occluded and accessible areas is equal to C. Bubbles are circular volumes in C_f and represented by their center point (q_c) and radius (r). The probabilistic foam method defines this radius by the distance between the nearest obstacle and the bubble center (q_c). In the next section, the HPF algorithm will be explained in detail.

4.2 Heuristic-guided Probabilistic Foam Method

In the original paper [28], which proposes Heuristic-guided Probabilistic Foam Method, the authors proved that the HPF method found shorter trajectories than the alternative methods like PF and GPF. In order to maintain safety, similarly to others,

this algorithm checks the minimum bubble radius. The efficiency of HPF stems from the heuristic function. This function prioritizes the bubbles concerning a hybrid cost value. This cost is calculated by using origin to bubble ($h(q_c)$) and bubble to goal ($g(q_c)$) distances. The first function, h, is calculated by adding the parent bubble h cost to the distance between the child and parent bubble. The function g is the euclidean distance between q_c and the goal node.

Method steps are demonstrated in Algorithm 1. Between lines 2-7, the algorithm initializes the variables. OpenList variable stores information about unvisited bubbles. The bubble has the minimum cost selected in line 9. This bubble is checked whether the goal point was inside it or not. If this bubble covers the goal point, the path is found by backward search (in line 11). If not, the bubble will try to expand on the other case. Similar to previous probabilistic foam methods, the number of children of a bubble is limited concerning its radius. This expression is shown in equation 4.1. In this formula, n represents the number of dimensions, K is the predefined constant and r_{min} is the minimum allowed bubble radius.

$$\#children = K \left(\frac{r_{bubble}}{r_{min}} \right)^{n-1} \tag{4.1}$$

This algorithm tries to create new children from the current bubble after the 15th line. Random configuration is created at the current bubble border. The bubble is added to OpenList if it is not inside another bubble (line 18) and its distance to obstacles is greater than the minimum bubble radius (line 20). The algorithm repeats until a path is found or all elements in OpenList are visited, which means the path is not found.

Even though this algorithm finds a safe path between the start position and goal position in a known map, the following drawbacks are observed in local planning in an unknown environment.

- Due to local planners using measurements within a limited range, bubbles tend to expand to unknown areas.
- Random expansion slows down finding a solution. A more informed expansion policy should be developed.

Algorithm 1 Heuristic-Guided Probabilistic Foam

procedure HPF($C_o, start_p, goal_p$)
    $r_{min} = const$
    $F = \varnothing$
    $OpenList = \varnothing$
    $R \leftarrow ExpandBubble(start_p, C_o)$
    $F.add(start_p, R)$
    while $OpenList \neq \varnothing$ **do**
        $q_p, r_p, cost_p \leftarrow OpenList.MinCostBubble()$
        if $euclideanDist(q_p, goal_p) \leq r_p$ **then**
            $path \leftarrow RetracePath(q_p)$
            return $path$
        end if
        $numChild = int(K(r_p/r_{min}))$
        for $i \leftarrow 0, numChild$ **do**
            $angle_i \leftarrow RandomAngle(q_p, goal_p)$
            $q_i \leftarrow BorderPoint(q_p, angle_i)$
            if $interior(q_i, F) == False$ **then**
                $r_i \leftarrow ExpandBubble(q_i, C_o)$
                if $r_i \geq r_{min}$ **then**
                    $F.add(q_i, r_i)$
                    $cost_i \leftarrow h(q_i) + g(q_i)$
                    $OpenList.add(q_i, r_i, cost_i)$
                end if
            end if
        end for
    end while
end procedure

These drawbacks are addressed by modifying some of the features. For readers needed more information about the HPF method, reading the paper [28] is recommended. The following section will introduce the approach to addressing local replanning and obstacle avoidance problems.

4.3 Angle and Heuristic Guided Probabilistic Foam

As mentioned before, local planners are used in unknown or partially known environments. These methods use sensory information about the obstacles and create a plan in a limited range. Their task is to avoid obstacles while following the waypoints of the global path. Global planners cannot produce obstacle-free paths in unknown environments because they do not know the obstacles' coordinates. Mapping solutions

are found costly, especially for aerial devices. In these situations, path planners and local planners should work together. In this work, the following features are added to the HPF method.

- Waypoint tracking procedure.
- Maximum bubble radius feature.
- Angle biased sampling.

4.3.1 Waypoint tracking procedure

In order to track a predefined path, intermediate points between the global path and the initial condition are needed to be selected. The procedure called GetWayPoint is implemented to calculate intermediate goal points for the local planner. The pseudocode is shown in algorithm 2. Similar to the look-ahead distance parameter in the Pure Pursuit path tracking algorithm, an offset is used to choose the waypoints ahead. It is also checked whether this waypoint was close to the obstacles or not. If obstacles block the vehicle from reaching that point, the waypoint is conveyed into a safe area. This feature is crucial for planning success.

Algorithm 2 Waypoint Following Method

procedure GETWAYPOINT($pose_{robot}, global\_path$)
  $i_{near} \leftarrow NearestIndex(pose_{robot}, global_path)$
  $w_{forward} \leftarrow global\_path[i_{near} + offset]$
  while $w_{forward} \notin C_f$ **do**
    $i_{near} \leftarrow i_{near} + 1$
    $w_{forward} \leftarrow global\_path[i_{near} + offset]$
  end while
  return $w_{forward}$
end procedure

4.3.2 Maximum bubble radius feature

Since local planners consider only close obstacles for navigation, further obstacles should be removed to speed up reactive algorithms. In robotics applications, 2D lidar sensors are often used to detect obstacles. For low-cost and low-speed applications typical lidar sensor range is about 10 meters. Originally, the HPF method is designed

to work with map representations. To demonstrate it's behaviour, the HPF algorithm is run with lidar data, and the results are illustrated in Figure 4.1. Light red circles represent bubbles, and the black circle indicates sensor range.

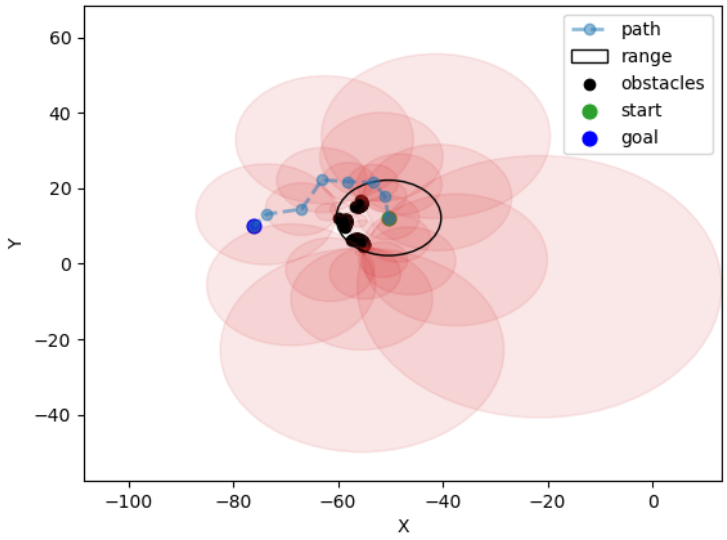

Figure 4.1 : HPF algorithm result.

From the results, it is apparent that bigger circles are created in unknown areas (outside the lidar range), and path length is relatively higher. Each bubble is limited with a maximum radius parameter (r_{max}) to address this problem. After applying this modification, the same experiment is conducted, and the results are shown in Figure 4.2. Light red circles represent bubbles; the dashed blue trajectory is the generated path. As shown in the image, this modification increases path resolution and decreases path length.

Also, the ExpandBubble function in Algorithm 1 is modified, and the modified version is shown in Algorithm 3. This function calculates the minimum distance between the bubble center (pxy) and the obstacles and limits this distance with r_{max} parameter.

4.3.3 Angle biased sampling

Selecting bubbles with respect to their heuristic costs contributes to the planning process, but the random expanding procedure slows down the planning process. Goal-biased Probabilistic Foam method solves this problem by sampling on the mixture of random points and the goal point. In order to push bubbles to expand

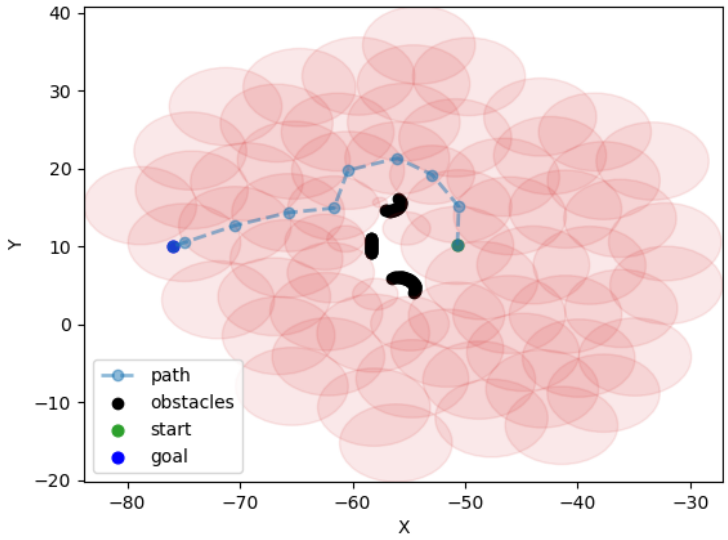

Figure 4.2 : Radius limited HPF algorithm result.

Algorithm 3 ExpandBubble Method

procedure EXPANDBUBBLE(pxy, C_{obs}, r_{max})
  $min_{dist} = \infty$
  for each obs $\in C_{obs}$ **do**
    $dist = \|obs, pxy\|$
    if $dist \leq min_{dist}$ **then**
      $min_{dist} = dist$
    end if
  end for
  return $\min(min_{dist}, r_{max})$
end procedure

towards the goal point, angle-biased sampling is proposed. This sampling strategy works as follows. First, the angle that starts at the bubble center and points to the goal point is calculated using equation 4.2.

$$goal_{angle} = tan^{-1} \frac{goal_y - bubble_y}{goal_x - bubble_x} \tag{4.2}$$

A random sample is taken from $[goal_{angle} - \frac{\pi}{2}, goal_{angle} + \frac{\pi}{2}]$ interval to expand the bubble. This feature forces the bubbles to expand towards the goal point and prevents bubbles from expanding opposite direction of the goal. This restriction is valid when we assume that obstacles are cluttered in the environment and there are no local minimum traps. After angle-biased sampling is applied, the previous experiment is

repeated, and the results are shown in Figure 4.3. Light red circles represent bubbles; the dashed blue trajectory is the generated path. These results show that bubbles are more biased to the goal point, and more informative exploration is done by the proposed method.

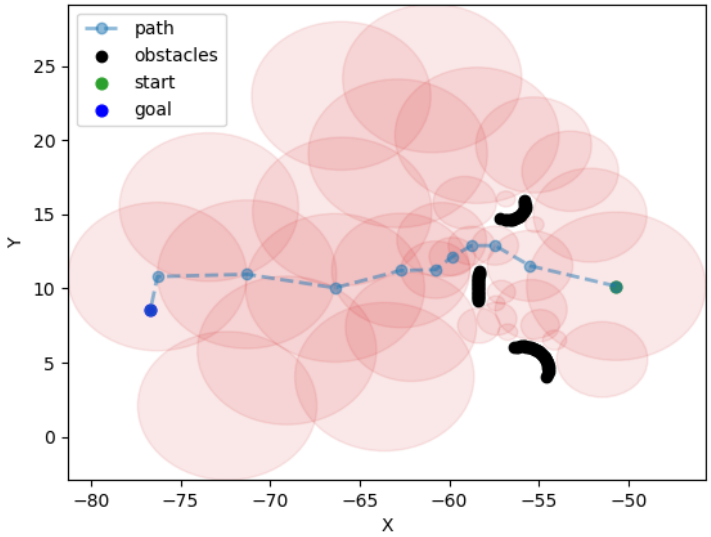

Figure 4.3 : The Proposed AHPF algorithm result.

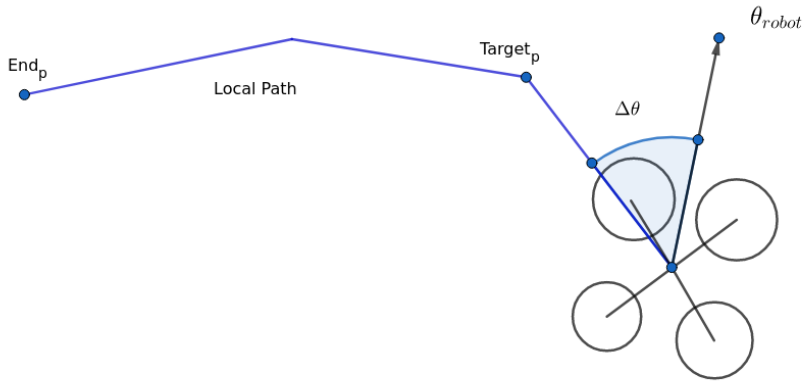

Figure 4.4 : A simple representation of a robot in 2d coordinate axes.

4.3.4 Tracking control

So far, proposed algorithm steps are explained to generate obstacle-free local trajectories. To track these trajectories, a controller is needed. In this study, a simple P-type controller is used for non-holonomic robots. Since a new plan is created with respect to obstacles at each timestep, the plan starts at the robot's current position and

ends at the waypoint on the global path. In this manner, the second node of the path is selected on each iteration to track these routes. An example scenario is illustrated in Figure 4.4. The local path is represented with blue lines. Firstly, the angle between the robot orientation (θ_{robot}) and vector points to the target node ($target_p$) is calculated and demonstrated with $\Delta\theta$ term in this figure. Secondly, a simple proportional angle controller is used to generate the angular velocity of the robot. It is assumed that the robot navigates with constant linear velocity. Angular velocity is calculated by equation 4.3. Trajectory tracking performance can be further improved by using more sophisticated control algorithms. So far, perception to control stages have been explained. In the following chapter, simulation results will be discussed.

$$w = \begin{cases} -w_{max} \leftarrow \text{if } K * \Delta\theta \leq -w_{max} \\ w_{max} \leftarrow \text{if } K * \Delta\theta \geq +w_{max} \\ \text{else} \rightarrow K * \Delta\theta \end{cases} \tag{4.3}$$

5. SIMULATION RESULTS

In this chapter, several tests are conducted to prove the efficiency of the proposed algorithm. In the first section, quantitive comparisons between the proposed method and the HPF method are conducted. In the later section, the local obstacle avoidance behaviour of the proposed algorithm is tested on the coverage path planning task.

5.1 Quantitative Comparisons

Obstacle configuration in Figure 4.3 is a challenging situation for local planning tasks. Because all the mentioned methods are based on probabilistic sampling, they produce a different result each time, so the same case is repeated a hundred times, and results are obtained. For a fair comparison, the proposed AHPF method is compared with the HPF algorithm by using the following metrics:

- Execution time: It is the planning duration measured by timers.
- Path length: It is the sum of euclidean distances between all consecutive waypoints on the path.
- Number of bubbles: Because all algorithms use bubbles to expand, the number of created bubbles is also compared.

Execution time, path length and number of bubbles metric results are illustrated in Figure 5.1, Figure 5.2, Figure 5.3 respectively. As shown in these figures, modifications made so far improved performance on these metrics. Planning duration roughly dropped eighty percent, and path length also decreased fifty percent. The number of bubbles is relatively low in the proposed method because the goal point is found faster than the HPF method. The safety of paths is not mentioned since all Probabilistic Foam methods ensure safety because of the minimum radius condition. All tests are conducted on a mini PC equipped with an Intel i5 1135G7 2.4GHz processor, 16GB ram and Ubuntu 20.04 OS.

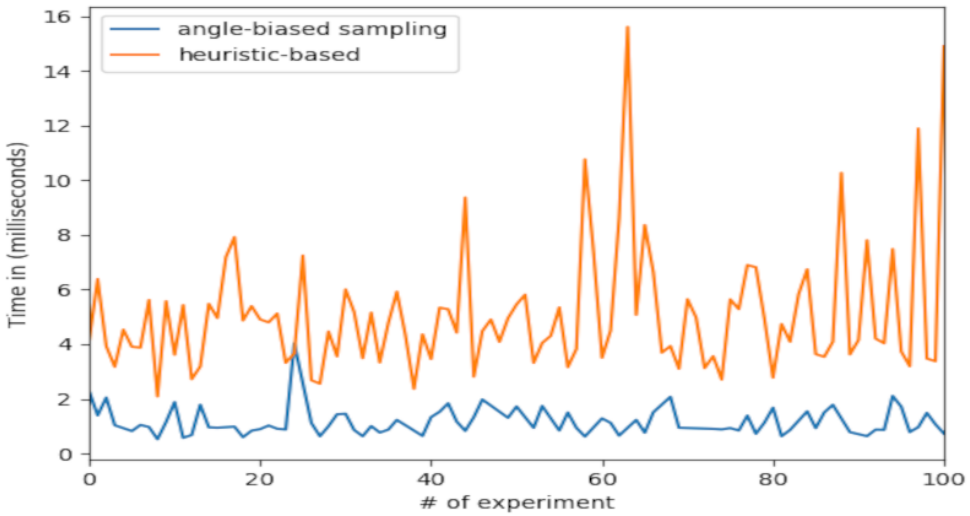

Figure 5.1 : Execution time comparison.

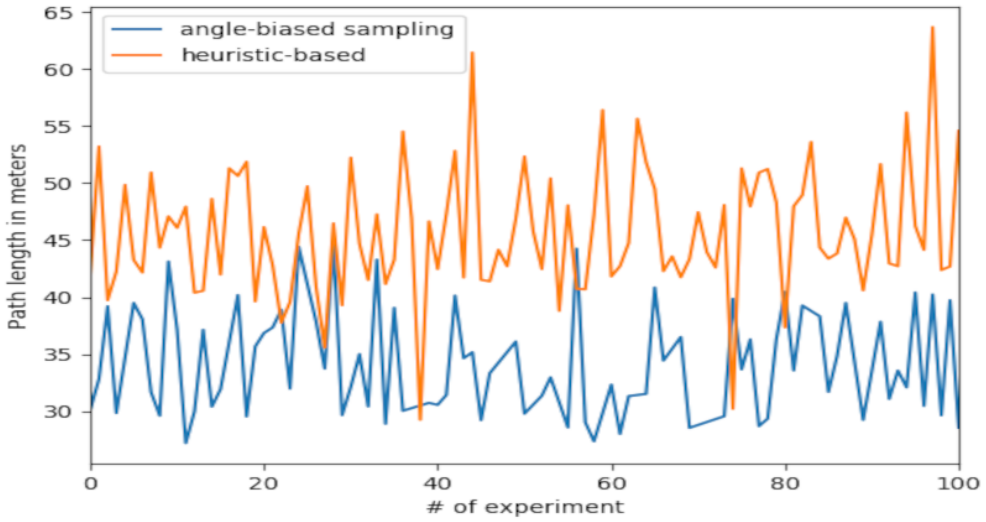

Figure 5.2 : Path length.

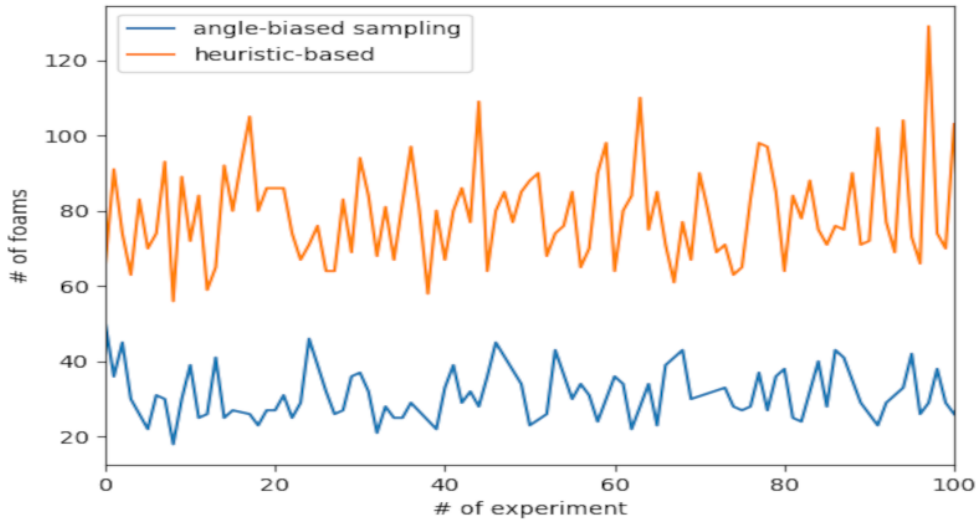

Figure 5.3 : Number of bubbles.

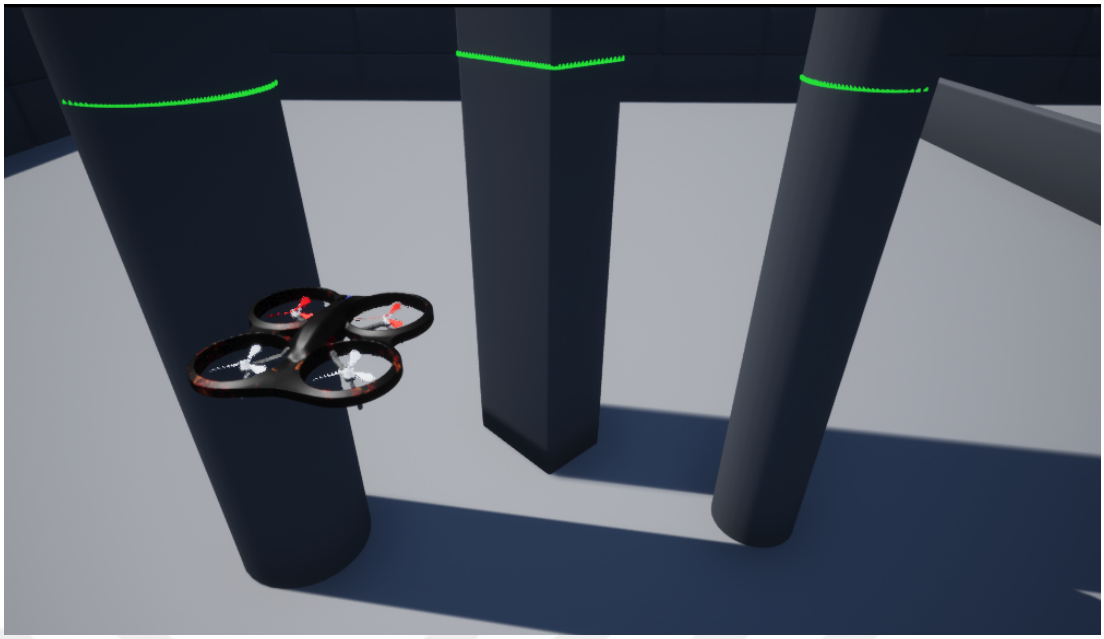

Figure 5.4 : The simulation environment.

These results also showed that the execution time is around 1ms, which means this algorithm can run at 1kHz on a medium-level computer. This means it can run real-time even on 100x slower processors, assuming a typical lidar publish frequency is around 10Hz.

5.2 CPP and Obstacle Avoidance Application on a Quadrotor

The proposed AHPF method is tested on a simulation setup which includes ROS2, Unreal Engine 4, Microsoft AirSim [34] and PX4 flight stack. Four different environments are created to test the algorithm's performance. The obstacle densities of these environments are adjusted incrementally. The unreal Engine simulation environment is demonstrated in Figure 5.4. 2D lidar readings are visualized with a green point cloud, and obstacles are arbitrary-sized 3D rectangles and circular shapes.

Quadrotor perception is simulated with a 2D lidar sensor model. This sensor is adjusted to perceive the obstacles within 10 meters range, and its field of view is determined as 360 degrees. In order to fully simulate AHPF behaviour in local planning tasks, the Grid-based Spiral Coverage planning algorithm proposed in [35] is implemented. This coverage path planner (CPP) is used to generate a global plan that covers a region by visiting all grids in this polygonal area. A polygonal area for each instance is specified,

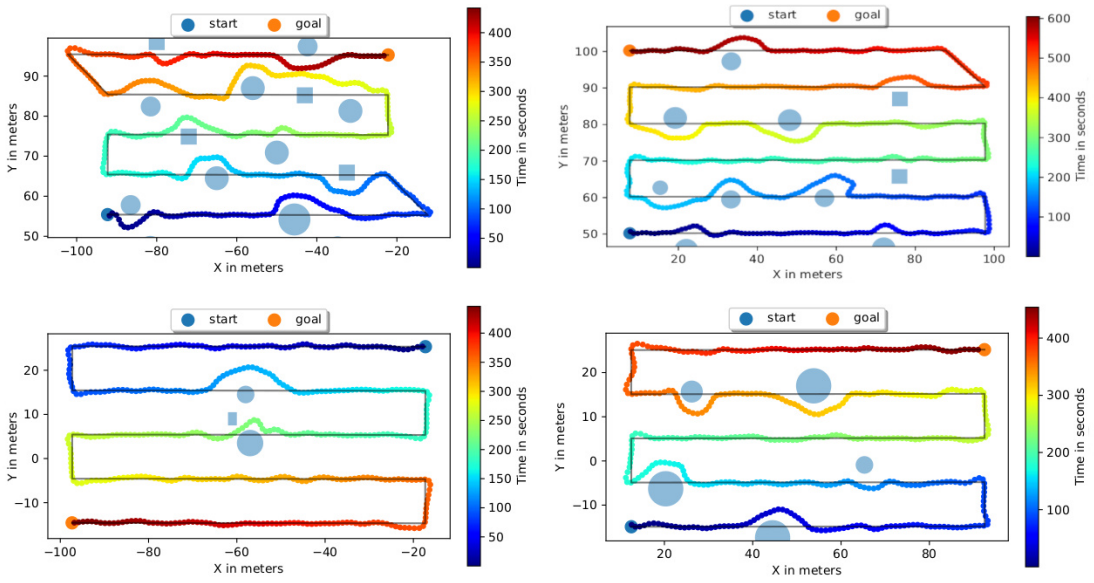

Figure 5.5 : Test results

and planned paths are shared with the AHPF algorithm. The proposed method is tasked with both tracking this path and avoiding obstacles. Since obstacles are not given beforehand to the CPP algorithm, the global path has obstacles on it. The AHPF algorithm uses a 2D lidar sensor to detect and avoid obstacles. The proposed method is tested in four different environments to measure its performance, and the test results are illustrated in Figure 5.5. In this figure, spiral coverage path planning routes are indicated with black lines. Obstacles are represented with transparent blue shapes, and the robot trajectory is plotted with respect to time.

From Figure 5.5, it is apparent that the proposed algorithm performed well on all tasks and collisions were not observed during these tests. The proposed AHPF method is able to track the global plan correctly when no obstacles are observed by the lidar sensor. When an obstacle becomes apparent (in sensor range), the algorithm plans alternative routes in obstacle-free coordinates, as demonstrated in Figure 4.3. The algorithm chose to pass in a narrow passage between obstacles in the less cluttered environment. This behaviour stems from its efficient and goal-biased sampling features.

The proposed algorithm is not tested with dynamic, moving obstacles. It is assumed that since the proposed algorithm does not know the obstacles beforehand, it could successfully avoid moving objects slower than the quadcopter. But in complex

movements and with objects faster than the quadcopter, the algorithm should be tested in a simulation environment, and necessary improvements must be made before using this algorithm with dynamic obstacles.

6. CONCLUSIONS

In this thesis, custom quadcopter hardware is built, and low-level and high-level software is developed. After building the development platform, sensor drivers and calibration software are developed for the NUCLEO-H7A3 development board. The gyroscope sensor is calibrated to get more accurate angular velocity readings. The accelerometer is calibrated and filtered to remove misalignment errors from the sensor data. The magnetometer sensor is calibrated to eliminate hard and soft iron effects. These calibration stages are crucial for the filtering algorithm. A robust quadcopter attitude estimation is achieved by fusing calibrated GPS, accelerometer, barometer and gyroscope readings.

In order to control the quadcopter for attitude, position and velocity tracking, a cascaded PID controller structure is proposed. The attitude control of the quadcopter is achieved using six cascaded PID controllers, and the position control of the quadcopter is performed using seven cascaded PID controllers. In total, thirteen PID controllers are used in the flight control system. The controller tracking performance is tested on the platform for different references.

In high-level navigation software, a novel local replanning approach is proposed for solving reactive planning problems. Because of their safe nature and low computational cost of probabilistic foam-type planners, the algorithm is developed on top of these methods. Required tunings and modifications are proposed to enable these types of planners to be able to work in real-time reactive planning tasks. As a result, the proposed method showed several advantages over the above-mentioned methods. Please note that probabilistic foam planners do not guarantee to generate optimal paths in planning. Produced paths are often found longer than RRT-type path planning methods. On the other hand, PF-type algorithms have advantages over RRT-type methods in terms of time complexity and safety. Besides, this study is the first application in which PF types methods are used for local replanning purposes.

In future studies, we will focus on implementing the proposed algorithm on an embedded system. Since a typical 2D lidar data publish rate is around 10Hz, and the proposed method can run at 1000Hz on regular hardware, our algorithm can be deployed in an embedded system for real-time applications. We will also research further improvements on the proposed method for more efficient and intelligent sampling.

REFERENCES

[1] **Bouabdallah, S.** (2007). Design and control of quadrotors with application to autonomous flying, **Technical Report**, Epfl.

[2] **Salih, A.L.**, **Moghavvemi, M.**, **Mohamed, H.A. and Gaeid, K.S.** (2010). Modelling and PID controller design for a quadrotor unmanned air vehicle, *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, volume 1, IEEE, pp.1–5.

[3] **Jiao, Q.**, **Liu, J.**, **Zhang, Y. and Lian, W.** (2018). Analysis and design the controller for quadrotors based on PID control method, *2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, IEEE, pp.88–92.

[4] **Bouabdallah, S.**, **Noth, A. and Siegwart, R.** (2004). PID vs LQ control techniques applied to an indoor micro quadrotor, *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, IEEE, pp.2451–2456.

[5] **Li, J. and Li, Y.** (2011). Dynamic analysis and PID control for a quadrotor, *2011 IEEE International Conference on Mechatronics and Automation*, IEEE, pp.573–578.

[6] **Besnard, L.**, **Shtessel, Y.B. and Landrum, B.** (2012). Quadrotor vehicle control via sliding mode controller driven by sliding mode disturbance observer, *Journal of the Franklin Institute*, *349*(2), 658–684.

[7] **Bouabdallah, S. and Siegwart, R.** (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor, *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, pp.2247–2252.

[8] **Matouk, D.**, **Gherouat, O.**, **Abdessemed, F. and Hassam, A.** (2016). Quadrotor position and attitude control via backstepping approach, *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, IEEE, pp.73–79.

[9] **Dydek, Z.T.**, **Annaswamy, A.M. and Lavretsky, E.** (2012). Adaptive control of quadrotor UAVs: A design trade study with flight evaluations, *IEEE Transactions on control systems technology*, *21*(4), 1400–1406.

[10] **Santos, M.**, **Lopez, V. and Morata, F.** (2010). Intelligent fuzzy controller of a quadrotor, *2010 IEEE international conference on intelligent systems and knowledge engineering*, IEEE, pp.141–146.

[11] **Gómez-Avila, J.**, **López-Franco, C.**, **Alanis, A.Y. and Arana-Daniel, N.** (2018). Control of Quadrotor using a Neural Network based PID, *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, IEEE, pp.1–6.

[12] **Kalman, R.E.** (1960). A new approach to linear filtering and prediction problems.

[13] **Pham, V.T.**, **Nguyen, V.T.**, **Chu, D.T. and Tran, D.T.** (2015). 15-state extended Kalman filter design for INS/GPS navigation system, *Journal of Automation and Control Engineering*, *3*(2).

[14] **St-Pierre, M. and Gingras, D.** (2004). Comparison between the unscented Kalman filter and the extended Kalman filter for the position estimation module of an integrated navigation information system, *IEEE Intelligent Vehicles Symposium, 2004*, IEEE, pp.831–835.

[15] **Lynen, S.**, **Achtelik, M.W.**, **Weiss, S.**, **Chli, M. and Siegwart, R.** (2013). A robust and modular multi-sensor fusion approach applied to mav navigation, *2013 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, pp.3923–3929.

[16] **Durrant-Whyte, H. and Bailey, T.** (2006). Simultaneous localization and mapping: part I, *IEEE robotics & automation magazine*, *13*(2), 99–110.

[17] **Bailey, T. and Durrant-Whyte, H.** (2006). Simultaneous localization and mapping (SLAM): Part II, *IEEE robotics & automation magazine*, *13*(3), 108–117.

[18] **LaValle, S.M. and Kuffner, J.J.** (2001). Rapidly-Exploring Random Trees: Progress and Prospects: Steven M. LaValle, Iowa State University, A James J. Kuffner, Jr., University of Tokyo, Tokyo, Japan, *Algorithmic and Computational Robotics*, 303–307.

[19] **Kavraki, L.E.**, **Svestka, P.**, **Latombe, J.C. and Overmars, M.H.** (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE transactions on Robotics and Automation*, *12*(4), 566–580.

[20] **Karaman, S. and Frazzoli, E.** (2011). Sampling-based algorithms for optimal motion planning, *The international journal of robotics research*, *30*(7), 846–894.

[21] **Akgun, B. and Stilman, M.** (2011). Sampling heuristics for optimal motion planning in high dimensions, *2011 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, pp.2640–2645.

[22] **Gammell, J.D.**, **Srinivasa, S.S. and Barfoot, T.D.** (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp.2997–3004.

[23] **Webb, D.J. and Van Den Berg, J.** (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics, *2013 IEEE international conference on robotics and automation*, IEEE, pp.5054–5061.

[24] **Janson, L.**, **Schmerling, E.**, **Clark, A. and Pavone, M.** (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions, *The International journal of robotics research*, *34*(7), 883–921.

[25] **Gammell, J.D.**, **Srinivasa, S.S. and Barfoot, T.D.** (2015). Batch Informed Trees (BIT): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs, *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, pp.3067–3074.

[26] **Silveira, Y. and Alsina, P.** (2016). A new robot path planning method based on probabilistic foam, *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, IEEE, pp.217–222.

[27] **Nascimento, L.B.**, **Pereira, D.S.**, **Alsina, P.J.**, **Silva, M.R.**, **Fernandes, D.H.**, **Roza, V.C. and Sanca, A.S.** (2018). Goal-biased probabilistic foam method for robot path planning, *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, IEEE, pp.199–204.

[28] **Nascimento, L.B.**, **Barrios-Aranibar, D.**, **Santos, V.G.**, **Pereira, D.S.**, **Ribeiro, W.C. and Alsina, P.J.** (2021). Safe path planning algorithms for mobile robots based on probabilistic foam, *Sensors*, *21*(12), 4156.

[29] **Otte, M. and Frazzoli, E.** (2016). RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning, *The International Journal of Robotics Research*, *35*(7), 797–822.

[30] **Arslan, O. and Tsiotras, P.** (2013). Use of relaxation methods in sampling-based algorithms for optimal motion planning, *2013 IEEE International Conference on Robotics and Automation*, IEEE, pp.2421–2428.

[31] **Arslan, O.**, **Berntorp, K. and Tsiotras, P.** (2017). Sampling-based algorithms for optimal motion planning using closed-loop prediction, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp.4991–4996.

[32] **STM32H7A3ZIT6**. `https://www.st.com/en/microcontrollers-microprocessors/stm32h7a3zi.html`, date retrieved: 03.10.2022.

[33] **Arpaci-Dusseau, R.H. and Arpaci-Dusseau, A.C.** (2018). *Operating systems: Three easy pieces*, Arpaci-Dusseau Books LLC Boston.

[34] **Shah, S.**, **Dey, D.**, **Lovett, C. and Kapoor, A.** (2018). Airsim: High-fidelity visual and physical simulation for autonomous vehicles, *Field and service robotics*, Springer, pp.621–635.

[35] **Cabreira, T.M.**, **Di Franco, C.**, **Ferreira, P.R. and Buttazzo, G.C.** (2018). Energy-aware spiral coverage path planning for uav photogrammetric applications, *IEEE Robotics and automation letters*, *3*(4), 3662–3668.

APPENDICES

CURRICULUM VITAE

Name SURNAME : Onur YILDIRIM

EDUCATION:

- **B.Sc.:** 2013, Kocaeli University, Faculty of Engineering, Mechatronics Engineering
- **M.Sc.:** 2022, Istanbul Technical University, Graduate School, Mechatronics Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2021 - 2022 Sertplas Oto Yan San. ve Tic. A.Ş - Embedded Software Team Lead
- 2020 - 2021 AirCar Corp. & Softtech A.Ş, - Senior Embedded Software Engineer / Engineering Team Lead
- 2018 - 2020 Altinay Aerospace & Advanced Technologies - Embedded Software Engineer
- 2016 - 2018 Cyber Physical Systems Laboratory, Control Engineering, ITU - Research Assistant

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Yildirim O., Özdemir A., Bogosyan O.S.** (2022). Angle and Heuristic Guided Probabilistic Foam for Local Path Replanning. *IARCE 2022*